مفهوم قواعد البيانات Database Concepts

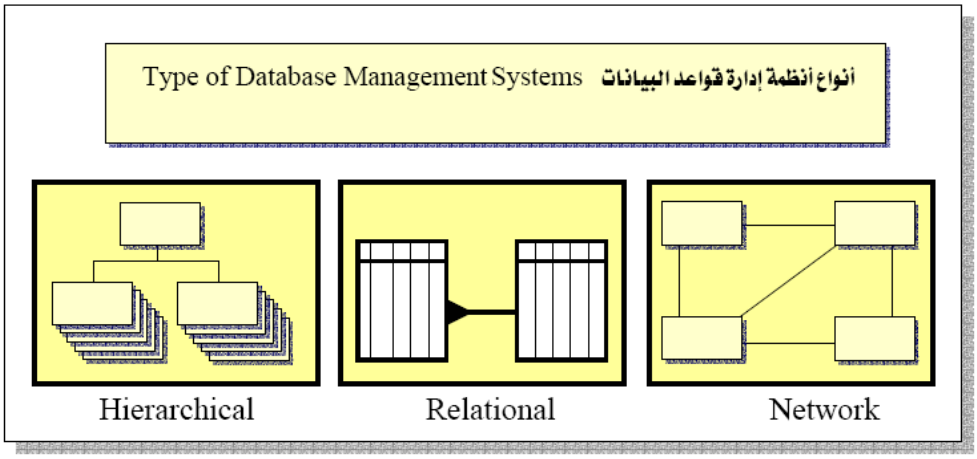
نفرض أننا نريد جمع البيانات عن المتدربين في كلية معينة ،فإنه من المعروف أن لكل متدرب بيانات مثل (رقم المتدرب الأكاديمي، اسم المتدرب ، القسم الذي ينتمي إليه ، الشعبة ..... ألخ) ، فلو جمعنا بيانات كل متدرب في بطاقة وسميناها (سجل المتدرب RECORD) وكل بيان من بيانات المتدرب في هذا السجل سميناه (حقل FIELD) معنى ذلك أننا سوف نحصل على سجلات للمتدربين . عند جمع هذه السجلات مع بعضها نحصل على قاعدة بيانات للمتدربين تسمى DATABASE . كما في الشكل رقم (١) .

مراحل تطور قواعد البيانات :

لقد مرت عملية التعامل مع البيانات وكيفية تخزينها ومعالجتها بمراحل عديدة من قبل علماء قواعد البيانات فقد تم وضع نظريات وأساليب كثيرة للتعامل مع البيانات ومنها على سبيل المثال الآتي :

- حفظ البيانات في بطاقات نشر إليكترونية Electronic Spread sheets .
- تعتمد هذه الطريقة على حفظ البيانات داخل بطاقات إلكترونية يتم التعامل معها بشكل معين وتعتبر من أقدم الأساليب للتعامل مع البيانات .

- حفظ البيانات في ملفات تسمى مخازن معبأة Filling Cabinet .
- تعتمد هذه الطريقة على تخزين البيانات في ملفات ليتم التعامل معها ، وتعتبر أيضاً من الأساليب القديمة للتعامل مع قواعد البيانات .
- حفظ البيانات في قواعد بيانات Database وتعتبر هذه الطريقة هي الأحدث بالنسبة للطرق السابقة الذكر حيث تم عمل أنظمة للتعامل مع قواعد البيانات لتسهيل عملية تخزين البيانات واسترجاعها والتعديل فيها بسهولة ودقة (معالجتها) وتسمى هذه الأنظمة أنظمة إدارة قواعد البيانات Database Mangement System (DBMS) ومن هذه الأنظمة ما هو موضح الشكل رقم (٢) .



الشكل رقم (٢)

أنواع أنظمة إدارة قواعد البيانات:

- **نظام إدارة قواعدالبيانات الهرمية** Hierarchical database Mangement system

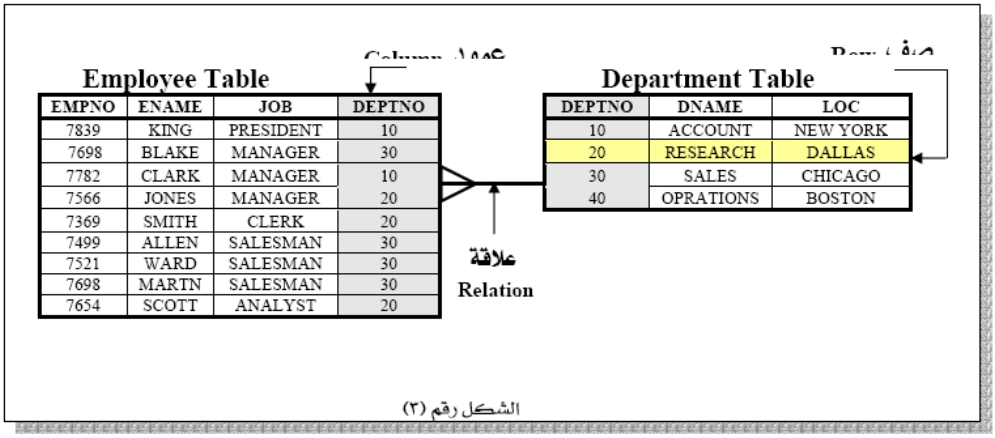
هـذا النظـام يسـتخدم في الماضـي وبخاصـة مـع أجهـزة الحاسـب الكـبيرة الـتي يطلـق عليهـا Main Frame حيث كان هذا النظام يتناسب معها بشكل جيد .

- **نظام إدارة قواعد البيانات الشبكية** Network database Mangement system

ظهر هذا النظام بعد النظام الهرمي وبخاصة بعد التوسع في أنظمة الشبكات ولكن كان هناك صعوبات كثيرة في عملية فهم وطبيعة التعامل مع البيانات كما في النظام الهرمي .

- **نظام إدارة قواعد البيانات العلائقية** Relational database Mangement system

يعتبر هذا النظام هو النظام الذي تعتمد عليه أغلب برامج قواعد البيانات مثل أوراكل لأنه من أقوى أنظمة قواعد البيانات لقدرته الفائقة على استيعاب كميات كبيرة من البيانات دون التأثير على أدائه من حيث السرعة والدقة ، ولأن هذا النظام يتمتع بالسرية والأمان لاحتوائه على نظام إعطاء الصلاحيات والحقوق لمستخدميه ولسهولته في الاستخدام والفهم وسهولة برمجة تطبيقاته .



الشكل رقم (٣)

قواعد البيانات العلائقية Relational Databse

تعتمد قواعد البيانات العلائقية على جمع البيانات في جداول بسيطة ثنائية الأبعاد يسهل فهمها تتكون من صفوف وأعمدة ، و كل عمود (Column) في الجداول عبارة عن حقل (Field) و كل صف (Row) من صفوف هذه الجداول عبارة عن سجل (Record) . وتم ربط هذه الجداول مع بعضها بروابط تسمى (Relations) ومن هنا جاءت تسميت قواعد البيانات العلائقية .

فقواعد البيانات العلائقية هي مجموعة من الجداول التي لها علاقة ما ببعضها . والشكل رقم (٣) يبين جدولين أحدهما يمثل بيانات الإدارات والآخر يمثل بيانات الموظفين ، كما يبين الشكل العلاقة بين الجدولين من خلال وجود العمود (DEPTNO) في كلا الجدولين .

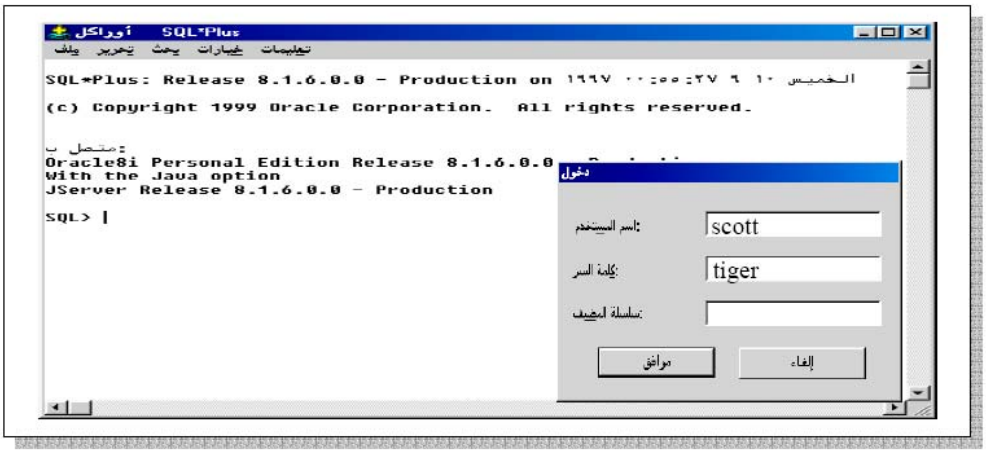
التعامل مع قواعد البيانات العلائقية Manipulate with relational database

قامت شركة أوراكل باعتماد لغة تسمى لغة الاستفسارات SQL (Structured Query Language) للتعامل مع قواعد البيانات العلائقة وهي لغة سهلة تقوم بإنشاء الأشياء (Objects) الخاصة بقاعدة البيانات مثل الجداول والتعامل معها وتقوم بعمل جميع الاستفسارات اللازمة والتي نريد أن نعرفها من قاعدة البيانات ويطلق عليها لغة (SQL) ، كما قامت شركة أوراكل أيضاً بعمل تطبيق أو بيئة تستقبل الأوامر الخاصة بلغة الاستفسارات SQL وهذه البيئة تسمى محرر الـ (SQL*PLUS) ويمكن من خلال هذا المحرر استقبال الأوامر الخاصة بلغة SQL وتنفيذها وتعديل الأخطاء الموجودة في الأمر وجميع العمليات الآخرى ، وسوف نقوم بشرح مفصل عن كل من لغة SQL والمحرر SQL*PLUS فيما يلي :

لغة الاستفسارات Structured Query Language (SQL)

هي لغة تستخدم لإصدار جميع الأوامر التي تتعلق بقاعدة البيانات ، وتنقسم هذه اللغة إلى خمسة أقسام رئيسة يمكن من خلالها إصدار الأوامر الخاصة بكل قسم ، والجدول التالي يوضح الأقسام المختلفة من هذه اللغة ووصف الأوامر لكل قسم .

القسم	الأمر	وصف الأمر
Data Retrieveal	SELECT	أمر استرجاع البيانات من جدول أو كائن
(DML) Data Manipulation Language	INSERT	أمر إضافة بيانات إلى جدول أو كائن
	UPDATE	أمر التعديل في بيانات جدول أو كائن
	DELETE	أمر حذف بيانات جدول أو كائن
(DDL) Data Definition Language	CREATE	أمر إنشاء جدول أو كائن
	Alter	أمر التعديل في جدول أو كائن
	DROP	أمر إلغاء جدول أو كائن
	RENAME	أمر تغيير الاسم جدول أو كائن
	TRUNCATE	إلغاء جزء أو بتر جزء من جدول أو كائن
Transaction Control	COMMIT	تثبيت البيانات في الجدول
	ROLLBACK	الرجوع عن تثبيت البيانات
	SAVEPOINT	الرجوع لنقطة معينة
(DCL) Data Control Language	GRANT	إعطاء الصلاحيات للمستخدمين للدخول على البيانات
	REVOKE	سحب الصلاحيات من المستخدمين



الشكل رقم (٤)

محرر (بيئة) ألـ SQL* PLUS .

الشكل رقم (٤) يبين شاشة الدخول على محرر sql*plus حيث تقوم بكتابة اسم المستخدم وهو (SCOTT) وكلمة المرور وهي (TIGER) ثم الضغط على مفتاح (موافق) وذلك للدخول على المحرر الذي يستقبل أوامر لغة الاستفسارات SQL . علماً بأن أسم المستخدم وكلمة المرور يمكن أن تتغير وذلك على حسب المستخدم هل له صلاحية الدخول أم لا ، فمن الممكن أن تدخل على المحرر باسم المستخدم (SYSTEM) وكلمة المرور (MANAGER) وفي هذه الحالة تدخل عليه وكأنك (مدير قاعدة البيانات) ، ويتكون محرر SQL*PLUS من قائمة تساعدك على تحرير الأمر والتعديل فيه وتنفيذه ، ووجود الرمز (SQL >) وهذا يشير إلى أنك تستطيع كتابة أي أمر بعده ، وهناك بعض الأوامر البسيطة التي تساعدك في كتابة وتعديل وتنفيذ الأمر ومنها على سبيل المثال .

- **SQL > EDIT**

يستخدم هذا الأمر لتعديل آخر أمر تم كتابته على محرر SQL*PLUS ، وعند تنفيذ هذا الأمر ستظهر لك شاشة (المفكرة) وبها آخر أمر تم كتابته حيث يمكن من خلال هذه الشاشة التعديل في الأمر ثم حفظه وتنفيذه مرة أخرى من خلال محرر SQL*PLUS ، ويمكن اختصار هذا الأمر فيكتب كالآتي : SQL > ED .

- **SQL > RUN**

يستخدم هذا الأمر لإعادة تنفيذ آخر أمر تم كتابته في محرر SQL*PLUS ، ويمكن كتابة هذا الأمر بالشكل التالي : SQL > R .

- **SQL > SPOOL Filename**

يستخدم هذا الأمر عندما نريد حفظ كل ما تم عمله داخل محررSQL*PLUS في ملف نصي بامتداد (LST) وذلك بغرض استرجاعها ومراجعتها ، ومن الممكن أن نحصل على نسخة مطبوعة بواسطة الأمر التالي : SQL > SPOOL OUT .

- **SQL > SAVE filename**

يستخدم هذا الأمر لحفظ الأوامر في ملف وذلك لاسترجاعها مرة أخرى وتنفيذها وهنا لابد من حفظ الملف بامتداد (sql) وذلك لنتمكن من تشغيله مرة أخرى . فإذا أردنا حفظ أمر ما داخل ملف اسمه test.sql نكتب الأمر التالي : SQL > SAVE test.sql .

- **SQL > GET filename**

يستخدم هذا الأمر لاسترجاع الأوامر التي تم حفظها بواسطة الأمر السابق . وذلك لتنفيذها مرة أخرى . فإذا أردنا استرجاع الأوامر من الملف test.sql نكتب الأمر التالي : SQL > GET test.sql .

- **SQL > START filename**

يستخدم هذا الأمر في تنفيذ الأوامر الموجودة في ملف تم حفظه بامتداد sql ، فإذا أردنا تنفيذ الأوامر الموجودة في الملف (test.sql) مثلاً نقوم بكتابة الأمر التالي :

SQL > START test.sql

- **SQL > @ filename**

هذا الأمر مثل الأمر السابق تماماً .

- **SQL > LIST**

يستخدم هذا الأمر في استعراض سطور آخر أمر تم كتابته ، ويمكن استعراض سطور معينة فمثلاً لو أردت استعراض السطور من ١ إلى ٣ نكتب الأمر كالتالي :

SQL > L 1 3

أسئلة الفصل الأول

١ – ماذا تعني الكلمات التالية (TABLE , ROW , COLUMN) .

٣ - لماذا يعتبر نظام إدارة قواعد البيانات العلائقة من أقوى أنظمة إدارة قواعد البيانات ؟

٤ – اذكر الفرق بين المحرر SQL*PLUS و لغة SQL ؟

٥ - ضع علامة صح (√) أمام الجُمل الصحيحة وعلامة خطأ (X) أمام الجمل الخاطئة ؟

- يعتبر أمر الاستعلام SELECT من أوامر محرر SQL*PLUS ويستخدم لاسترجاع البيانات. ( )
- تستخدم مجموعة أوامر DML لمعالجة بيانات الجداول . ( )
- يستخدم الأمر SPOOL <SQL لحفظ الأوامر التي نكتبها في المحرر داخل ملف ليسهل مراجعتها . ( )
- يستخدم الأمر L 2 4 لعرض السطور مبتدئ من السطر الثاني وحتى السطر الرابع ( )
- يستخدم الأمر START >SQL لتنفيذ عدة أوامر تم حفظها من قبل ( )
- يعتبر الأمر RUN من أوامر لغة الاستفسارات SQL ( )

الصيغة (الشكل) العامة لجملة SELECT .

```
SELECT      * or   Columns [alias]
FROM        Table
WHERE       condition or conditions
ORDER BY    Column or Alias    [ASC or DESC]   ;
```

تفسير الصيغة العامة :

SELECT	تستخدم في بداية الأمر لاسترجاع البيانات من الجداول .
*	هذا الرمز يستخدم عند استرجاع جميع الحقول من الجدول .
Columns	أسم الحقل أو الحقول المراد استرجاعها من الجدول .
Alises	الاسماء المستعارة للحقول .
FROM	تستخدم للإعلان عن اسم الجدول .
Table	أسم الجدول المراد استرجاع البيانات منه .
WHERE	تستخدم للإعلان عن الشرط أو الشروط .
Conditions	الشرط أوالشروط اللازمة لحصر البيانات الآتية من الجدول .
ORDER BY	تستخدم للإعلان عن كيفية ترتيب البيانات المسترجعة من الجدول .
Column or Alies	أسم الحقل أو الحقول أو الأسماء المستعارة المراد الترتيب بها .
;	فاصلة منقوطة للإعلان عن نهاية الأمر .

متطلبات وإرشادات كتابة جمل SQL .

يوجد هناك بعض الإرشادات التي يجب مراعاتها عند كتابة جملة SQL لتكون الجملة صحيحة وقابلة للتنفيذ ، وهذه الإرشادات هي :

١ - يمكن كتابة مكونات جملة SQL بالأحرف الكبيرة أو الصغيرة فهذا لا يؤثر على سلامة الجملة وذلك لأن جملة SQL غير حساسة للحروف Not Case Sensitive .

٢ - ينفصل بين أسماء الحقول باستخدام الفاصلة (,) .

٣ - يمكن كتابة جملة SQL في عدة سطور فهذا لا يؤثر في صحة الجملة .

٤ - لا يمكن فصل الكلمات المحجوزة للغة أو اختصارها ، والكلمات المحجوزة تسمى Keywords وهي مثل (SELECT , FROM , WHERE , ORDER BY) .

٥ - يفضل كتابة الجملة على أسطر ليسهل قراءتها وفهمها .

٦ - لابد من الإعلان عن نهاية الجملة بواسطة (;) .

٧ - ملحوظة : أوامر محرر SQL*PLUS لا يوضع بعدها الفاصلة المنقوطة (;) .

لتنفيذ جملة SQL :

لتنفيذ جملة SQL من الممكن استخدام إحدى الطرق التالية :

١ - نضع الفاصلة المنقوطة (;) في نهاية الجملة .

٢ - نضع علامة (/) في نهاية الجملة عند مؤشر SQL > .

٣ - نكتب الأمر (RUN) عند مؤشر SQL > .

ولفهم طبيعة جملة SQL وكيفية تنفيذها ، فسوف نقوم بعرض أمثلة لحالات الجملة ونتائج تنفيذها لنصل من خلال الأمثلة إلى الفهم المطلوب :

قبل أن نبدأ في عرض الأمثلة هل تتذكر الجدولين المرسومين في الفصل السابق ، كان الجدول الأول عبارة عن جدول يحتوي على بيانات الموظفين ويسمى (EMP) ، والجدول الثاني كان يحتوي على بيانات الإدارات ويسمى (DEPT) (راجع صفحة ١ -٦ شكل (٣)) وهنا سوف نستخدم هذين الجدولين بشكل أساسي ولذلك لابد من مراجعتهما ومعرفة أسماء الحقول في كلا الجدولين .

مثال (١) : عرض جميع الحقول من جدول الإدارات **DEPT** .

```
SQL> SELECT *
  2  FROM    dept ;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

في هذا المثال نقوم بعرض جميع الحقول والبيانات الموجودة في جدول الإدارات DEPT الذي يحتوي على الأعمدة التالية (DEPTNO , DNAME , LOC) وذلك باستخدام الرمز (*) والذي يعني إظهار جميع حقول الجدول ، لاحظ أن أسماء الحقول دائماً تظهر بالحروف الكبيرة .

مثال (٢) : عرض حقول معينة من جدول الإدارات **DEPT** .

```
SQL> SELECT  deptno , dname
  2  FROM dept ;
```

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

يوضح المثال السابق كيفية إظهار عدة حقول معينة من الجدول ، ونلاحظ هنا بأن أسماء الحقول يفصل بينها الفاصلة (,) ، فالمثال يقوم بعرض جميع أرقام الإدارات وأسمائها فقط من جدول الإدارات (DEPT) .

استرجاع الحقول باسماء مستعارة (Aliases) :

نستخدم طريقة الاسماء المستعارة (Aliases) عندما نريد إظهار الحقل باسم غير اسمه الموجود في الجدول وذلك لتوضيح معنى الحقل مثلاً . وهناك ثلاث طرق لإظهار الحقول باسماء مستعارة :

١ - استخدام كلمة (AS) بين أسم الحقل والاسم المستعار .

٢ - استخدام المسافة (Space) بين أسم الحقل والاسم المستعار .

٣ - استخدام علامة التنصيص المزدوجة التالية (" ") عندما يكون الاسم المستعار أكثر من كلمة ، والمثال التالي يوضح ذلك .

مثال (٣) : عرض حقول باسماء مستعارة من جدول الموظفين .

```
SQL> SELECT ename AS name , sal salary , job "employee job"
  2  FROM   emp ;
```

NAME	SALARY	employee job
SMITH	800	CLERK
ALLEN	1600	SALESMAN
WARD	1250	SALESMAN
JONES	2975	MANAGER
MARTIN	1250	SALESMAN
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
SCOTT	3000	ANALYST
KING	5000	PRESIDENT
TURNER	1500	SALESMAN
ADAMS	1100	CLERK

يوضح المثال السابق كيفية استخدام الطرق المختلفة لإظهار الحقول باسماء مستعارة ، فنلاحظ هنا أن أسم الحقل ename قد ظهر في النتيجة باسم NAME بحروف كبيرة وكذلك حقل الراتب SALARY ، كما نلاحظ بأن الاسم المستعار الموجود بين علامتي التنصيص المزدوجة (" ") قد ظهر في النتيجة كما هو دون تحويله إلى الأحرف الكبيرة .

استخدام العمليات الحسابية وأولويات تنفيذها مع جملة SELECT :

من الممكن إجراء عمليات حسابية على الحقول العددية للحصول على معلومة معينة فمثلاً إذا أردنا إظهار الموظفين ورواتبهم في سنة فإننا نقوم بضرب راتب كل موظف في العدد ١٢ بشكل التالي (SAL*12) ، وكذلك عند إظهار إجمالي الراتب لكل موظف بعد إضافة ٥٠٠ ريال عليه (SAL + 500) . لاحظ أن العمليات الحسابية على الحقول لا تؤثر على البيانات المخزنة داخل الجدول .

المعاملات الحسابية التي تستخدم في العمليات الحسابية Arithmetic Operators

١ - الجمع (+) .

٢ - الطرح (-) .

٣ - الضرب (*) .

٤ - القسمة (/) .

يمكن استخدام المعاملات الحسابية في جميع أجزاء جملة SQL ماعدا الجزء الخاص بـ FROM والأمثلة التالية توضح كيفية إجراء العمليات الحسابية على الحقول .

مثال (٤) : عرض رواتب الموظفين السنوية من جدول الموظفين .

```
SQL> SELECT ename , sal , sal*12  "annual salary"
  2  FROM emp ;
  3
```

ENAME	SAL	annual salary
SMITH	800	9600
ALLEN	1600	19200
WARD	1250	15000
JONES	2975	35700

يبين المثال السابق كيفية استخدام العمليات الحسابية للحصول على رواتب الموظفين السنوية وذلك بضرب راتب كل موظف في ١٢ شهر .

أولويات تنفيذ العوامل الحسابية Operator Precedence

عند إجراء عملية حسابية كبيرة على حقل من الحقول لابد أن تعرف كيفية حسابها ولمعرفة ذلك لابد أن تعرف أولوية تنفيذ العوامل داخل جملة SQL فهي تنفذ بالترتيب التالي :

١ - أولوية تنفيذ العمليات الحسابية للضرب والقسمة ثم للجمع والطرح .

٢ - العمليات من نفس الأولوية تنفذ من اليسار إلى اليمين .

٣ - عند وجود الأقواس في العمليات الحسابية يكون ما بداخلها له الأولوية وينفذ حسب الفقرة رقم (١) .

لاحظ الفرق بين العمليتين التاليتين :

1 – 100*(40+10) = 100*50=5000 .
2 – (100*40)+10 = 4000+10= 4010 .

والمثال التالي يوضح أولوية التنفيذ للعوامل الحسابية .

مثال (٥) : عرض رواتب الموظفين السنوية من جدول الموظفين .

```
SQL> SELECT  ename , sal , 12*sal+100
  2   FROM emp;

ENAME          SAL      12*SAL+100
---------- ---------- ----------
SMITH             800        9700
ALLEN            1600       19300
```

لاحظ أولوية التنفيذ :
العملية رقم (١) تنفذ أولاً ثم العملية رقم (٢) .

مثال (٦) : عرض رواتب الموظفين السنوية من جدول الموظفين .

```
SQL> SELECT  ename , sal , 12*(sal+100)
  2   FROM emp;

ENAME          SAL      12*(SAL+100)
---------- ---------- ----------
SMITH             800        10800
ALLEN            1600        20400
```

لاحظ أولوية التنفيذ :
العملية رقم (١) تنفذ أولاً لوجود الأقواس ثم العملية رقم (٢) .

بملحوظة الفرق بين المثالين السابقين ، نجد أن النتيجة قد اختلفت تماماً وذلك لوجود الأقواس في المثال رقم (٦) فتم حساب ما بداخل الأقواس أولاً .

استخدام أداة الربط بين الحقول (||) Concatenation .

لعمل سلسلة من الحقول نقوم بربط حقلين أو أكثر باستخدام أداة الربط (||) والتي تسمى Concatenation ، ويكون ناتج الربط بين الحقول هو حقل واحد فقط ، ومن الممكن أن نربط مع الحقول نص معين نضعه بين علامتي تنصيص فردية (' ') ، والمثال التالي يوضح ذلك .

```
SQL> SELECT ename, job , ename||job as "employees"
  2  FROM emp ;

ENAME      JOB              employees
---------- ---------        -------------------
SMITH      CLERK            SMITHCLERK
ALLEN      SALESMAN         ALLENSALESMAN
WARD       SALESMAN         WARDSALESMAN
JONES      MANAGER          JONESMANAGER
MARTIN     SALESMAN         MARTINSALESMAN
```

في المثال السابق تم ربط حقلي الاسم والوظيفة بإدارة الربط (||) وقد ظهر هذان الحقلان كأنهما حقل واحد باسم مستعار employees يجمع بين الاسم والوظيفة .

```
SQL> SELECT ename, job , ename||' is a '||job as "employees"
  2  FROM emp ;

ENAME      JOB              employees
---------- ---------        -------------------
SMITH      CLERK            SMITH is a CLERK
ALLEN      SALESMAN         ALLEN is a SALESMAN
WARD       SALESMAN         WARD is a SALESMAN
JONES      MANAGER          JONES is a MANAGER
MARTIN     SALESMAN         MARTIN is a SALESMAN
```

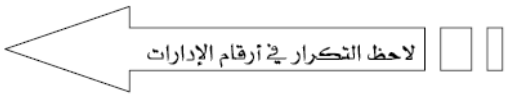
في المثال السابق تم ربط الحقلين الاسم والوظيفة وبينهما نص هو (is a) باستخدام أداة الربط (||) ، فظهر الناتج كما هو واضح بالمثال .

استخدام عبارة DISTINCT لمنع تكرار السجلات :

عند إظهار محتويات الجدول نجد تكرار بعض القيم للحقل الواحد دون فائدة ، فمثلاً لو طلب منك معرفة أرقام الإدارات التي ينتمي إليها الموظفين في جدول الموظفون (EMP) ، فإنك بالطبع سوف تكتب هذا الأمر .

```
SQL> SELECT deptno
  2  FROM emp ;

 DEPTNO
------------
        20
        30
        30
        20
        30
        30
```



وبالنظر إلى النتيجة سوف تجد أن هناك تكراراً في الأرقام دون فائدة كما هو واضح ، ولمنع هذا التكرار نستخدم كلمة (distinct) مباشرةً بعد كلمة SELECT كماهو مبين في المثال التالي :

```
SQL> SELECT  DISTINCT deptno
  2  FROM emp ;

  DEPTNO
  ----------
        10
        20
        30
```

لاحظ أن أرقام الإدارات لم تتكرر وذلك لاستخدام كلمة DISTINCT .

إظهار البناء الداخلي للجداول باستخدام الأمر DESCRIBE (DESC) .

لإظهار معلومات حول أسماء الحقول وأنواعها الموجودة في جدول معين أي لإظهار البناء الداخلي للجدول نستخدم الأمر (DESCRIBE) والذي يمكن اختصاره إلى الأحرف التالية (DESC) .

```
SQL> DESC emp ;

Name                                      Null?    Type
----------------------------------------- -------- ------------
EMPNO                                     NOT NULL NUMBER(4)
ENAME                                              VARCHAR2(10)
JOB                                                VARCHAR2(9)
MGR                                                NUMBER(4)
HIREDATE                                           DATE
SAL                                                NUMBER(7,2)
COMM                                               NUMBER(7,2)
DEPTNO                                             NUMBER(2)
```

في المثال السابق تم إظهار أسماء الحقول الموجودة في جدول الموظفين وأنواعها وبعض المعلومات الخاصة بكل حقل مثل: هل نوع الحقل نصي أو تاريخ أو رقم ؟ وما هو طوله ؟

التعامل مع القيمة NULL .

ماذا تعني القيمة NULL ، هذه القيمة تسمى قيمة غير معروفة أو قيمة خاوية بمعنى أنها لاتساوي الصفر ولا مسافة ولا أي رقم أو نص ، فمثلاً هناك حقل في جدول الموظفين اسمه COMM هذا الحقل يخزن به قيمة تكليف الموظف بمهمة أو تكليفه بعمل إضافي يأخذ عليه أجر ، فهناك موظفون يأخذون بدل تكليف وموظفون آخرون لايمكن تكليفهم أي لايأخذون بدل تكليف إطلاقاً وفي هذه الحالة يتم ترك حقل التكليف COMM خالياً ليس به أي قيمة ونطلق على القيمة الخالية هذه NULL .

المثال التالي يبين الموظفين الذين لا يكلفون بعمل إضافي أي لا يأخذون بدل تكليف .

```
SQL> SELECT ename, job , sal , comm
  2  FROM emp ;

ENAME      JOB              SAL      COMM
---------- ---------- ---------- ----------
SMITH      CLERK            800
ALLEN      SALESMAN        1600        300
WARD       SALESMAN        1250        500
JONES      MANAGER         2975
MARTIN     SALESMAN        1250       1400
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
SCOTT      ANALYST         3000
KING       PRESIDENT       5000
TURNER     SALESMAN        1500          0
```

NULL

يمكنك أن تلاحظ مثلاً الموظف SMITH لايأخذ بدل تكليف ولذلك تركت قيمة COMM خالية .

عند إجراء أي عملية حسابية عليها يكون النتائج دائماً (قيمة خالية NULL) ، يمكنك أن تلاحظ ذلك في المثال التالي :

```
SQL> SELECT ename, job , sal , 12*sal+comm
  2  FROM  emp ;

ENAME      JOB              SAL      12*SAL+COMM
---------- ---------- ---------- ----------
SMITH      CLERK            800
ALLEN      SALESMAN        1600      19500
```

NULL

أسئلة الفصل الثاني

١ – اكتب جملة استعلام لعرض البناء الداخلي لجدول الإدارات ، ثم اكتب جملة استعلام لعرض جميع بياناته ، بحيث تظهر النتيجة كالتالي :

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

٢ – اكتب جملة استعلام لعرض أسماء ووظائف وتواريخ تعيين وأرقام الموظفين بحيث يظهر رقم الموظف أولاً ؟

٣ – اكتب جملة استعلام لعرض وظائف الموظفين بدون تكرار ؟ بحيث تظهر النتيجة كالتالي :

JOB
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

٤ – اكتب جملة استعلام لعرض أرقام الموظفين وأسمائهم ووظائفهم مع تغيير أسماء الأعمدة كما هي في النتيجة التالية :

EMPLOYEE_NO	EMPLOYEE NAME	JOBS
7369	SMITH	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER

في الفصل السابق تم التعرف على جملة SELECT الأساسية والتي من خلالها يتم استرجاع البيانات من الجداول ، ومن الملاحظ أنه عند كتابة جملة SELECT في الفصل السابق كان الناتج دائماً يكون جميع الصفوف الموجودة بالجداول ، فلو أردنا استرجاع أسماء الموظفين الذين يعملون بوظيفة معينة وتكون هذه الاسماء مرتبة تصاعدياً أو تنازلياً حسب راتب كل منهم ، فماذا نعمل حتى نستطيع استرجاع الصفوف المطلوبة وترتيبها دون غيرها ؟ .

في هذا الفصل سوف نستخدم جملة الشرط (WHERE) لحصر الصفوف على أساس شرط معين، وأيضاً ترتيب الصفوف تصاعدياً أو تنازلياً باستخدام جملة (ORDER BY) .

جملة الشرط (WHERE) :

تُكتب هذه الجملة مباشرةً بعد جملة (FROM) وتستخدم في حصر البيانات على أساس شرط أو شروط معينة ، ويتكون الشرط من طرفين بينهم معامل مقارنة Comparison Operator ، وعند تحقق الشرط أي إن الشرط (TRUE) فإن جملة SELECT يكون لها ناتج ، أما إذا كان ناتج الشرط غير متحقق (FALSE) فإن جملة SELECT لا يكون لها أي ناتج وتظهر رسالة (No Row Selected) ومعناها لم يتم تحديد أي صف .

مكونات جملة الشرط WHERE :

يمكن أن تحتوي جملة الشرط (Where) على ما يلي :

- أسماء حقول Columns .
- معاملات مقارنة Comparison Operators .
- قيم ثابتة سواء كانت عددية أو نصية .
- تعبيرات حسابية .

يجب مراعاة الآتي عند كتابة جملة الشرط .

- عند استخدام قيم نصّية أو قيم تُعبر عن تاريخ لابد من وضعها داخل علامة التنصيص الفردية (' ') .
- في حالة استخدام القيم النصّية لابد من مراعاة حالة الأحرف كبيرة أم صغيرة .
- في حالة استخدام قيم تعبر عن تاريخ لابد من مراعاة صيغة التاريخ المستخدمة (FORMAT) علماً بأن الصيغة الأساسية للتاريخ داخل لغة SQL هي كالتالي : (DD-MON-YY) حيث إن (DD تعبر عن اليوم ، MON تعبر عن الشهر ، YY تعبر عن السنة) .

جملة الترتيب (ORDER BY) :

تستخدم هذه الجملة لترتيب الصفوف الناتجة ترتيباً تصاعدياً أو تنازلياً ، وتكتب دائماً في نهاية جملة SELECT .

متطلبات وإرشادات كتابة جملة الترتيب ORDER BY .

يجب مراعاة الآتي عند كتابة جملة الترتيب .

- يجب أن تُكتب في آخر جملة SELECT .
- تحتوي على أسماء حقول Columns أو أسماء مستعارة Alies .
- للترتيب تصاعدياً أكتب (ASC) وهي اختصار لكلمة (Ascending) وهي القيمة الافتراضية للترتيب (Default) .
- للترتيب تنازلياً لابد من كتابة (DESC) وهي اختصار لكلمة Descending .

مثال (١) : عرض أسماء و وظائف وأرقام إدارات الموظفين الذين يعملون بوظيفة (**CLERK**) ، مع ترتيب الناتج تصاعدياً حسب رقم الإدارة .

```
SQL> SELECT ename , job , deptno
  2  FROM    emp
  3  WHERE   job = 'CLERK'
  4  ORDER BY  deptno
```

لاحظ
كتابة النص بين علامتي ' ' وبالأحرف الكبيرة .

ENAME	JOB	DEPTNO
MILLER	CLERK	10
SMITH	CLERK	20
ADAMS	CLERK	20
JAMES	CLERK	30

المثال رقم (١) يبين طريقة استخدام جملة الشرط Where ، وذلك لعرض بيانات الموظفين الذين يعملون بوظيفة CLERK فقط ، لاحظ أن جملة الشرط قد احتوت على طرفين بينهما معامل حسابي وهو (=) ، وكانت النتيجة كما هي واضحة في المثال وذلك لأن جملة الشرط قد تحققت وكان هناك موظفون يعملون بوظيفة CLERK ، لاحظ أيضاً أن الطرف الثاني من جملة الشرط (**'CLERK'**) قد استخدمنا فيه ثابتاً نصياً و وضعنا هذا الثابت بين علامتي التنصيص الفردية . وأيضاً كتبنا الثابت النصي بالأحرف الكبيرة وذلك لأن البيانات داخل جدول الموظفين مسجلة بالأحرف الكبيرة ولذلك لابد من مراعاة حالة أحرف البيانات داخل الجدول كبيرة أم صغيرة حسب ما هو موجود في الجدول . كما تم ترتيب الناتج تصاعدياً بواسطة ORDER BY .

معاملات المقارنة المستخدمة في جملة الشرط Where . **Comparison Operators**

تستخدم معاملات المقارنة التالية للمقارنة بين طرفي الشرط في جملة Where .

المعامـــل	المعنـــى
=	يساوي
>	أكبر من
>=	أكبر من أو يساوي
<	أقل من
<=	أقل من أو يساوي
<> أو =!	لايساوي

الصيغة العامة لجملة الشرط **WHERE** .

```
SQL > WHERE   تعبير   OPERATOR   قيمة
```

أمثلة مختلفة :

- **WHERE** hiredate = '01-JAN-95' بشرط أن تاريخ التعيين يساوي (١ يناير ٩٥)
- **WHERE** sal >= 1500 بشرط أن الراتب أكبر من أو يساوي ١٥٠٠ دولار .
- **WHERE** ename = 'SMITH' بشرط أن أسم الموظف يكون SMITH

مثال (٢) : عرض أسماء و وظائف ورواتب الموظفين الذين رواتبهم أكبر من أو تساوي **3000** .

```
SQL> SELECT   ename , job , sal
  2  FROM     emp
  3  WHERE    sal >= 3000 ;
```

ENAME	JOB	SAL
SCOTT	ANALYST	3000
KING	PRESIDENT	5000
FORD	ANALYST	3000

الناتج : أكبر من أو يساوي (٣٠٠٠)

مثال (٣) : عرض أسماء و رواتب وعمولة الموظفين الذين رواتبهم أقل من أو تساوي العمولة الخاصة بهم .

```
SQL> SELECT  ename , sal , comm
  2  FROM      emp
                3  WHERE     sal<= comm ;

ENAME          SAL      COMM
----------  ----------  ----------
MARTIN         1250        1400
```

في المثال السابق تم استخدام أسم العمود (comm) في طرف جملة الشرط الأيمن ، أي إننا من الممكن أن نستخدم أسماء الأعمدة للمقارنة مع أسماء أعمدة أخرى . وكانت النتيجة هي عرض بيانات الموظفين الذين رواتبهم أقل من أو تساوي العمولة الخاصة بهم .

معاملات مقارنة أخرى تستخدم في جملة الشرط WHERE .

هناك معاملات مقارنة أخرى تُستخدم في جملة الشرط هذه المعاملات تسهل عملية حصر البيانات بشكل أكبر ، و هي كالتالي :

المعامـــل	المعنـــي
قيمة AND قيمة BETWEEN	حصر البيانات بين رقمين
(مجموعة من القيم) IN	حصر البيانات ضمن مجموعة من القيم
LIKE { % , _ }	حصر البيانات حسب مطابقة النص أو الحروف
IS NULL	حصر البيانات الخالية Null

مثال (٤) : عرض أسماء الموظفين ورواتبهم الذين تنحصر رواتبهم بين 1500 و 2500 .

```
SQL> SELECT  ename , sal
  2  FROM      emp
  3  WHERE   sal   BETWEEN 1500  AND  2500 ;
```

القيمة القيمة

ENAME	SAL
ALLEN	1600
CLARK	2450
TURNER	1500

لاحظ :
الراتب محصور بين ١٥٠٠ و ٢٥٠٠

في المثال السابق تم عرض بيانات الموظفين الذين تنحصر رواتبهم بين ١٥٠٠ و ٢٥٠٠ ، لاحظ أن القيمة ١٥٠٠ قد ظهرت في النتيجة لأن ناتج جملة BETWEEN يشتمل على القيمة الصغرى والقيمة الكبرى ، ونؤكد هنا على أن القيمة الصغرى لابد أن تكون أقل من القيمة الكبرى . فإذا عكسنا القيمة الكبرى مكان الصغرى (BETWEEN 2500 AND 1500) فإن جملة الشرط لن تتحقق أي تكون قيمتها (FALSE) وتظهر رسالة (No Row Selected) ومعناها لم يتم تحديد أي صف .

مثال (٥) : عرض رقم وأسم وراتب ورقم المدير للموظفين الذين لديهم مديرين بالأرقام التالية (٧٩٠٢ و ٧٥٦٦ و ٧٧٨٨) .

```
SQL> SELECT  empno , ename , sal , mgr
  2 FROM       emp
  3 WHERE      mgr  IN  (7902,7566,7788,7839)  ;
```

EMPNO	ENAME	SAL	MGR
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788
7902	FORD	3000	7566

في المثال السابق تم استرجاع بيانات الموظفين الذين لديهم مديرين من بين الأرقام التالية : (٧٩٠٢ و ٧٥٦٦ و ٧٧٨٨) أما المدير ذو الرقم (٧٨٣٩) فليس لديه موظفين لذلك لا يوجد له نتيجة . وبهذا فإن جملة IN تقوم بالبحث عن الرقم من بين قائمة الأرقام الموجودة داخل الأقواس .

معامل { _ , % } LIKE .

يستخدم هذا المعامل للبحث عن نص معين داخل ثابت أو حقل نصي ، حيث يتم مطابقة حروف النص المذكورة في جملة الشرط .

- **(%)** هذا الرمز يعني أي حرف أو أحرف ، مثلاً التعبير ('A%') يعني مطابقة النصوص التي تبدأ بحرف A مهما كانت باقي الحروف التالية له . فهو يستخدم للبحث عن نص يبدأ بالحرف A .

والتعبير ('%A') يعني مطابقة النصوص التي تنتهي بحرف A مهما كانت الحروف التي تسبقه .ويستخدم للبحث عن النصوص التي تنتهي بالحرف A .

أما التعبير ('%A%') فهو يستخدم للبحث عن النصوص التي تحتوي على الحرف A .

❖ (_) هذا الرمز يعني مطابقة حرف واحد فقط ، فمثلاً التعبير ('_A%') يعني أنه بغض النظر عن الحرف الأول، ويستخدم هذا التعبير عندما نريد البحث عن نص يكون الحرف الثاني فيه هو A .

أما التعبير ('_ _A') ، فيستخدم للبحث عن نص يكون الحرف الثالث فيه هو A .

ملحوظة :

يجب مراعاة حالة الأحرف هل هي كبيرة أم صغيرة عند استخدام المعامل LIKE .

مثال (٦) : عرض أسماء الموظفين الذين تبدأ أسمائهم بالحرف S .

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename  LIKE  'S%'   ;

ENAME
----------
SMITH
SCOTT
```

مثال (٧) : عرض أسم وتاريخ تعيين الموظفين الذين تم تعيينهم في العام ١٩٨١ م .

```
SQL> SELECT   ename , hiredate
 2  FROM      emp
 3  WHERE     hiredate LIKE '%81' ;

ENAME      HIREDATE
---------- ----------
ALLEN      20/02/81
WARD       22/02/81
JONES      02/04/81
MARTIN     28/09/81
BLAKE      01/05/81
CLARK      09/06/81
KING       17/11/81
TURNER     08/09/81
JAMES      03/12/81
FORD       03/12/81
```

مثال (٨) : عرض أسماء الموظفين الذين يكون الحرف الثاني في أسمائهم هو A .

```
SQL> SELECT   ename
  2  FROM    emp
  3  WHERE   ename  LIKE  '_A%'  ;

ENAME
----------                    الحرف الثاني ( A )
WARD
MARTIN
JAMES
```

في المثال السابق تم البحث عن الاسماء التي يكون الحرف الثاني فيها هو (A) ثم عرض هذه الاسماء .

مثال (٩) : عرض أسم ورقم المدير للموظفين الذين لا يوجد لديهم مدير .

```
SQL> SELECT  ename , mgr
  2  FROM    emp
  3  WHERE   mgr  IS NULL  ;

ENAME          MGR
----------     ----------
KING                    <-------  NULL
```

في المثال رقم (٩) تم استخدام المعامل IS NULL والذي يقوم بحصر البيانات الخالية ، ففي المثال تم عرض بيانات الموظفين الذين لا يوجد لديهم مدير أي إن حقل المدير(MGR) لهذا الموظف خالي ليس به بيانات .

ملحوظة :

لا يمكن استخدام المعامل (=) مع القيم الخالية NULL ولكن لابد من استخدام المعامل IS NULL ، يمكن أن تجرب الأمر في المثال السابق بالشكل التالي لتعرف الفرق ؟

```
SQL> SELECT  ename , mgr          <-------  الأمر هنا خطأ لاستخدام المعامل ( = )
  2  FROM    emp
  3  WHERE   mgr = NULL ;
```

المعاملات المنطقية في جملة الشرط WHERE .

المعامـــل	المعنـــــي
AND	ترجع النتيجة TRUE إذا كانت جملتا الشرط TRUE
OR	ترجع النتيجة TRUE إذا كانت إحدى جملتي الشرط TRUE
NOT	تنفي النتيجة ، أي ترجع النتيجة TRUE إذا كانت جملة الشرط FALSE

تستخدم المعاملات المنطقية التالية لتكوين أكثر من شرط في جملة WHERE ، وهي معاملات تربط بين جملتين شرطيتين أو أكثر وتكون النتيجة أما (TRUE) أي تحقق الشرط أو (FALSE) لم يتحقق الشرط .

المعامل AND :

هذا المعامل يربط بين جملتين شرطيتين ويكون الناتج TRUE إذا كانت كلتا الجملتين TRUE .
والجدول التالي يوضح ناتج المعامل AND مع الحالات المختلفة لجملتي الشرط .

جملة الشرط الأولى	جملة الشرط الثانية	ناتج المعامل **AND**
True	True	True
True	False	False
False	False	False
True	Null	Null
False	Null	False
Null	Null	Null

بالتدقيق في الجدول السابق يمكن أن نستخلص النتائج التالية :

أولاً : ناتج المعامل AND يكون دائماً FALSE إلا في حالة أن الجملتين TRUE فقط .

ثانياً : عند استخدام القيمة NULL مع المعامل AND يكون الناتج دائماً NULL إلا في حالة أن إحدى الجملتين تكون NULL والأخرى FALSE فقط .

مثال (١٠) : عرض رقم وأسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من أو تساوي 1100 وفي نفس الوقت وظيفتهم CLERK .

```
SQL> SELECT   empno , ename , job , sal
  2    FROM    emp
  3    WHERE   sal >=1100  AND  job='CLERK' ;
                   T                T
                  جملة             جملة
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

المثال السابق يبين أن جملتي الشرط sal >=1100 و job='CLERK' قد تحققت أي إن نتيجة كلاً منهما كانت (TRUE) ولذلك فإن ناتج المعامل AND هو (TRUE) ولهذا قد تم عرض البيانات كما هو واضح من المثال .

مثال (١١) : عرض أسم وراتب وعمولة الموظفين الذين يزيد راتبهم عن 1100 وفي نفس الوقت تقل عمولتهم عن 500 .

```
SQL> SELECT ename , sal , comm
  2    FROM   emp
  3   WHERE  sal>1100  AND comm<500 ;
```

ENAME	SAL	COMM
ALLEN	1600	300
TURNER	1500	0

المعامل OR :

هـذا المعامـل يـربط بـين جملـتين شـرطيتين ويكـون النـاتج TRUE إذا كانـت إحـدى الجملـتين أو كلاهما TRUE . والجدول التالي يوضح ناتج المعامل OR مع الحالات المختلفة لجملتي الشرط .

جملة الشرط الأولى	جملة الشرط الثانية	ناتج المعامل OR
True	True	True
True	False	True
False	False	False
True	Null	True
False	Null	Null
Null	Null	Null

بالتدقيق في الجدول السابق يمكن أن نستخلص النتائج التالية :

أولاً : ناتج المعامل OR يكون دائماً TRUE إلا في حالة أن الجملتين FALSE فقط .

ثانياً : عند استخدام القيمة NULL مع المعامل OR يكون الناتج دائماً NULL إلا في حالة أن إحدى الجملتين تكون NULL والأخرى TRUE فقط فيكون الناتج TRUE .

مثال (١٢) : عرض رقم وأسم ووظيفة وراتب الموظفين الذين رواتبهم أكبر من (2500) أو تكون وظيفتهم MANAGER .

```
SQL> SELECT   empno , ename , job , sal
  2  FROM     emp
  3  WHERE    sal >2500  OR   job='MANAGER' ;
```

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

المثال السابق يبين أنه لابد من تحقق أي من الجملتين حتى يتم استرجاع بيانات ، وبالتدقيق في الموظف رقم (7782) نجد أنه بالرغم من أن راتبه يقل عن ٢٥٠٠ إلا أنه ظهر في النتيجة وذلك لتحقق شرط الوظيفة (MANAGER) ، أي إنه يجب أن يتحقق أحد الشرطين لاسترجاع البيانات .

مثال (١٣) : عرض أسم وراتب ورقم الإدارة للموظفين الذين رواتبهم أقل من (1000) أو تكون إداراتهم رقم (10) .

```
SQL> SELECT   ename , sal , deptno
  2  FROM     emp
  3  WHERE    sal<1000  OR  deptno=10 ;
```

ENAME	SAL	DEPTNO
SMITH	800	20
CLARK	2450	10
KING	5000	10
JAMES	950	30
MILLER	1300	10

في المثال السابق تم عرض بيانات الموظفين الذين تقل رواتبهم عن ١٠٠٠ أو الموظفين المسجلين في الإدارة رقم (١٠) . نلاحظ من المثال أن الموظفين المسجلين في الإدارة رقم ٢٠ يأخذون راتباً أقل من ١٠٠٠ . والموظفين الذين يأخذون راتباً أكبر من ١٠٠٠ هم مسجلين في الإدارة رقم (١٠) وهذا يؤكد أنه لاسترجاع بيانات لابد من تحقق إحدى جملتي الشرط على الأقل .

المعامل NOT :

هذا المعامل يقوم بعكس ناتج جملة الشرط ، أي إنه إذا كانت جملة الشرط (TURE) فإن ناتج المعامل NOT يكون (FALSE) والعكس ، والجدول التالي يبين تأثير هذا المعامل على جملة الشرط .

جملة الشرط	ناتج المعامل NOT
True	False
False	True
Null	Null

يستخدم المعامل NOT أيضاً لنفي المعاملات الموضحة بالجدول التالي :

المعامـــل	نفي المعامل	المعني
BETWEEN ... AND ...	NOT BETWEEN .. AND ..	ليست بين رقمي .... و .....
IN (....)	NOT IN (....)	ليست ضمن القائمة .....
LIKE { % , _ }	NOT LIKE { % , _ }	ليست مطابقة
IS NULL	IS NOT NULL	ليست قيمة خالية

أمثلة على استخدام المعامل NOT لعكس المعاملات السابقة الذكر .

- WHERE job NOT IN ('CLERK' , 'MANAGER')
- WHERE sal NOT BETWEEN 1000 AND 1500
- WHREE ename NOT LIKE '%A%'
- WHERE comm IS NOT NULL

مثـال (١٤) : عـرض أسـم ووظيفـة المـوظفين الـذين ليسـت وظـائفهم مـن ضـمن الوظـائف التاليـة : (CLERK , MANAGER , ANALYST) .

```
SQL> SELECT   ename , job
  2  FROM     emp
  3  WHERE    job  NOT  IN  ( 'CLERK' , 'MANAGER' , 'ANALYST' ) ;
```

ENAME	JOB
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
KING	PRESIDENT
TURNER	SALESMAN

في هذا المثال تم نفي المعامل IN(...) ولذلك تم عرض بيانات الموظفين الذين لهم وظائف ليست من ضمن قائمة الوظائف التالية (CLERK , MANAGER , ANALYST) .

مثال (١٥) : عرض أسم ووظيفة وراتب الموظفين الذين لا تنحصر رواتبهم بين **1000** و **3000** .

```
SQL> SELECT   ename , job , sal
  2  FROM     emp
  3  WHERE    sal NOT BETWEEN 1000 AND 3000 ;
```

ENAME	JOB	SAL
SMITH	CLERK	800
KING	PRESIDENT	5000
JAMES	CLERK	950

في المثال السابق تم استبعاد الموظفين الذين تنحصر رواتبهم بين ١٠٠٠ و ٣٠٠٠ ، وتم عرض باقي الموظفين . وبذلك قد تم نفي المعامل (.... AND ... BETWEEN) .

مثال (١٦) : عرض أسم ووظيفة وراتب وعمولة الموظفين الذين يأخذون عمولة .

```
SQL> SELECT  ename , job , sal , comm
  2  FROM    emp
  3  WHERE  comm IS NOT NULL ;
```

ENAME	JOB	SAL	COMM
ALLEN	SALESMAN	1600	300
WARD	SALESMAN	1250	500
MARTIN	SALESMAN	1250	1400
TURNER	SALESMAN	1500	0

في المثال السابق تم عرض بيانات الموظفين الذين يأخذون عمولة ، أي الذين ليست عمولتهم خالية NULL .

وإذا أردنا ترتيب الناتج تنازلياً حسب الراتب نضيف السطر التالي على المثال السابق .

ORDER BY sal DESC

١ – اكتب جملة استعلام لعرض أسماء ورواتب الموظفين الذين رواتبهم أكبر من 2850 والنتيجة مرتبة تنازلياً حسب الراتب ؟ بحيث تظهر النتيجة كالتالي :

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000
JONES	2975

٢ – اكتب جملة استعلام لعرض أسماء ورواتب الموظفين الذين رواتبهم لا تنحصر بين (1500,2850) ؟ بحيث تظهر النتيجة كالتالي :

ENAME	SAL
SMITH	800
WARD	1250
JONES	2975
MARTIN	1250
SCOTT	3000
KING	5000
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

٣ – اكتب جملة استعلام لعرض أسماء وراوتب الموظفين الذين رواتبهم أكبر من 1500 ومسجلين في الإدارة رقم 10 أو مسجلين في الإدارة رقم 30 والنتيجة مرتبة تنازلياً حسب الراتب ؟ بحيث تظهر النتيجة كالتالي :

ENAME	SAL
KING	5000
BLAKE	2850
CLARK	2450
ALLEN	1600

٤ – اكتب جملة استعلام لعرض أسماء وتواريخ تعيين الموظفين المعينين سنة 1982 ؟ بحيث تظهر النتيجة كالتالي :

ENAME	HIREDATE
SCOTT	09-DEC-82
MILLER	23-JAN-82

٥ – اكتب جملة استعلام لعرض أسماء ورواتب وعمولة الموظفين الذين يأخذون عمولة ؟ بحيث تظهر النتيجة كالتالي :

ENAME	SAL	COMM
ALLEN	1600	300
TURNER	1500	0
MARTIN	1250	1400
WARD	1250	500

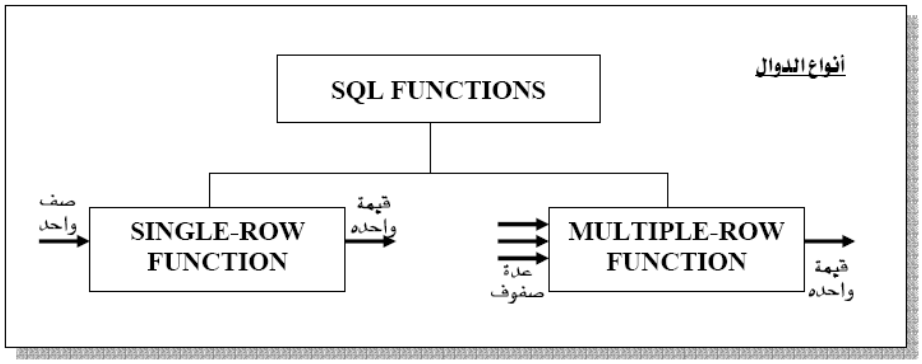
٦ – اكتب جملة استعلام لعرض أسماء الموظفين الذين يكون في أسمائهم الحرف الثالث A ؟ بحيث تظهر النتيجة كالتالي :

ENAME
BLAKE
CLARK
ADAMS

٧ – اكتب جملة استعلام لعرض أسماء الموظفين الذين تتضمن أسماؤهم الحرفين LL ؟ بحيث تظهر النتيجة كالتالي :

ENAME
ALLEN
MILLER

توجد دوال تستخدم داخل لغة الاستفسارات تسمى SQL FUNCTIONS وهذه الدوال تعتبر أداة قوية ومفيدة عند استخدام جملة SELECT ، وتنقسم هذه الدوال إلى نوعين كما هو مبين في الشكل التالي :



النوع الأول : يسمى دوال الصف الواحد SINGLE_ROW FUNCTIONS وهذا النوع يقوم بالتعامل مع بيانات صف واحد فقط لآخراج قيمة واحدة وهو الذي سيتم شرحه بالتفصيل في هذا الفصل .

النوع الثاني : يسمى الدوال التجميعية لأكثر من صف MULTIPLE_ROW FUNCTION وهذا النوع يقوم بالتعامل مع بيانات أكثر من صف لآخراج قيمة واحدة فقط وسوف يتم شرحه في الفصل القادم بإذن الله .

دوال الصف الواحد : Single-Row Functions

وهي كما ذكرنا عبارة عن دوال تتعامل مع بيانات صف واحد فقط وتكون نتيجتها قيمة واحدة فقط ، وتستخدم في أي مكان من جملة SELECT ما عدا الجزء FROM ، وتنقسم إلى عدة أنواع هي :

- دوال حرفية Character Functions .
- دوال رقمية Number Functions .
- دوال تاريخ Date Functions .
- دوال تحويل Conversion Functions .

الدوال الحرفية : Character Functions

وهي دوال تتعامل مع البيانات الحرفية وتكون نتيجتها إما حروفاً أو أرقاماً . والجدول التالي يبين جميع الدوال الحرفية ووظيفة كل منها .

الدالـــــة FUNCTION	وظيفتهـــــا
LOWER(column\expression)	دالة تستخدم لتحويل جميع الحروف (عمود أو سلسلة) إلى حروف صغيرة Small
UPPER(column\expression)	دالة تستخدم لتحويل حروف (عمود أو سلسلة) إلى حروف كبيرة Capital
INITCAP(column\expression)	دالة تستخدم لتحويل الحرف الأول فقط من (عمود أو سلسلة) إلى حرف كبير Capital وباقي الحروف تحول إلى حروف صغيرة
CONCAT(column1\expression1, Column2\expression2)	دالة ربط عمودين أو سلسلتين معاً وهي تماماً مثل أداة الربط (||)
SUBSTR(column\expression,m,n)	دالة تستخدم لقطع جزء من عمود أو سلسلة بدايةً من الحرف رقم m وعدد الحروف المقطوعة هي n .
LENGTH(column\expression)	دالة تستخدم لإيجاد عدد حروف السلسلة أو العمود (الناتج عدد)
INSTR(column\expression,m)	دالة تستخدم لتحديد مكان حرف معين داخل سلسلة أو عمود (الناتج عدد) والحرف m يعبر عن الحرف المراد تحديد مكانه
LPAD(column\expression,n,'string')	دالة تستخدم لضبط بيانات عمود أو سلسلة ناحية اليمين وذلك بملء حرف معين من اليسار والحرف n لتحديد الطول بعد الضبط
RPAD(column\expression,n,'string')	دالة تستخدم لضبط بيانات عمود أو سلسلة ناحية اليسار وذلك بملء حرف معين من اليمين والحرف n لتحديد الطول بعد الضبط
TRIM('character' FROM column\expression)	دالة تستخدم لقطع حرف معين من بداية أو نهاية الكلمة فقط

والجدول التالي يحتوي على أمثلة لكل دالة من الدوال الحرفية مع النتيجة لكل منها .

المثـــــــــــال	النتيجـــــة
Select LOWER('GOOD by') from dual ;	good by
Select UPPER('GOOD by') from dual ;	GOOD BY
Select INITCAP('GOOD') from dual ;	Good
Select CONCAT('GOOD' , 'BY') from dual;	GOODBY
Select SUBSTR('GOOD BY',2,3) from dual;	OOD
Select LENGTH('GOOD') from dual;	4
Select INSTR('GOOD','D') from dual;	4
Select LPAD('AHMED',10,'*') from dual ;	*****AHMED
Select RPAD('AHMED',10,'*') from dual ;	AHMED*****
Select TRIM('S' FROM 'SAMI') from dual;	AMI

ملحوظة : الجدول (DUAL) هو جدول وهمي موجود داخل لغة أوراكل SQL يستخدم لإجراء العمليات التي لا يدخل فيها أي جدول من داخل قاعدة البيانات .

بعض الأمثلة على الدوال الحرفية .

مثال (١) :

```
SQL> SELECT LOWER(ename) , UPPER(job) , INITCAP(job) , CONCAT(ename, job)
  2  FROM emp
  3  WHERE sal=3000 ;
```

LOWER(ENAME)	UPPER(JOB)	INITCAP(JOB)	CONCAT(ENAME, JOB)
scott	ANALYST	Analyst	SCOTTANALYST
ford	ANALYST	Analyst	FORDANALYST

في المثال السابق تم عرض أسماء الموظفين بالحروف الصغيرة ووظائفهم بالحروف الكبيرة وأيضاً وظائفهم بحيث يكون الحرف الأول كبيراً والباقي صغيراً وربطنا بين أسماء الموظفين ووظائفهم لعرضهم كصف واحد .

مثال (٢) :

```
SQL> SELECT ename,SUBSTR(ename,2,3),LENGTH(ename),INSTR(ename,'K')
  2  FROM emp
  3  WHERE LOWER( job)='manager' ;
```

ENAME	SUBSTR(ENAME,2,3)	LENGTH(ENAME)	INSTR (ENAME, 'K')
JONES	**ONE**	5	0
BLAKE	**LAK**	5	4
CLARK	**LAR**	5	5

شرح مثال (٢) :

سوف نقوم بشرح كل جزء من جملة SELECT السابقة على حده حتى يسهل فهم عمل كل دالة .

- في الجزء الأول (ENAME) تم عرض أسماء الموظفين وذلك لمعرفة تأثير الدوال عليه .
- في الجزء الثاني SUBSTR(ENAME,2,3) تم عرض جزء من الاسماء بدايةً من الحرف رقم (٢) من اليسار وعدد الحروف المقطوعة (٣) أحرف بفعل الدالة SUBSTR والتي تقوم بقطع جزء معين من الكلمة .
- في الجزء الثالث من المثال تم عرض عدد أحرف الاسماء بفعل الدالة LENGTH ونلاحظ هنا أن ناتج هذه الدالة يكون رقمياً .
- في الجزء الرابع من المثال تم عرض ترتيب الحرف (K) بدايةً من اليسار داخل أسماء الموظفين بفعل الدالة INSTR فنلاحظ أن ترتيب الحرف K داخل اسم الموظف BLAKE هو الرابع بينما ترتيبه داخل أسم الموظف CLARK هو الخامس .
- لاحظ إننا قد استخدمنا الدالة LOWER في جملة الشرط WHERE وذلك لتفادي الخطأ الذي ينتج من عدم معرفة حالة الحروف المكتوبة في الجدول .

مثال (٣) :

```
SQL>SELECT ename,TRIM('S' FROM ename), LPAD(ename,10,'*') , RPAD(ename,10,'#')
  2  FROM    emp
  3  WHERE   sal>2500 ;
```

ENAME	TRIM('S' FROM ENAME)	LPAD(ENAME,10,'*')	LPAD(ENAME,10,'#')
JONES	JONE	*****JONES	JONES#####
BLAKE	BLAKE	*****BLAKE	BLAKE#####
SCOTT	COTT	*****SCOTT	SCOTT#####
KING	KING	******KING	KING######
FORD	FORD	******FORD	FORD######

شرح مثال (٣) :

- في الجزء الأول (ENAME) تم عرض أسماء الموظفين وذلك لمعرفة تأثير الدوال عليه .
- في الجزء الثاني من المثال (TRIM('S' FROM ENAME تم عرض الاسماء بعد قص الحرف S من بدايتها أو نهايتها وذلك بفعل الدالة TRIM فنلاحظ أن الموظف JONES قد تم قص الحرف S من نهايته وأيضاً الموظف SCOTT تم قص الحرف S من بدايته .
- في الجزء الثالث (LPAD(ENAME,10,'*') تم عرض أسماء الموظفين بعد إضافة عدد من الرمز (*) إلى الاسماء من اليسار بحيث يصبح إجمالي طول الاسماء (١٠) أحرف ، فمثلاً أسم الموظف JONES عبارة عن خمسة أحرف وبفعل الدالة LPAD قد تم إضافة الرمز (*) خمسة مرات من اليسار ليصبح طوله عشرة أحرف ، ونلاحظ أن هذه الدالة تقوم بمحاذاة الاسماء ناحية اليمين (right-justified) .
- الجزء الرابع (RPAD(ENAME,10,'#') يماثل تماماً الجزء الثالث ولكن إضافة الرمز (#) إلى الاسماء تمت من اليمين ، ونلاحظ أيضاً أن هذه الدالة تقوم بالمحاذاة ناحية اليسار (left-justified) .

تذكر دائماً أنه يمكن استخدام هذه الدوال في أي جزء من جملة SELECT ماعدا الجزء FROM .

الدوال رقمية : Number Functions

وهي دوال تعمل مع البيانات الرقمية وتكون نتيجتها أرقاماً فقط . والجدول التالي يبين جميع الدوال الرقمية ووظيفة كل منها .

الدالـــــة FUNCTION	وظيفتهـــــا
ROUND(column\expression,n)	دالة تستخدم لقص عدد معين من الجزء العشري مع تقريب الأعداد إلى أقرب عدد العشري أو إلى عدد صحيح والحرف n يبين عدد الأرقام بعد العلامة العشرية ، وتوجد حالات للحرف n . • إذا كان (n=0) فإن التقريب يكون إلى أقرب عدد صحيح . • إذا كان (n>0) أي عدد موجب فإن التقريب يكون في الجزء بعد العلامة العشرية (الجزء العشري) . • إذا كان (n<0) أي عدد سالب فإن التقريب يكون في الجزء قبل العلامة العشرية (الجزء الصحيح) .
TRUNC(column\expression,n)	دالة تستخدم لقص عدد معين من الجزء العشري بدون تقريب ، وأيضاً توجد حالات للحرف n . • إذا كان (n=0) فإنه يتم قص الجزء العشري كله ويكون الناتج عدد صحيح . • إذا كان (n>0) أي عدد موجب فإن القص يكون في الجزء بعد العلامة العشرية (الجزء العشري) . • إذا كان (n<0) أي عدد سالب فإن القص يكون في الجزء قبل العلامة العشرية (الجزء الصحيح) .
MOD(m,n)	دالة تستخدم لإيجاد باقي قسمة العدد m على العدد n .

مثال (٤) :

```
SQL>SELECT  ROUND(45.923,0),ROUND(45.923,2),ROUND(45.923,-1),ROUND(45.923,-2)
  2  FROM dual ;
```

الأعداد

ROUND(45.923,0)	ROUND(45.923,2)	ROUND(45.923,-1)	ROUND(45.923,-2)
46	45.92	50	0

المثال السابق يوضح تأثير استخدام الدالة الرقمية ROUND على الأرقام ، فنجد أنه عندما كانت قيمة n مساوية للصفر فإنه تم التقريب إلى عدد صحيح فكانت النتيجة مساوية (٤٦)، وعندما كانت n تساوي العدد (٢) فإنه تم التقريب إلى عددين عشريين فكانت النتيجة (٤٥.٩٢) ، وعندما كانت n تساوي (-١) فإن التقريب تم على الجزء قبل العلامة العشرية في العدد (٥) فأصبحت النتيجة (٥٠) ، ونلاحظ أنه عندما كانت n تساوي (-٢) فإنه تم تقريب العدد (٤٥) إلى الصفر وذلك لأن العدد (٤٥) أقل من (٥٠) لأننا نقرب إلى مئات .

مثال (٥) :

```
SQL> SELECT TRUNC(45.923,0),TRUNC(45.923,2),TRUNC(45.923,-1),TRUNC(45.923,-2)
  2  FROM  dual ;
```

TRUNC(45.923,0)	TRUNC(45.923,2)	TRUNC(45.923,-1)	TRUNC(45.923,-2)
45	45.92	40	0

المثال السابق يوضح تأثير استخدام الدالة الرقمية TRUNC على الأرقام ، فنجد أنه عندما كانت n مساوية للصفر تم قص الجزء العشري ، وعندما كانت n تساوي (٢) فإنه تم قص العدد العشري ليصبح رقمين فقط ، عندما كانت n تساوي (-١) تم قص العدد (٥) من الجزء قبل العلامة العشرية فكانت النتيجة (٤٠) ، وعندما كانت n تساوي (-٢) فإنه تم قص العدد الصحيح فأصبح صفراً .

مثال (٦) :

```
SQL> SELECT ename , sal , comm , MOD(sal,comm)
  2  FROM emp
  3  WHERE sal=1600 ;
```

ENAME	SAL	COMM	MOD(SAL,COMM)
ALLEN	1600	300	100

في المثال السابق تم عرض أسم الموظف وراتبه وعمولته وباقي قسمة راتبه على ما يأخذه من عمولة .

دوال التاريخ : Date Functions

هي دوال تتعامل مع البيانات التي من النوع (تاريخ)، ولما للتاريخ والوقت من أهمية بالغة في الحياة اليومية بل إنه يمثل الحياة فقد قامت أوراكل بتوفير دوال بسيطة وقوية للتعامل مع التاريخ ، علماً بأن أوراكل قامت بتخزين التاريخ بالشكل التالي (DD-MON-YY) كما يمكن تغيير هذا الشكل عن طريق دوال أخرى تسمى دوال التحويل والتي سوف ندرسها بالتفصيل لاحقاً ، والجدول التالي يبين جميع دوال التاريخ ووظيفة كل منها .

الدالـــــة FUNCTION	وظيفتهـــــا
SYSDATE	دالة تستخدم لعرض تاريخ النظام الموجود بجهاز الحاسب الآلي (تاريخ اليوم الحالي)
MONTHS_BETWEEN(date1,date2)	دالة لإيجاد عدد الأشهر بين تاريخين
ADD_MONTHS(date,n)	دالة لإضافة عدد معين من الأشهر على تاريخ معطى
NEXT_DAY(date,'day')	دالة لإيجاد تاريخ يوم معين بعد تاريخ معطى
LAST_DAY(date)	دالة لإيجاد آخر يوم في الشهر لتاريخ معطى
ROUND(date)	دالة تستخدم لتقريب التاريخ لأقرب شهر أو سنة
TRUNC(date)	دالة تستخدم لقص التاريخ لأقرب شهر أو سنة

الجدول التالي يحتوي على أمثلة لكل دالة من دوال التاريخ ونتيجة ومعنى كل منها .

المعني	النتيجـــــة	المثـــــــــــال
تم إيجاد عدد الأشهر بين التاريخين المعطيين	19.6774194	**MONTHS_BETWEEN('01-SEP-95' , '11-JAN-94)**
تم إضافة ٦ أشهر على التاريخ المعطى	'11-JUL-94'	**ADD_MONTHS('11-JAN-94' , 6)**
إيجاد تاريخ يوم الجمعة بعد التاريخ '01-SEP-95'	'08-SEP-95'	**NEXT_DAY('01-SEP-95' , 'FRIDAY')**
إيجاد آخر يوم في شهر سبتمبر September	'30-SEP-95'	**LAST_DAY('01-SEP-95')**
تم تقريب التاريخ المعطى إلى أقرب شهر وبما أن اليوم هو ٢٥ أي أكبر من ١٥ فتم التقريب إلى أول الشهر الذي يليه	01-AUG-95	**ROUND('25-JUL-95' , 'MONTH')**
تم تقريب التاريخ المعطى إلى أقرب سنة وبما أن شهر (JULY) هو شهر (٧) فإنه تم التقريب إلى أول السنة التالية	01-JAN-96	**ROUND('25-JUL-95' , 'YEAR')**
تم قص التاريخ المعطى بدون تقريب إلى أول يوم في الشهر	'01-JUL-95'	**TRUNC('25-JUL-95' , 'MONTH')**
تم قص التاريخ المعطى بدون تقريب إلى أول السنة الحالية	'01-JAN-95'	**TRUNC('25-JUL-95' , 'YEAR')**

تذكر أن شهور السنة الميلادية هي :

(يناير – فبراير – مارس – أبريل – مايو – يونيو – يوليو – أغسطس – سبتمبر – أكتوبر – نوفمبر – ديسمبر)

(JANURY-FEBRUARY-MARCH-APRIL-MAY-JUNE-JULY-AUGUST-SEPTEMBER-OCTOBER-NOVEMBER-DECEMBER)

بعض الأمثلة على دوال التاريخ :

مثال (٧) :

```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
---------
25-01-2004
```

في المثال السابق تم عرض تاريخ اليوم وهذا التاريخ هو التاريخ المسجل داخل جهاز الحاسب الذي تعمل عليه الآن .

مثال (٨) :

```
SQL> SELECT empno, hiredate , MONTHS_BETWEEN(sysdate,hiredate)
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

EMPNO	HIREDATE	MONTHS_BETWEEN(SYSDATE,HIREDATE
7788	19-04-1987	201.200404
7876	23-05-1987	200.071371

في المثال السابق تم عرض رقم الموظفين وتاريخ تعيينهم وعدد الشهور بين تاريخ تعيينهم وتاريخ اليوم أي عدد الشهور التي قضوها في العمل ، نلاحظ أن عدد الأشهر عبارة عن أرقام عشرية ويمكن تقريبها باستخدام الدالة الرقمية ROUND كالآتي :

```
SQL> SELECT empno, hiredate , ROUND(MONTHS_BETWEEN(sysdate,hiredate))
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

بعض الأمثلة على دوال التاريخ :

مثال (٧) :

```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
---------
25-01-2004
```

في المثال السابق تم عرض تاريخ اليوم وهذا التاريخ هو التاريخ المسجل داخل جهاز الحاسب الذي تعمل عليه الآن .

مثال (٨) :

```
SQL> SELECT empno, hiredate , MONTHS_BETWEEN(sysdate,hiredate)
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

EMPNO	HIREDATE	MONTHS_BETWEEN(SYSDATE,HIREDATE
7788	19-04-1987	201.200404
7876	23-05-1987	200.071371

في المثال السابق تم عرض رقم الموظفين وتاريخ تعيينهم وعدد الشهور بين تاريخ تعيينهم وتاريخ اليوم أي عدد الشهور التي قضوها في العمل ، نلاحظ أن عدد الأشهر عبارة عن أرقام عشرية ويمكن تقريبها باستخدام الدالة الرقمية ROUND كالآتي :

```
SQL> SELECT empno, hiredate , ROUND(MONTHS_BETWEEN(sysdate,hiredate))
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

مثال (٩) :

```
SQL>SELECT empno,hiredate,ADD_MONTHS(HIREDATE,6),LAST_DAY(HIREDATE)
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

EMPNO	HIREDATE	ADD_MONTHS (HIREDATE,6)	LAST_DAY(HIREDATE)
7788	19-04-1987	19-10-1987	30-04-1987
7876	23-05-1987	23-11-1987	31-05-1987

في المثال السابق تم عرض رقم الموظفين وتاريخ تعيينهم ، وتاريخ التعيين بعد إضافة (٦) أشهر عليه ، وآخر يوم في الشهر لتاريخ تعيينهم .

مثال (١٠) :

```
SQL> SELECT empno,hiredate,NEXT_DAY(hiredate,'FRIDAY')
  2  FROM emp
  3  WHERE hiredate like '%1987' ;
```

EMPNO	HIREDATE	NEXT_DAY(hiredate,'FRIDAY')
7788	19-04-1987	24-04-1987
7876	23-05-1987	29-05-1987

في المثال السابق تم عرض رقم الموظفين وتاريخ تعيينهم ، وتاريخ أول يوم يوافق الجمعة بعد تاريخ تعيينهم .

دوال التحويل : Conversion Functions

يوجد أنواع كثيرة من البيانات(DATATYPE) يمكن تخزينها داخل الجداول ومن أهمها بيانات رقمية (NUMBER) وبيانات حرفية(CHARACTER) وبيانات التاريخ (DATE) . ودوال التحويل تقوم بتحويل البيانات من نوع إلى آخر ، وهي عبارة عن ثلاثة أنواع والجدول التالي يبين هذه الأنواع ووظيفة كل منها :

الدالـــــة FUNCTION	وظيفتهـــــا
TO_CHAR(DATE/NUMBER ,'fmt')	تستخدم هذه الدالة لتحويل البيانات الرقمية أو بيانات التاريخ إلى بيانات حرفية بشكل معين (FORMAT) حسب الطلب fmt .
TO_DATE(CHAR , 'fmt')	تستخدم لتحويل البيانات الحرفية إلى بيانات من نوع التاريخ بشكل معين (FORMAT) حسب الطلب fmt .
TO_NUMBER(CHAR , 'fmt')	تستخدم لتحويل البيانات الحرفية إلى بيانات رقمية بشكل معين (FORMAT) حسب الطلب fmt .

ولما لهذه الدوال من أهمية بالغة في عملية تحويل البيانات من نوع إلى آخر فسوف نقوم بدراسة هذه الدوال كل دالة على حده .

دالة التحويل TO_CHAR .

أولاً : استخدام هذه الدالة لتحويل بيانات التاريخ DATE إلى بيانات حرفية .

الشكل العام :

```
TO_CHAR(DATE , 'fmt')
```

تقوم هذه الدالة بتحويل بيانات التاريخ إلى بيانات حرفية بشكل معين يسمى format ويعبر عنه بالجزء fmt . والمثال التالي يوضح كيفية استخدامها .

مثال (١١) :

```
SQL> SELECT sysdate,TO_CHAR(sysdate,'DD/MM/YYYY')
  2  FROM dual ;
```

التاريخ المراد تحويله إلى بيانات حرفية — شكل التاريخ بعد تحويله FORMAT

SYSDATE	TO_CHAR(SYADTE,'DD/MM/YYYY')
26-01-2004	26/01/2004

في المثال السابق تم عرض تاريخ النظام لليوم الحالي كما تم عرض هذا التاريخ بعد تحويله إلى بيانات حرفية بشكل مختلف هو (DD/MM/YYYY) . فمثلاً إذا أردنا عرض تاريخ النظام بحيث يظهر الشهر والسنة فقط ، نكتب الشكل FORMAT التالي (MM/YYYY) .

ملحوظة هامة : يجب كتابة الشكل الجديد للتاريخ محصوراً بين علامتي (' ') . وإذا أردنا إضافة بعض الكلمات في الشكل FORMAT يجب كتابتها بين علامتي (" ") فمثلاً إذا أردنا أن يظهر التاريخ بالشكل (01/2004 OF 26) نقوم بكتابة الشكل FORMAT كالتالي : (DD "OF" MM/YYYY) .

الجدول التالي يوضح بعض الأشكال التي يمكن استخدامها عند تحويل التاريخ إلى بيانات حرفية بشكل معين .

YYYY	إظهار السنة كاملة بالأرقام (القرن + السنة) مثل 2004 .
YY	إظهار رقمين فقط من السنة 04 .
YEAR	إظهار السنة كاملة كتابة(TWO THOUSAND FOUR) وحالة أحرف الكتابة تتوقف على حالة أحرف كلمة YEAR .
MM	إظهار الشهر في شكل رقمين 01 .
MONTH	إظهار الشهر كتابة (**JANURY**) .
DY	إظهار الثلاث حروف الأولى من اليوم (JAN) .
DAY	إظهار اليوم كاملاً كتابتاً (FRIDAY) .
HH12:MI:SS AM	إظهار الوقت بنظام ١٢ ساعة وهل هو صباحاً أم مساء (04:30:50 PM)

مثال (١٢) :

```
SQL> SELECT empno,TO_CHAR(hiredate,'DAY "OF" MONTH YYYY HH12:MI:SS AM')
  2  FROM emp
  3  WHERE ename=upper('king') ;
```

EMPNO	TO_CHAR(HIREDATE,'DAY "OF" MONTH YYYY HH12:MI:SS AM')
7839	TUESDAY OF NOVEMBER 1981 12:00:00 AM

في المثال السابق تم عرض رقم الموظف KING وتاريخ تعيينه بشكل خاص ، لاحظ وتأمل هذا الشكل .

دالة التحويل **TO_CHAR** .

ثانياً : استخدم هذه الدالة لتحويل البيانات الرقمية NUMBER إلى بيانات حرفية .

الشكل العام :

TO_CHAR(NUMBER , 'fmt')

تقوم هذه الدالة بتحويل البيانات الرقمية إلى بيانات حرفية بشكل معين يسمى format ويعبر بالجزء fmt .

وفي المثال التالي نقوم بعرض أرقام الموظفين ورواتبهم الذين يأخذون رواتب أكبر من ٢٥٠٠ دولار بعد تحويل بيانات رواتبهم إلى بيانات حرفية لتصبح بالشكل التالي مثلاً ($3,000) بدلاً من (3000) .

مثال (١٣) :

```
SQL> SELECT empno,TO_CHAR(sal , '$99,999') salary
  2  FROM emp
  3  WHERE sal > 2500 ;
```

EMPNO	SALARY
7566	$2,975
7698	$2,850
7788	$3,000
7839	$5,000
7902	$3,000

الجدول التالي يوضح بعض الرموز التي يمكن استخدامها عند تحويل البيانات الرقمية إلى بيانات حرفية لتظهر بشكل معين .

عدد تكرار هذا الرقم يمثل عدد الخانات التي تظهر ، مثلاً عندما نكتب (99) معناها ظهور رقمين وهكذا .	9
يعني ظهور الرقم وقبله صفر .	099
يعني ظهور صفراً إذا كانت القيمة معدومة	990
إظهار علامة $ قبل الرقم .	$99
إظهار العلامة العشرية .	.
إظهار فواصل بين كل ثلاثة أرقام (فاصلة الألوف) .	,
إظهار علامة السالب (-) يمين الرقم إذا كان سالباً .	MI

دالة التحويل TO_DATE .

الشكل العام :

```
TO_DATE(CHAR , 'fmt')
```

تقوم هذه الدالة بتحويل البيانات الحرفية التي تعبر عن تاريخ إلى بيانات من نوع تاريخ DATE بشكل معين يسمى format ويعبر عنه بالجزء fmt .

والمثال التالي يوضح كيفية استخدامها .

مثال (١٤) :

```
SQL> SELECT TO_DATE( 'FEBRUARY 22, 1981' , 'MONTH DD, YYYY' )
  2  FROM dual

TO_DATE( 'FEBRUARY 22, 1981' , 'MONTH DD, YYYY' )
-------------------------------------------------
22-FEB-1981
```

في المثال السابق تم تحويل البيانات الحرفية ('FEBRUARY 22, 1981') والتي تعبر عن تاريخ إلى بيانات تاريخ وتم عرض هذا التاريخ .

ملحوظة هامة : يجب أن تكون صيغة التاريخ المدخل FORMAT متناسبة مع البيانات الحرفية التي يراد تحويلها إلى تاريخ . كما هو موضح في المثال السابق .

دالة التحويل **TO_NUMBER** .

الشكل العام :

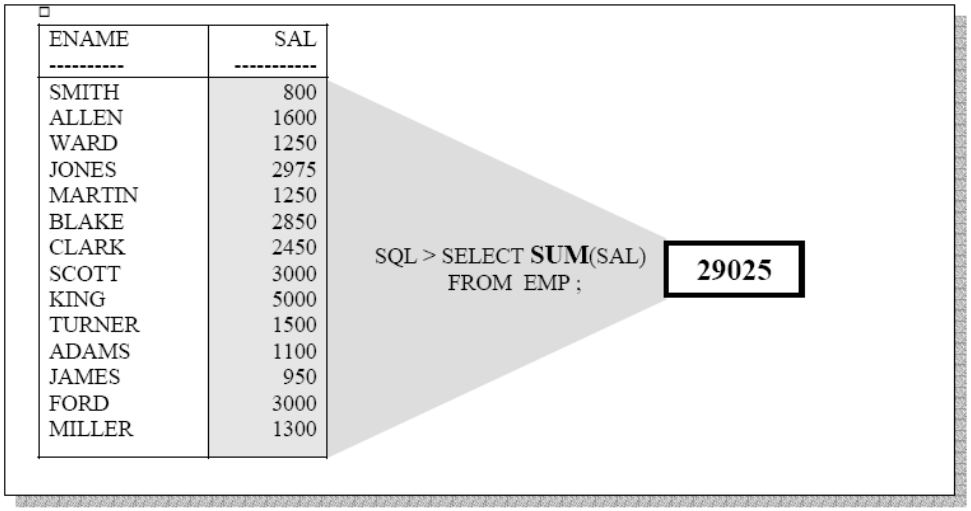
```
TO_NUMBER(CHAR , 'fmt')
```

تستخدم هذه الدالة لتحويل البيانات الحرفية التي تعبر عن رقم إلى بيانات رقمية NUMBER بشكل معين يسمى format ويعبر عنه بالجزء fmt .

مثلاً إذا كان لدينا عمود وصفت بياناته حرفية داخل جدول ومسجل به بيانات رقمية ونريد إجراء عمليات حسابية على هذا العمود هنا لابد من استخدام هذه الدالة لتحويل البيانات داخل العمود إلى بيانات رقمية .

الدوال التجميعية لأكثر من صف والتي تسمى (GROUP FUNCTIONS) هي دوال تتعامل مع بيانات مجموعة من الصفوف لآخراج قيمة واحدة فقط ، فمثلاً إذا أردنا إيجاد مجموع ما يأخذه الموظفون من رواتب فبدلاً من جمع كل الرواتب معاً ، نستخدم دالة من الدوال التجميعية تسمى SUM وهذه الدالة تقوم بعملية جمع للرواتب وتكون نتيجتها قيمة واحدة فقط وهي المجموع .

والشكل التالي يوضح كيفية عمل الدالة التجميعية SUM .



كما هو واضح من الشكل أن الدالة SUM قامت بجمع كل الرواتب لآخراج قيمة واحدة فقط وهي (29025) ، وبالمثل عندما نريد إيجاد أكبر راتب من بين رواتب الموظفين فهناك أيضاً دالة من الدوال التجميعية تقوم بعمل ذلك ، أي إن هذا النوع من الدوال يتعامل مع أكثر من صف لآخراج قيمة واحدة فقط .

أنواع الدوال التجميعية : Type of Group Functions

الجدول التالي يبين أنواع الدوال التجميعية ووظيفة كل دالة :

الدالـــــة FUNCTION	وظيفتهـــــا
SUM	دالة تستخدم لإيجاد المجموع لعدد من القيم
MAX	دالة تستخدم لإيجاد أكبر قيمة من بين مجموعة من القيم
MIN	دالة تستخدم لإيجاد أقل قيمة من بين مجموعة من القيم
AVG	دالة تستخدم لإيجاد المتوسط الحسابي لمجموعة من القيم
COUNT	دالة تستخدم لإيجاد عدد القيم أو عدد الصفوف وهذه الدالة تتجاهل القيم الفارغة NULL عند عملية العد .
STDDEV DEVIATION	دالة تستخدم لإيجاد الانحراف المعياري لمجموعة من القيم
VARIANCE	دالة تستخدم لإيجاد مقدار التباين (التشتت) لمجموعة من القيم

وجميع هذه الدوال تتجاهل القيم الفارغة NULL داخل الأعمدة .

أمثلة على الدوال التجميعية .

مثال (١) :

```
SQL> SELECT SUM(sal) , MAX(sal) , MIN(sal) , AVG(sal)
  2  FROM emp ;
```

SUM(SAL)	MAX(SAL)	MIN(SAL)	AVG(SAL)
29025	5000	800	2073.21429

في المثال السابق تم عرض مجموع رواتب الموظفين وأكبر راتب وأقل راتب والمتوسط الحسابي للرواتب .

ملحوظة هامة : كل من الدالتين MAX و MIN تتعامل مع جميع البيانات ، أي عند استخدامهما مع البيانات الحرفية تكون النتيجة حسب الترتب الأبجدي . لاحظ المثال التالي .

مثال (٢) :

```
SQL> SELECT MAX(ename) , MIN(ename)
  2  FROM emp ;
```

MAX(ENAME)	MIN(ENAME)
WARD	ADAMS

في المثال السابق تم عرض أول أسماء الموظفين حسب الترتيب الأبجدي وآخر الاسماء .

مثال (٣) :

```
SQL> SELECT AVG( NVL(comm , 0) )
  2  FROM emp ;
```

AVG(NVL(comm , 0))
157.14286

في المثال السابق تم عرض المتوسط الحسابي لمكافآت الموظفين ، ونلاحظ استخدام الدالة NVL هنا لحل مشكلة القيم الفارغة NULL الموجودة داخل العمود comm وذلك لأن المتوسط الحسابي عبارة عن مجموع القيم مقسومة على عدد الموظفين وبما أن الدوال التجميعية تتجاهل القيم الفارغة فكان لابد من استخدام الدالة NVL حتى يتم القسمة على عدد الموظفين وهم (14) موظفاً .

إذا لم نستخدم الدالة NVL فسوف يتم القسمة على أربعة (4) بدلاً من (14) وبالتالي يكون المتوسط الحسابي خطأ كما هو واضح كالتالي :

```
SQL> SELECT AVG( comm )
  2  FROM emp ;
```

AVG(comm)
550

المتوسط الحسابي هنا خطأ لأنه تم القسمة على أربعة بينما عدد الموظفين هو ١٤ موظفاً .

التعامل مع الدالة COUNT .

يوجد حالتان للدالة count هما :

- COUNT(*) .
- COUNT(column) .

- تستخدم الدالة count(*) لعد جميع الصفوف داخل الجدول بما فيها الصفوف المكررة والصفوف التي تحتوي على القيم الفارغة NULL ، وإذا كانت جملة الاستفسار تحتوي على شرط where فإنها تقوم بعد الصفوف حسب جملة الشرط .
- تستخدم الدالة count(column) لعد قيم أو بيانات عمود معين مع تجاهل القيم NULL .

والأمثلة التالية توضح ذلك .

مثال (٤) :

```
SQL> SELECT COUNT(*),COUNT(comm), COUNT(deptno)
  2  FROM emp  ;
```

COUNT(*)	COUNT(COMM)	COUNT(DEPTNO)
14	4	14

تم عرض إجمالي عدد الصفوف داخل جدول الموظفين وهم 14 صفاً ، وأيضاً تم عرض الموظفين الذين يأخذون مكافآت وعددهم داخل الجدول (4) ، نلاحظ هنا أن الدالة COUNT(comm) قد تجاهلت القيم الفارغة NULL داخل العمود COMM . كما تم عرض عدد الإدارات داخل العمود deptno وهم (14) بالرغم من أنها مكررة داخل العمود أي إن الدالة COUNT تقوم بعد القيم المكررة .

مثال (٥) :

```
SQL> SELECT COUNT(comm) , COUNT(*)
  2  FROM emp
  3   WHERE deptno=30 ;
```

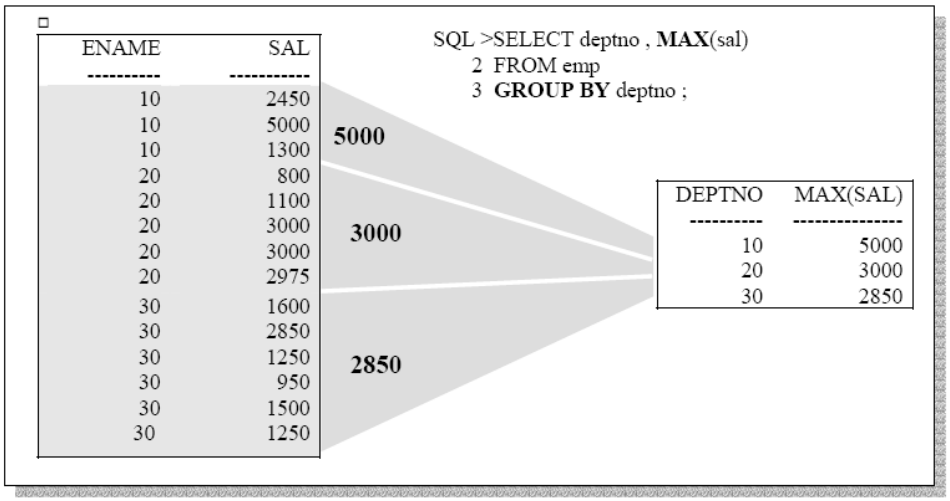
COUNT(COMM)	COUNT(*)
4	6

في المثال السابق تم عرض عدد الموظفين الذين يأخذون عمولة في الإدارة رقم 30 وكان عددهم أربعة بينما كان إجمالي عدد الموظفين في الإدارة هو ستة موظفين .

إنشاء مجموعات من البيانات باستخدام الجزء GROUP BY :

بفرض أننا نريد إيجاد أكبر راتب يأخذه موظف في كل إدارة معنى ذلك أننا نقسم الموظفين داخل الجدول إلى مجموعات حسب الإدارات ثم نقوم بإيجاد أكبر راتب في كل إدارة ، ولعمل ذلك نستخدم الجزء GROUP BY والذي يقوم بتقسيم البيانات إلى مجموعات على حسب عمود معين أو أكثر .

والشكل التالي يبين تأثير استخدام الجزء group by .



من الشكل السابق يتضح أنه تم تقسيم الموظفين إلى مجموعات حسب رقم الإدارة وكان أكبر راتب في الإدارة رقم 10 هو (5000) وأكبر راتب في الإدارة رقم 20 هو (3000) وأيضاً أكبر راتب في الإدارة رقم ٣٠ هو (2850) .

مثال (٦) :

```
SQL> SELECT deptno , AVG(sal)
  2  FROM emp
  3   GROUP BY deptno
  4   ORDER BY AVG(sal)  ;
```

DEPTNO	AVG(SAL)
30	1566.66667
20	2175
10	2916.66667

في المثال السابق تم عرض المتوسط الحسابي لمرتبات الموظفين في كل إدارة كما تم ترتيب المخرجات تصاعدياً حسب المتوسط الحسابي .

ملحظات على استخدام الدوال التجميعية :

- عند كتابة أي عمود داخل قائمة SELECT لابد من كتابته مع الجزء GROUP BY وذلك لأن الدوال التجميعية تتعامل مع عدة صفوف .
- يمكن استخدام الجزء ORDER BY لترتب الصفوف مع الدوال التجميعية كما هو مبين في المثال السابق .
- لا يمكن استخدام الدوال التجميعية في الجزء WHERE ولكن نستخدم الجزء HAVING بدلاً منها .

مثال (٧) :

```
SQL> SELECT deptno , AVG(sal)
  2  FROM emp
  3  ORDER BY AVG(sal)  ;

ERROR at line 1 :
ORA-00937: not a single-group  group function
```

عند تنفيذ هذا المثال تم إعطاء رسالة خطا وذلك لأننا كتبنا أسم العمود (deptno) ضمن قائمة select ولم نكتبه ضمن الجزء group by ولتصحيح هذا الخطا انظر المثال رقم (٦) .

رسالة خطا

مثال (٨) :

```
SQL> SELECT deptno , AVG(sal)
  2  FROM emp
  3  WHERE AVG(sal) > 2000
  4  GROUP BY deptno   ;

ERROR at line 3 :
ORA-00934: group function  is not allowed here  ◄──── رسالة خطأ
```

عند تنفيذ المثال السابق تظهر رسالة خطأ ، وذلك لأننا استخدمنا الدالة AVG(sal) داخل الجزء WHERE وهذا غير مسموح ، ولتصحيح هذا الخطأ لابد من استبدال جملة الشرط WHERE بجملة شرط خاصة بالدوال التجميعية وهي HAVING كما في المثال رقم (٩) :

مثال (٩) :

```
SQL> SELECT deptno , AVG(sal)
  2  FROM emp
  3  GROUP BY deptno
  4  HAVING AVG(sal) > 2000 ;
```

DEPTNO	AVG(SAL)
10	2916.66667
20	2175

في هذا المثال تم عرض المتوسط الحسابي لمرتبات الموظفين في كل إدارة بشرط أن تكون المتوسطات الحسابية للمرتبات أكبر من (2000) . لاحظ استخدام الجزء HAVING لتطبيق شرط معين على الدوال التجميعية .

يمكن استخدام جميع أجزاء جملة SELECT بشرط مراعاة الملاحظات السابقة عند الاستخدام . كما في المثال التالي .

مثال (١٠) :

```
SQL> SELECT job , SUM(sal)
  2  FROM emp
  3   WHERE job not like 'SALES%'
  4   GROUP BY job
  5   HAVING SUM(sal) >5000
  6   ORDER BY SUM(sal) ;
```

JOB	SUM(SAL)
ANALYST	6000
MANAGER	8275

في هذا المثال تم استخدام جميع أجزاء جملة SELECT لعرض مجموع رواتب الموظفين حسب كل وظيفة ، بشرط استبعاد الوظيفة التي تتضمن الحروف (SALES) وأيضاً استبعاد المجموع الأصغر من (5000) وترتيب المخرجات حسب مجموع الرواتب .

لاحظ عدم استخدام الدوال التجميعية في الجزء WHERE لوجود الجزء GROUP BY .

١ – اكتب جملة استعلام لعرض أعلى وأقل قيمة للرواتب وأيضاً المتوسط الحسابي للرواتب وقم بملاحظه الناتج ؟ بحيث تظهر النتيجة كالتالي :

MAXIMUM	MINIMUM	SUM	AVERAGE
5000	800	29025	2073

٢ – اكتب جملة استعلام لعرض أسماء الوظائف وأعلى وأقل راتب لهذه الوظائف كل على حده ؟ بحيث تظهر النتيجة كالتالي :

JOB	MAXIMUM	MINIMUM
ANALYST	3000	3000
CLERK	1300	800
MANAGER	2975	2450
PRESIDENT	5000	5000
SALESMAN	1600	1250

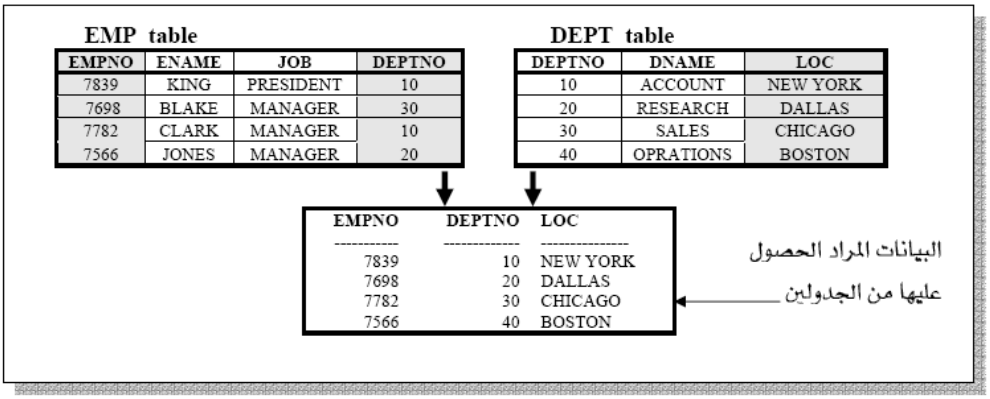
٣ – اكتب جملة استعلام لعرض الوظائف وعدد الموظفين في كل وظيفة ؟ بحيث تظهر النتيجة كالتالي :

JOB	COUNT(*)
ANALYST	2
CLERK	4
MANAGER	3
PRESIDENT	1
SALESMAN	4

٤ – اكتب جملة استعلام لعرض عدد المديرين ؟ بحيث تظهر النتيجة كالتالي :

NUMBER OF MANAGERS
6

في بعض الأحيان نريد أن نقوم بعرض بيانات من أكثر من جدول لعمل تقارير مفيدة وشاملة ، فمثلاً لو أردنا عرض رقم الموظف ورقم الإدارة التابع لها وموقع هذه الإدارة نجد أننا لابد من الحصول على هذه البيانات من جدول الموظفين وجدول الإدارات لكون رقم الموظف موجود في جدول الموظفين ورقم الإدارة موجود في جدول الإدارات وأيضاً موجود في جدول الموظفين بينما موقع الإدارة موجود في جدول الإدارات ، كما هو موضح بالشكل التالي :



وللحصول على تلك البيانات لابد من عمل ربط بين الجدولين . وسوف نقوم في هذا الفصل بشرح أنواع الربط المختلفة وكيفية عمل كل نوع من هذه الأنواع .

تعريف الربط : Join defination

هو عبارة عن ربط بين جدولين أو أكثر للحصول على بيانات من تلك الجداول .

أنواع الربط : Types of Joins

توجد عدة أنواع من الربط (Joins) وهي كالتالي :

- الربط بالتساوي Equijoin .
- الربط بعدم التساوي Non-Equijoin .
- الربط الخارجي Outer Join .
- الربط الداخلي في نفس الجدول Self Join .

ويتم عمل هذه الأنواع عن طريق جملة الاستفسار SELECT وبخاصة في جزء الشرط WHERE .

الربط بالتساوي : Equijoin

في هذا النوع من الربط يتم ربط جدولين أو أكثر عن طريق عمودين متساويين ، العمود الأول عادةً ما يكون مفتاح أساس (Primary Key) في الجدول الأول والعمود الثاني يكون عبارة عن عمود ربط (Foreign Key) في الجدول الثاني .

والشكل التالي يبين الربط بالتساوي بين جدول الموظفين وجدول الإدارات عن طريق العمود (deptno) الموجود في كل منهم .

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

Foreign Key **Primary Key**

سوف يتم عمل الربط بين الجدولين باستخدام جملة SELECT عن طريق العمودين المشار إليهما بالسهم ، كما في المثال التالي .

مثال (١) :

```
SQL> SELECT emp.empno , emp.ename , emp.deptno ,
  2              dept.deptno , dept.loc
  3   FROM emp , dept
  4   WHERE emp.deptno=dept.deptno ;
```

شرط الربط بين الجدولين

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7369	SMITH	20	20	DALLAS
7499	ALLEN	30	30	CHICAGO
7521	WARD	30	30	CHICAGO
7566	JONES	20	20	DALLAS
7654	MARTIN	30	30	CHICAGO
7698	BLAKE	30	30	CHICAGO
7782	CLARK	10	10	NEW YORK
7788	SCOTT	20	20	DALLAS
7839	KING	10	10	NEW YORK
7844	TURNER	30	30	CHICAGO
7876	ADAMS	20	20	DALLAS

في المثال السابق تم عرض بيانات من جدولين عن طريق الربط بالتساوي ، وسوف نقوم بشرح كل جزء من أجزاء جملة SELECT على حدا كالتالي:

- في قائمة SELECT تم عرض رقم الموظفين وأسمائهم وأرقام إداراتهم من جدول الموظفين (EMP) ، كما تم عرض رقم الإدارات وموقعها من جدول الإدارات (DEPT) ، ونلاحظ هنا أننا حددنا من أين تأتي البيانات عن طريق كتابة أسم الجدول قبل أسم العمود ويفصل بينهما العلامة (.) كالتالي (emp . empno) .
- في الجزء FROM تم كتابة أسماء الجداول التي ستأتي منها البيانات كالتالي (FROM emp,dept) .
- في الجزء WHERE تم كتابة شرط الربط بين الجدولين وهذا الشرط مهم جداً لإتمام عملية الربط ، وبدون هذا الشرط سوف تكون النتيجة ليس لها معنى أو فائدة .

أستخدام الاسماء المستعارة للجداول :

يمكن استخدام الاسماء المستعارة للجداول لتسهل عملية كتابة الأعمدة ، فمثلاً نقوم باستبدال أسم الجدول (emp) بالحرف (e) ، وأسم الجدول (dept) بالحرف (d) كالتالي :

```
SQL> SELECT e.empno , e.ename , e.deptno ,
  2             d.deptno , d.loc
  3  FROM emp  e , dept  d
  4  WHERE e.deptno=d.deptno ;
```

وطبعاً سوف تكون النتيجة مماثلة تماماً للمثال رقم (١) .

عندما نريد عرض البيانات الموجودة في المثال رقم (١) ولكن للموظف KING فقط فإننا هنا لابد من زيادة شرط على جملة SELECT كالتالي :

مثال (٢) :

```
SQL> SELECT e.empno , e.ename , e.deptno ,
  2             d.deptno , d.loc
  3  FROM emp  e , dept  d <----------  (e , d) الاسماء المستعارة
  4  WHERE e.deptno=d.deptno
  5  AND e.ename = upper('king') ;
```

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7839	KING	10	10	NEW YORK

الربط بعدم التساوي : Non-Equijoin

يتم استخدام هذا النوع من الربط عندما لا توجد علاقة مباشرة بين الجدولين المراد ربطهما أي إننا لا نستخدم فيه علامة التساوي (=) ، ولكن لابد من وجود علاقة غير مباشرة مثل شرط معين ينطبق عليهما ، مثلاً عندنا جدول الموظفين وفيه عمود الراتب (SAL) وأيضاً لدينا جدول آخر يسمى جدول الفئات (SALGRADE) في هذا الجدول يتم وضع فئات للرواتب وكل فئة تنحصر بين أعلى راتب وأقل راتب ، فمثلاً الموظف الذي يأخذ راتب (3000) يتبع الفئة رقم (٤) كما هو واضح من جدول الفئات أدناه وبذلك نجد أن هناك علاقة بين الجدولين وهي أن كل راتب في جدول الموظفين لابد أن يتبع فئة معينة داخل الجدول (SALGRADE) أي إنه يقع بين أعلى قيمة وأقل قيمة داخل الجدول .

والشكل التالي يوضح جدول الموظفين EMP وجدول الفئات SALGRADE والعلاقة بينهم .

EMP table

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250

SALGRADE table

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

SAL	GRADE
5000	5
2850	4
2450	4
2975	4
1250	2

البيانات التي توضح فئة كل راتب من رواتب الموظفين والتي حصلنا عليها من الجدولين

المثال التالي يوضح كيفية الربط بين الجدولين عن طريق الربط بعدم التساوي Non-Equijoin .

مثال (٣) :

```
SQL> SELECT e.ename , e.sal , s.grade
  2  FROM emp e , salgrade s
  3  WHERE e.sal BETWEEN s.losal AND s.hisal ;
```

شرط الربط بين الجدولين

ENAME	SAL	GRADE
SMITH	800	1
ADAMS	1100	1
JAMES	950	1
WARD	1250	2
MARTIN	1250	2
MILLER	1300	2
ALLEN	1600	3
TURNER	1500	3
JONES	2975	4
BLAKE	2850	4
CLARK	2450	4

في المثال السابق تم عرض أسماء الموظفين ورواتبهم من جدول الموظفين وعرض الفئات لكل راتب من جدول الفئات وذلك عن طريق الربط بعدم التساوي بين الجدولين وتم ذلك في جزء الشرط WHERE e.sal BETWEEN s.losal AND s.hisal والذي يوضح شرط انحصار رواتب الموظفين في جدول الموظفين بين أقل راتب وأكبر راتب في جدول الفئات .

الربط الخارجي : Outer Join

يتم استخدام هذا النوع من الربط عندما توجد بيانات في أحد الجداول ولكنها لا تظهر في حالة الربط بالتساوي (Equijoin) بين الجدولين أي إنها غير مطابقة لشرط التساوي ونريد لهذه البيانات أن تظهر ، في هذه الحالة نقوم بالربط بين الجدولين باستخدام الربط بالتساوي ولكن نضيف الجزء (+) بجانب العمود الفاقد للبيانات ويسمى الربط في هذه الحالة بالربط الخارجي (Outer Join) ، فمثلاً توجد الإدارة رقم (40) في جدول الإدارات ولكن لا يوجد بها موظفين مسجلين في جدول الوظائف عند استخدام الربط بالتساوي فإن هذه الإدارة لا تظهر في المخرجات لعدم تطابق شرط التساوي عليها ، ولإظهارها لابد من استخدام الربط الخارجي .

المثال التالي يبين كيفية الربط بين جدولين باستخدام الربط الخارجي (Outer Join) لإظهار كافة البيانات الموجودة بالجدولين سواء كانت البيانات المطابقة لشرط التساوي أو غير المطابقة لشرط التساوي .

مثال (٤) :

```
SQL> SELECT   e.empno , e.ename , d.deptno , d.dname
  2  FROM     emp e , dept d
  3  WHERE    e.deptno(+)=d.deptno  ;
```

علامة الربط الخارجي

EMPNO	ENAME	DEPTNO	DNAME
7782	CLARK	10	ACCOUNTING
7839	KING	10	ACCOUNTING
7934	MILLER	10	ACCOUNTING
7369	SMITH	20	RESEARCH
7876	ADAMS	20	RESEARCH
7902	FORD	20	RESEARCH
7788	SCOTT	20	RESEARCH
7566	JONES	20	RESEARCH
7499	ALLEN	30	SALES
7698	BLAKE	30	SALES
7654	MARTIN	30	SALES
7900	JAMES	30	SALES
7844	TURNER	30	SALES
7521	WARD	30	SALES
		40	OPERATIONS

هذه الإدارة ظهرت لاستخدامنا الربط الخارجي

في المثال السابق تم عرض أرقام الموظفين وأسمائهم من جدول الموظفين كما تم عرض أرقام الإدارات وأسمائها من جدول الإدارات باستخدام الربط الخارجي (Outer Join) ولذلك قد ظهرت الإدارة رقم (40) بالرغم أنها غير مطابقة لشرط التساوي أي لا يوجد بها موظفين مسجلين في جدول الموظفين .

الربط الداخلي لنفس الجدول : Self Join

EMP table

EMPNO	ENAME	JOB	MGR
7839	KING	PRESIDENT	
7698	BLAKE	MANAGER	7839
7782	CLARK	MANAGER	7839
7566	JONES	MANAGER	7839
7654	MARTIN	SALESMAN	7698

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698

EMP (MANAGER)

EMPNO	ENAME
7839	KING
7698	BLAKE
7782	CLARK
7566	JONES
7654	MARTIN

عندما ندقق في جدول الموظفين نجد أنه يحتوي على عمود يسمى (MGR) هذا العمود يمثل رقم المدير للموظف ، فنجد أن الموظف (BLAKE) مديره هو الموظف ذو الرقم (7839) أي إنه الموظف (KING) ، ومن ذلك يتضح لنا أن هناك علاقة بين عمود المدير (MGR) ورقم الموظف (EMPNO) فالمدير هو نفسه عبارة عن موظف أي يوجد له رقم موظف داخل العمود (EMPNO) ، أي إننا نستطيع ربط الجدول بنفسه عن طريق العمودين (MGR) و (EMPNO) .

ولعمل هذا النوع من الربط لابد من تقسيم جدول الموظفين إلى جدولين أحدهما يمثل جدول الموظفين ونسميه مثلاً (WORKER) والآخر يمثل جدول المدراء ونسميه مثلاً (MANAGER) كما هو واضح من الشكل السابق ، ونقوم بعد ذلك بربط الجدولين عن طريق الربط بالتساوي (Equijoin) .

والمثال التالي يوضح كيفية عمل الربط الداخلي لنفس الجدول .

مثال (٥):

```
SQL> SELECT WORKER.empno , WORKER.ename , MANAGER.ename manager
  2  FROM emp worker , emp manager
  3  WHERE worker.mgr = manager.empno ;
```

شرط الربط بين الجدولين

EMPNO	ENAME	MANAGER
7369	SMITH	FORD
7499	ALLEN	BLAKE
7521	WARD	BLAKE
7566	JONES	KING
7654	MARTIN	BLAKE
7698	BLAKE	KING
7782	CLARK	KING
7788	SCOTT	JONES
7844	TURNER	BLAKE
7876	ADAMS	SCOTT
7900	JAMES	BLAKE
7902	FORD	JONES
7934	MILLER	CLARK

في المثال السابق تم عرض أرقام الموظفين وأسمائهم من جدول الموظفين(WORKER) كما تم عرض أسماء المدراء من جدول المدراء (MANAGER) عن طريق استخدام الربط الداخلي لنفس الجدول (Self Join) .

الربط بين أكثر من جدولين :

لربط أكثر من جدولين لابد أن تتوفر علاقة ما بينهم جميعاً علماً بأنه لابد أن تكون جمل الشرط المستخدمة في عملية الربط تساوي (عدد الجداول -١) ، أي إذا كان لدينا جدولان فلابد من أن هناك شرط واحد لربطهما ، وإذا كان لدينا ثلاثة جداول فيجب أن يتوفر شرطان لربطهما وهكذا . ولابد من وضع المعامل (AND) بين هذه الشروط .

المثال التالي يوضح كيفية ربط ثلاثة جداول معاً لعرض بيانات من كل منهم .

مثال (٦) :

```
SQL> SELECT e.empno , e.ename , e.sal , d.dname , s.grade
  2  FROM   emp e , dept d , salgrade s
  3  WHERE  e.deptno=d.deptno
  4  AND    e.sal BETWEEN s.losal and s.hisal ;
```

شرط الربط بين جدول الموظفين وجدول الإدارات

شرط الربط بين جدول الموظفين وجدول الفئات

EMPNO	ENAME	SAL	DNAME	GRADE
7369	SMITH	800	RESEARCH	1
7876	ADAMS	1100	RESEARCH	1
7900	JAMES	950	SALES	1
7521	WARD	1250	SALES	2
7654	MARTIN	1250	SALES	2
7934	MILLER	1300	ACCOUNTING	2
7499	ALLEN	1600	SALES	3
7844	TURNER	1500	SALES	3
7566	JONES	2975	RESEARCH	4
7698	BLAKE	2850	SALES	4
7782	CLARK	2450	ACCOUNTING	4
7788	SCOTT	3000	RESEARCH	4
7902	FORD	3000	RESEARCH	4
7839	KING	5000	ACCOUNTING	5

في المثال السابق تم ربط ثلاثة جداول مع بعضها وهي جدول الموظفين (EMP) وجدول الإدارات (DEPT) وجدول الفئات (SALGRADE) وذلك لعرض أرقام الموظفين وأسمائهم ورواتبهم وأسماء الإدارات التابعين لها والفئات التي تنتمي لها رواتبهم .

لاحظ أننا استخدمنا شرطين لربط ثلاثة جداول .

١ – اكتب جملة استعلام لعرض أسماء وأرقام إدارات وأسم الإدارات للموظفين المسجلين في الإدارة رقم (30) ، بحيث تظهر النتيجة كالتالي :

ENAME	DEPTNO	DNAME
ALLEN	30	SALES
WARD	30	SALES
MARTIN	30	SALES
BLAKE	30	SALES
TURNER	30	SALES
JAMES	30	SALES

٢ – اكتب جملة استعلام لعرض أسماء الوظائف ومكانها للموظفين المسجلين بالإدارة رقم (٣٠) مع منع التكرار في الوظائف . بحيث تظهر النتيجة كالتالي :

JOB	LOC
CLERK	CHICAGO
MANAGER	CHICAGO
SALESMAN	CHICAGO

٣ – اكتب جملة استعلام لعرض أسماء ووظائف وأسماء الإدارات للموظفين المسجلين بالإدارة التي تقع في مدينة (DALLAS) ؟ بحيث تظهر النتيجة كالتالي :

ENAME	JOB	DNAME
SMITH	CLERK	RESEARCH
JONES	MANAGER	RESEARCH
SCOTT	ANALYST	RESEARCH
ADAMS	CLERK	RESEARCH
FORD	ANALYST	RESEARCH

٤ – اكتب جملة استعلام لعرض أسماء ووظائف وأسماء إدارات ورواتب وفئات الرواتب للموظفين المسجلين بالإدارة رقم (10) ؟ بحيث تظهر النتيجة كالتالي :

ENAME	JOB	DNAME	SAL	GRADE
MILLER	CLERK	ACCOUNTING	1300	2
CLARK	MANAGER	ACCOUNTING	2450	4
KING	PRESIDENT	ACCOUNTING	5000	5

لمعرفة وفهم أهمية الاستعلامات الفرعية(Subqueries) أجب عن هذا سؤال .

من هم الموظفون الذين يأخذون رواتب أكبر من راتب الموظف (Jones) ؟

للإجابة على هذا السؤال لابد أولاً من معرفة ما هو مرتب الموظف (Jones) ، ثم نأخذ هذا المرتب ونبني جملة استعلام تحتوي على عرض بيانات الموظفين بشرط أن تكون رواتبهم أكبر من راتب الموظف (Jones) . أي إننا سوف نقوم بعمل استعلامين كالآتي :

الاستعلام الأول : إيجاد مرتب الموظف (Jones) كالآتي :

```
SQL > SELECT sal
  2  FROM emp
  3  WHERE ename = 'JONES' ;
```

SAL
2975

الاستعلام الثاني : استخدام ناتج الاستعلام الأول لمعرفة الموظفين الذين يأخذون رواتب أكبر من هذا الناتج كالآتي :

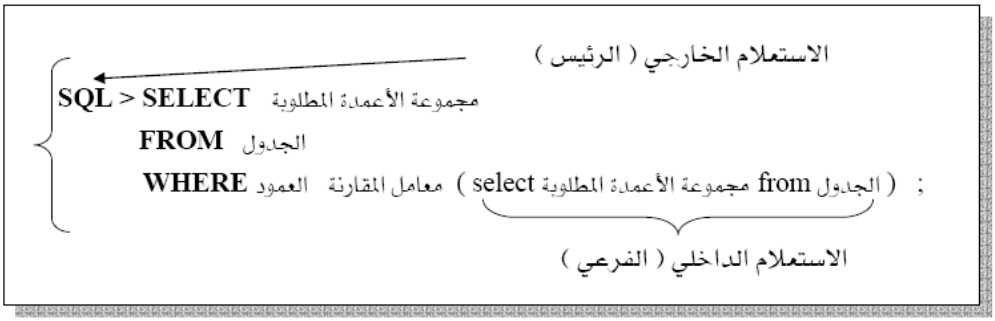
```
SQL > SELECT ename
  2  FROM emp
  3  WHERE sal > 2975 ;
```

ENAME
SCOTT
KING
FORD

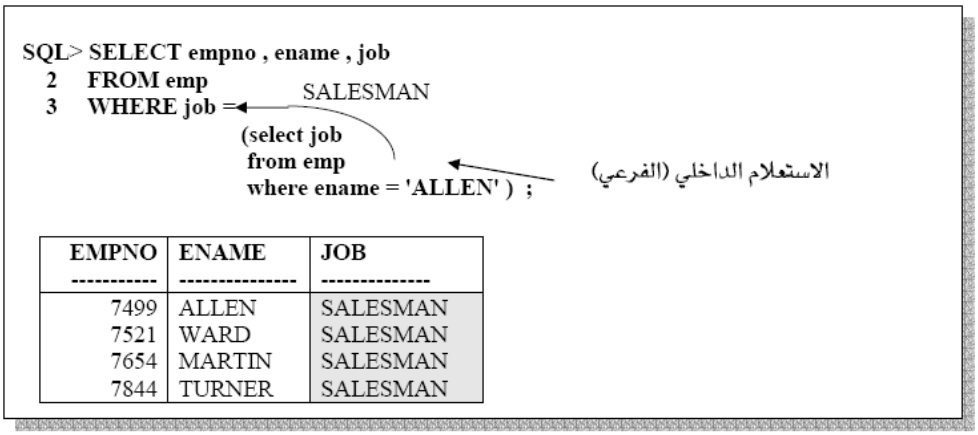
وبالتالي قد حصلنا على الإجابة المطلوبة ولكن بطريقة عادية وطويلة . و إذا أردنا الإجابة على هذا السؤال بجملة استعلام واحدة ، نقوم بدمج الاستعلامين السابقين بحيث يكون الاستعلام الأول استعلاماً داخلياً (فرعي) والاستعلام الثاني والذي سوف يأتي بالإجابة يكون استعلاماً خارجياً (رئيس) كالتالي :

```
SQL > SELECT ename
  2  FROM emp
  3  WHERE sal > (SELECT sal  FROM emp WHERE ename = 'JONES' )  ;
```

الشكل العام لجملة الاستعلامات الفرعية :



مثال (١) : عرض أرقام وأسماء ووظائف الموظفين الذين يعملون على نفس وظيفة الموظف ALLEN .



EMPNO	ENAME	JOB
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7654	MARTIN	SALESMAN
7844	TURNER	SALESMAN

في المثال السابق تم إيجاد وظيفة الموظف (ALLEN) أولاً بعمل استعلام داخلي (فرعي) وكانت النتيجة هي (SALESMAN) ، ثم تم عرض أرقام وأسماء ووظائف الموظفين الذين يعملون بوظيفة (SALESMAN) عن طريق الاستعلام الخارجي (الرئيس) .

ملحوظة هامة : يقوم أوراكل بتنفيذ الاستعلام الداخلي (الفرعي) أولاً ثم ينفذ الاستعلام الخارجي (الرئيس) .

أنواع الاستعلامات الفرعية :

تنقسم الاستعلامات الفرعية إلى ثلاثة أنواع هي :

- استعلام فرعي أحادي الصف ويرجع بصف واحد Single-Row Subquery .
- استعلام فرعي متعدد الصفوف ويرجع بأكثر من صف Multiple-Row Subquery .
- استعلام فرعي متعدد الأعمدة ويرجع بأكثر من عمود Multiple -Column Subquery .

متطلبات وإرشادات الاستعلام الفرعي :

يجب مراعاة الآتي عند استخدام الاستعلام الفرعي .

- يجب وضع الاستعلام الفرعي بين قوسين .
- يجب وضع الاستعلام الفرعي يمين معامل المقارنة (<>,=,>=, <= .. إلخ) .
- يجب التأكد من استخدام المعاملات الأحادية الصف مع الاستعلامات الفرعية الأحادية الصف (Single-Row Subquery) .
- يجب التأكد من استخدام المعاملات المتعددة الصفوف مع الاستعلامات الفرعية المتعددة الصفوف (Multiple-Row Subquery) .

أنواع معاملات المقارنة المستخدمة مع الاستعلام الفرعي :

يوجد نوعان من معاملات المقارنة وهي :

- معاملات أحادية الصف مثل (<> , >= , <= , > , < , =) وتستخدم مع الاستعلامات الفرعية أحادية الصف أي التي ترجع بصف واحد (قيمة واحدة) .
- معاملات متعددة الصف مثل (IN , ANY , ALL) وتستخدم مع الاستعلامات الفرعية متعددة الصفوف أي التي ترجع بأكثر من صف .

أماكن كتابة الاستعلامات الفرعية داخل جملة Select :

يجب كتابة الاستعلامات الفرعية في الأجزاء التالية من جملة SELECT :
(WHERE , HAVING , FROM) .

الاستعلامات الفرعية الأحادية الصف : Single-Row Subqueries

وهي استعلامات (دائماً) تكون نتيجتها صف واحد فقط ولذلك يستخدم معها المعاملات الأحادية الصف (<> , >= , <= , > , < , =) .

مثال (٢) :

قم بعرض أسماء ورواتب وأرقام إدارات الموظفين الذين يعملون في نفس إدارة الموظف (KING) ؟

```
SQL> SELECT ename , sal , deptno
  2   FROM emp
  3   WHERE deptno =                    10
  4                 ( select deptno
  5                   from emp
  6                   where ename='KING' ) ;
```

ENAME	SAL	DEPTNO
CLARK	2450	10
KING	5000	10
MILLER	1300	10

في المثال السابق تم إيجاد رقم إدارة الموظف (KING) أولاً بعمل استعلام داخلي (فرعي) ، ثم عرض أسماء ورواتب وأرقام إدارات الموظفين في الإدارة رقم (10) وهي إدارة الموظف (KING) .

مثال (٣) :

قم بعرض أسماء ووظائف ورواتب الموظفين الذين رواتبهم مساوية لأقل راتب ؟

```
SQL> SELECT ename , job , sal
  2   FROM emp                    800
  3   WHERE sal =
  4                 ( select  MIN(sal)
  5                   from  emp ) ;
```

ENAME	JOB	SAL
SMITH	CLEARK	800

لاحظ في المثال السابق استخدام الدالة التجميعية MIN(SAL) في الاستعلام الفرعي ، وأيضاً يمكن استخدامها في الاستعلام الرئيس كما هو واضح في المثال التالي .

مثال (٤) :

قم بعرض أرقام الإدارات وأقل راتب يأخذه موظف فيها بحيث يكون أقل راتب فيها أكبر من أقل راتب في الإدارة رقم (20) ؟

```
SQL> SELECT deptno , MIN(sal)
  2   FROM emp
  3  GROUP BY deptno
  4  HAVING MIN(sal) >          800
  5                          ( select MIN(sal)
  6                            from emp
  7                             where deptno = 20 ) ;
```

DEPTNO	MIN(SAL)
10	1300
30	950

في المثال السابق تم استخدام الدالة التجميعية MIN(SAL) في الاستعلام الرئيس لذلك استخدمنا HAVING ومعها الاستعلام الفرعي .

مثال (٥) :

اعرض أسماء ووظائف الموظفين الذين يعملون على نفس وظيفة الموظف ذي الرقم (7369) ويأخذون مرتب أكبر من مرتب الموظف ذو الرقم (7876) .

الإجابة :

في هذا المثال ثلاثة استعلامات اثنان منها استعلامات فرعية الأول يأتي بوظيفة الموظف ذي الرقم (7369) والثاني يأتي براتب الموظف ذي الرقم (7876) والاستعلام الثالث هو الرئيس .

إجابة المثال رقم (٥)

```
SQL> SELECT ename , job
  2  FROM emp
  3  WHERE job =                    <-- CLERK
  4                (select job
  5                 from emp
  6                 where empno=7369 )
  7  AND   sal >                    <-- 1100
  8                (select sal
  9                 from emp
 10                 where empno=7876)  ;
```

ENAME	JOB
MILLER	CLERK

ملحوظة هامة : يجب أن تكون نتيجة الاستعلام الداخلي (الفرعي) عبارة عن صف واحد فقط حتى تتم المقارنة بشكل صحيح ، وإلا فسوف تظهر رسالة خطأ كما في المثال التالي :

مثال (٦) :

اعرض أسماء ووظائف الموظفين الذين يعملون على نفس وظيفة الموظف (SMITH) ؟

إجابة خطأ

```
SQL> SELECT ename , job
  2   FROM  emp
  3   WHERE   job =
  4                  (select   job                <-- رسالة خطأ
  5                   from emp )   ;

ERROR : ORA-01427 : single –row subquery return more than one row.
```

ظهرت رسالة خطأ في المثال السابق وذلك لأن الاستعلام الفرعي رجع بأكثر من وظيفة أي أكثر من صف في حين أننا مستخدمون المعامل الأحادي ، ولتصحيح هذا الخطأ نكتبه كالتالي :

```
SQL> SELECT ename , job
  2   FROM  emp
  3   WHERE   job =
  4                  (select   job
  5                   from emp
                      where ename='SMITH' )   ;
```

سوف يرجع الاستعلام الداخلي (الفرعي) بوظيفة الموظف (SMITH) وهي (CLERK) أي إنه رجع بصف واحد فقط .

الاستعلامات الفرعية المتعددة الصفوف : Multiple-Row Subqueries

وهي استعلامات ترجع دائماً بأكثر من صف ولذلك يستخدم معها المعاملات المتعددة الصفوف (ALL , ANY , IN) .

والجدول التالي يبين معنى المعاملات المتعددة والتي تستخدم لمقارنة مجموعة من الصفوف .

المعامـــل	المعنـــي
IN	المساواة بأي قيمة داخل قائمة
ANY	مقارنة قيمة بأي من قيم داخل قائمة
<ANY	معناها أقل من أكبر قيمة داخل قائمة
>ANY	معناها أكبر من أقل قيمة داخل قائمة
ALL	مقارنة قيمة بكل ما هو موجود داخل قائمة
<ALL	معناها أقل من أقل قيمة داخل قائمة
>ALL	معناها أكبر من أعلى قيمة داخل قائمة

مثال (٧) :

قم بعرض أسماء ورواتب وأرقام إدارات الموظفين الذين يأخذون رواتب مساوية لأقل راتب في كل إدارة .

```
SQL> SELECT ename , sal , deptno
  2  FROM emp
  3  WHERE  sal  IN ( select min(sal)
  4                   from emp
  5                   group by deptno ) ;
```

نتيجة الاستعلام الفرعي هي (800,950,1300)

ENAME	SAL	DEPTNO
SMITH	800	20
JAMES	950	30
MILLER	1300	10

في المثال السابق لابد من إيجاد أقل راتب في كل إدارة وهذا بعمل استعلام فرعي يرجع بأقل الرواتب في كل إدارة وهي (1300 , 950 , 800) ، ثم استخدام هذه الرواتب في عرض البيانات بشرط أن يكون الراتب مساوياً لإحدى هذه القيم .

ملحوظات على المثال السابق:

نتيجة الاستعلام الفرعي عبارة عن مجموعة من القيم (800,950,1300) تمثل مجموعة من الصفوف وهي أقل رواتب في كل إدارة .

تم استخدام المعامل المتعدد (IN) لأننا هنا نقارن الراتب بمجموعة من الصفوف .

في الاستعلام الرئيس تم عرض بيانات الموظفين الذين يأخذون راتباً مساوياً لأي من (800,950,1300) .

مثال (٨) :

قم بعرض أرقام وأسماء ووظائف ورواتب الموظفين الذين رواتبهم أقل من راتب الموظفين أصحاب الوظيفة (CLERK) دون عرض موظفي الوظيفة (CLERK) .

```
SQL> SELECT empno , ename ,  job , sal
  2   FROM emp
  3   WHERE sal <ANY
  4                    ( select SAL
  5                      from emp
  6                     where job ='CLERK' )
  7   AND job <>'CLERK' ;
```

نتيجة الاستعلام الفرعي هي (800,950,1100,1300)

EMPNO	ENAME	JOB	SAL
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250

في المثال السابق تم عمل استعلام فرعي لإيجاد رواتب الموظفين المسجلين بالوظيفة (CLERK) وكانت (800,950,1100,1300) ثم استخدام هذه القيم في الاستعلام الرئيس لعرض بيانات الموظفين الذين يأخذون رواتب أقل من أي من هذه القيم ، كما تم استبعاد الموظفين المسجلين بالوظيفة (CLERK) . ونلاحظ أن المعامل (SAL<ANY) معناه بشرط أن يكون الراتب أقل من أكبر قيمة من القيم (800,950,1100,1300) .

مثال (٩):

قم بعرض أرقام وأسماء ووظائف ورواتب الموظفين الذين رواتبهم أكبر من كل المتوسطات الحسابية للمرتبات في كل إدارة .

```
SQL> SELECT empno , ename , job , sal
  2  FROM emp
  3  WHERE sal >ALL
  4                ( select AVG(sal)
  5                  from emp
  6                  group by deptno ) ;
```

نتيجة الاستعلام الفرعي هي المتوسطات الحسابية
(2916.6667, 2175,1566.6667)

EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000

في المثال السابق تم عمل استعلام فرعي لإيجاد المتوسطات الحسابية لرواتب الموظفين وكانت (2916.6667 , 2175 , 1566.6667) ثم استخدام هذه القيم في الاستعلام الرئيس لعرض بيانات الموظفين الذين رواتبهم أكبر من كل هذه القيم ، ونلاحظ أن المعامل (SAL>ALL) معناه بشرط أن يكون الراتب أكبر من أعلى قيمة من القيم (2916.6667, 2175,1566.6667) .

١ – اكتب جملة استعلام لعرض أسماء وتواريخ تعيين الموظفين الذين لهم نفس رقم إدارة الموظف BLAKE ؟ بحيث تظهر النتيجة كالتالي :

ENAME	HIREDATE
ALLEN	20-FEB-81
WARD	22-FEB-81
MARTIN	28-SEP-81
TURNER	08-SEP-80
JAMES	03-DEC-81

٢ – قم بعرض أرقام وأسماء الموظفين الذين يأخذون راتباً أكبر من المتوسط الحسابي لجميع الرواتب ، ورتب الناتج تنازلياً حسب الراتب ؟ بحيث تظهر النتيجة كالتالي :

EMPNO	ENAME
7839	KING
7902	FORD
7788	SCOTT
7566	JONES
7698	BLAKE
7782	CLARK

٣ – اعرض أسماء وأرقام إدارات ووظائف الموظفين المسجلين في الإدارة (DALLAS) ؟ بحيث تظهر النتيجة كالتالي :

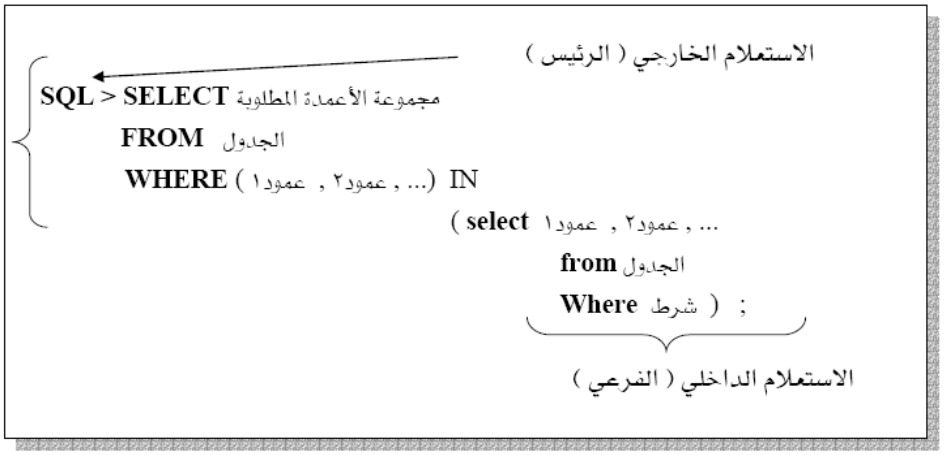
ENAME	DEPTNO	JOB
JONES	20	MANAGER
FORD	20	ANALYST
SMITH	20	CLERK
SCOTT	20	ANALYST
ADMES	20	CLERK

٤ - قم بعرض أرقام وأسماء ووظائف ورواتب الموظفين الذين يأخذون رواتب أقل من راتب الموظفين أصحاب الوظيفة (SALESMAN) دون عرض موظفي الوظيفة (SALESMAN) . ثم قم بالتحقق من الناتج ؟ .

في الفصل السابق تم معرفة نوعين من الاستعلامات الفرعية النوع الأول كان الاستعلام الفرعي ذو الصف الواحد والذي يرجع دائماً بصف واحد ، أما النوع الآخر فهو الاستعلام متعدد الصفوف الذي يرجع بأكثر من صف ، ويوجد نوع ثالث من الاستعلامات وهو الاستعلام متعدد الأعمدة والذي يرجع بأكثر من عمود وأكثر من صف ولمقارنة ناتج هذا النوع لابد من استخدام المعاملات متعددة الصفوف والتي تم دراستها في الفصل السابق .

والشكل التالي يبين الصيغة العامة التي نستخدمها في كتابة هذا النوع من الاستعلامات .

الصيغة العامة لجملة الاستعلامات الفرعية متعددة الأعمدة :



نلاحظ أن الاستعلام الفرعي يرجع بأكثر من عمود والتي يتم مقارنتها بالأعمدة الموجودة في جملة WHERE .

في هذا الفصل سوف نتعرف على جدول جديد يتم من خلاله شرح الأمثلة التي توضح كيفية استخدام الاستعلامات متعددة الأعمدة وهذا الجدول هو جدول يمثل الأصناف المسجلة داخل طلب معين للأصناف .

ITEM Table جدول الأصناف

رقم الطلب	رقم الصنف	رقم المنتج	السعر الحقيقي	الكمية	إجمالي الطلب
ORDER_ID	ITEM_ID	PRODUCT_ID	ACTUAL_PRICE	QUANTITY	TOTAL
603	1	100860	32	7	224
604	1	100890	58	3	174
604	2	100861	42	2	84
604	3	100860	32	12	384
605	1	100861	45	100	4500
605	2	100870	2.8	500	1400
605	3	100890	58	5	290
605	4	101860	24	50	1200
605	5	101863	9.5	100	950
605	6	102130	3.4	10	34
606	1	102130	3.4	1	3.4

هذا الجدول موجود في قاعدة البيانات الافتراضية تحت المستخدم DEMO وكلمة المرور أيضاً DEMO .

مثال (١) :

في هذا المثال نريد عرض رقم الطلب ورقم المنتج والكمية لجميع الأصناف التي لها نفس رقم المنتج والكمية للأصناف المسجلة داخل الطلب رقم 605 ، مع استبعاد الأصناف للطلب 605 من النتيجة .

أولاً : عمل استعلام فرعي متعدد الأعمدة لإيجاد رقم المنتج والكمية للأصناف داخل الطلب رقم 605 . كالآتي :

```
SQL> select  product_id , quantity
  2  from item
  3  where order_id=605  ;
```

نتيجة الاستعلام الفرعي السابق هي :

PRODUCT_ID	QUANTITY
100861	100
100870	500
100890	5
101860	50
101863	100
102130	10

ثانياً : عمل استعلام رئيس يتم فيه مقارنة رقم المنتج والكمية لجميع الأصناف برقم المنتج والكمية للأصناف داخل الطلب رقم 605 أي بالنتيجة السابقة . كالآتي :

```
SQL> SELECT order_id ,  product_id , quantity
  2   FROM item
  3   WHERE (product_id , quantity) IN
  4                                  (select product_id , quantity
  5                                   from item
  6                                   where order_id=605 )
  7  AND              order_id <> 605  ;
```

الاستعلام الداخلي (الفرعي)
يرجع بالنتيجة المذكورة أعلاه

PRDER_ID	PRODUCT_ID	QUANTITY
617	100861	100
617	100870	500
616	102130	10

في المثال السابق تم عرض الأصناف التي لها نفس رقم المنتج والكمية للأصناف المسجلة داخل الطلب رقم 605 . ونلاحظ في المثال أننا قد أضفنا شرطاً لاستبعاد الأصناف المسجلة في الطلب رقم (605) من النتيجة النهائية . وذلك لعرض النتيجة للأصناف الآخرى فقط .

نتيجة الاستعلام الرئيس عندما يرجع الاستعلام الفرعي بالقيمة NULL :

عندما تكون نتيجة الاستعلام الفرعي NULL أي إنه لا توجد نتيجة للاستعلام الفرعي ، في هذه الحالة لا يكون للاستعلام الرئيس أي نتيجة ويعطي الرسالة التالية :

(no rows selected) وتعني أنه لا توجد أي صفوف مطابقة للشرط في جملة where الموجودة داخل الاستعلام الرئيس . والمثال التالي يبين ذلك :

مثال(٢) :

اعرض رقم الطلب ورقم المنتج والكمية لجميع الأصناف التي لها نفس رقم المنتج والكمية للأصناف المسجلة داخل الطلب رقم 700 ، مع استبعاد الأصناف للطلب 700 من النتيجة .

```
SQL> SELECT order_id , product_id , quantity
  2  FROM item
  3  WHERE (product_id , quantity) IN
  4                                (select product_id , quantity
  5                                 from item
  6                                 where order_id=700 )
  7  AND          order_id <> 700  ;

no rows selected
```

الاستعلام الداخلي (الفرعي) يرجع بالنتيجة NULL

هنا كانت نتيجة الاستعلام الفرعي NULL لعدم وجود الطلب ذي الرقم (700) مسجل داخل الجدول (غير مسجل) ، ولهذا كانت نتيجة الاستعلام الرئيس رسالة تبين بأنه لا توجد أي صفوف تم اختيارها .(no rows selected) .

استخدام الاستعلام الفرعي في جملة FROM .

يمكن استخدام الاستعلام الفرعي في الجزء FROM من جملة الاستعلام SELECT ، وذلك لعمل مصدر بيانات آخر غير الجداول ، فكما نعرف أنه دائماً يستخدم أسم الجدول ضمن الجزء FROM لأنه من مصادر البيانات ، والمثال التالي يوضح كيفية استخدام الاستعلام الفرعي ضمن الجزء FROM .

مثال (٣) :

اعرض أسماء ورواتب وأرقام إدارات والمتوسط الحسابي للرواتب للموظفين الذين يأخذون رواتب أعلى من المتوسط الحسابي لإداراتهم .

```
SQL> SELECT e.ename,e.sal,e.deptno,esub.salavg
  2  FROM emp e, (select deptno,avg(sal) salavg
  3               from emp                          <-- الاستعلام الداخلي (الفرعي)
  4               group by deptno) esub             يعامل معاملة الجدول
  5  WHERE e.deptno = esub.deptno
  6  AND e.sal > esub.salavg  ;
```

ENAME	SAL	DEPTNO	SALAVG
KING	5000	10	2916.66667
FORD	3000	20	2175
SCOTT	3000	20	2175
JONES	2975	20	2175
ALLEN	1600	30	1566.66667
BLAKE	2850	30	1566.66667

في المثال السابق تم عمل استعلام فرعي في الجزء FROM وذلك لإيجاد المتوسطات الحسابية للرواتب حسب الإدارات ثم تم عمل استعلام رئيس لعرض البيانات المطلوبة بشرط أن يكون مرتب الموظفين الذين سيتم عرضهم أكبر من المتوسط الحسابي للرواتب داخل الإدارة التابع لها .

نلاحظ هنا أننا استخدمنا الاستعلام الفرعي وكأنه جدول موجود باسم (esub) ويتكون هذا الجدول من عمودين الأول يمثل أرقام الإدارات والثاني يمثل المتوسطات الحسابية للرواتب في كل إدارة ، وبياناته كالآتي :

```
SQL> select deptno,avg(sal) salavg
  2  from emp
  3  group by deptno  ;
```

DEPTNO	SALAVG
10	2916.66667
20	2175
30	1566.66667

ولأننا هنا نقوم بعرض بيانات من جدولين ، فلابد من ربطهما ببعض أي إننا نقوم بربط جدول الموظفين emp مع الجدول (esub) المكون بالاستعلام الفرعي بالجملة التالية :

```
WHERE e.deptno = esub.deptno
```

١ – اكتب جملة استعلام لعرض أسماء وأرقام الإدارة ورواتب الموظفين الذين لهم نفس رقم الإدارة وراتب الموظفين الذين يأخذون عمولة ؟ بحيث تظهر النتيجة كالتالي :

ENAME	DEPTNO	SAL
MARTIN	30	1250
WARD	30	1250
TURNER	30	1500
ALLEN	30	1600

٢ – اعرض أسماء وأسماء الإدارة ورواتب الموظفين الذين لهم نفس رواتب وعمولة الموظفين المسجلين في الإدارة (DALLAS) ؟ بحيث تظهر النتيجة كالتالي :

ENAME	DNAME	SAL
SMITH	RESEARCH	800
ADAMS	RESEARCH	1100
JONES	RESEARCH	2975
FORD	RESEARCH	3000
SCOTT	RESEARCH	3000

٣ - اعرض أسماء وتواريخ التعيين ورواتب الموظفين الذين لهم نفس راتب وعمولة الموظف (SCOTT) ؟ بحيث تظهر النتيجة كالتالي :

ENAME	HIREDATE	SAL
FORD	03-DEC-81	3000

لقد قمنا في الفصول السابقة بشرح جملة الاستعلام SELECT والتي تُمكننا من عرض أي بيانات من جدول أو عدة جداول بالشكل المطلوب ، وفي هذا الفصل سوف نتعرف على قسم مهم جداً من أقسام لغة SQL وهو لغة التعامل مع البيانات (DML) وهذه اللغة تُمكننا من التعامل مع البيانات داخل الجدول ، حيث يتم من خلالها إضافة سجل أو سجلات (ROWS) جديدة إلى الجدول أو التعديل في سجل أو سجلات معينة أو حذف سجل أو سجلات معينة .

وتتكون لغة التعامل مع البيانات (DML) من عدة جمل وهي كالتالي :

- جملة إضافة بيانات إلى الجدول INSERT INTO .
- جملة التعديل في بيانات الجدول UPDATE .
- جملة حذف بيانات من الجدول DELETE FROM .

وسوف نقوم بشرح كل جملة من هذه الجمل بالتفصيل في هذا الفصل .

إضافة سجل أو عدد من السجلات إلى جدول معين (INSERT INTO).

إضافة سجل أو سجلات جديدة إلى جدول هي عملية إضافة بيانات جديدة إلى جدول معين عن طريق استخدام جملة الإضافة (INSET INTO) .

الصيغة العامة لإضافة سجلات جديدة إلى جدول .

```
SQL > INSERT INTO جدول ( ..... عمود٣ , عمود٢ , عمود١ )
      VALUES          (..... قيمة٣ , قيمة٢ , قيمة١ ) ;
```

شرح الشكل العام :

- جدول : اسم الجدول المطلوب إضافة سجلات فيه .
- (عمود١ ,عمود٢ , عمود٣ .....) : أسماء الأعمدة المطلوب إدخال البيانات إليها .
- (قيمة١ , قيمة٢ , قيمة ٣ .....) : القيم المطلوب إضافتها إلى الأعمدة .

القواعد التي يجب التقيد بها عند الإضافة:

- يجب أن يكون عدد القيم التي سيتم إدخالها مساوياً لعدد الأعمدة المذكورة في جملة INSERT .
- يجب أن تكون القيم مرتبة بنفس ترتيب الأعمدة المراد إدخال القيم إليها ، حيث إن (القيمة١) سوف تسجل في (العمود١) وهكذا . كما يجب أن تكون القيم أيضاً من نفس نوع بيانات الأعمدة .
- عند إدخال قيم التاريخ والنصوص لابد من وضعها داخل علامتي تنصيص فرديتين ' ' .
- يجب إدخال قيماً للأعمدة التي لا تقبل قيماً فارغة NULL مثل أعمدة المفتاح الأساسي (Primary key) مثلاً (empno) في جدول الموظفين .
- يجوز عدم ذكر أسماء الأعمدة في جملة INSERT وفي هذه الحالة لابد من إدخال جميع قيم الأعمدة الموجودة في الجدول حسب ترتيب الأعمدة داخل الجدول مع مراعاة نوع البيانات لكل عمود .

ملحوظة : يمكن عرض أسماء الأعمدة ونوع بياناتها داخل جدول معين باستخدام الأمر (DESC table) ، راجع الفصل الثاني صفحة (١٠) .

مثال (١) : إضافة سجل جديد إلى جدول الإدارات أي إضافة إدارة جديدة .

```
SQL> INSERT  INTO  dept (deptno , dname , loc )
  2  VALUES             ( 50 , 'DEVELOPMENT' , 'DETROIT') ;
```

جدول الإدارات (DEPT) قبل الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

جدول الإدارات (DEPT) بعد الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON
50	DEVELOPMENT	DETROIT

السجل (الصف) الذي تم إضافته

في المثال السابق تم إضافة إدارة جديدة برقم (50) واسمها (DEVELOPMENT) وموقعها (DETROIT) ، ونلاحظ في هذا المثال اننا أخذنا في الاعتبار القيود التي يجب اتباعها عند الإضافة كما هو مذكور سابقاً . أي إننا أضفنا رقم الإدارة داخل العمود deptno الذي نوع بياناته (number) ، وأضفنا أسم الإدارة وموقعها ووضعناهما داخل علامتي التنصيص الفردية (' ') وذلك لأن نوع بياناتهما حروف (varchar2) .

إضافة قيمة فارغة (NULL) إلى عمود :

يتم إضافة القيمة NULL إلى الأعمدة بطريقتين :

الأولى : عدم كتابة الأعمدة المراد تسجيل القيمة NULL بها في الجزء INSERT .

الثانية : أن نكتب الأعمدة ولكن نكتب قيمتها NULL داخل الجزء VALUES .

بشرط أن تقبل الأعمدة هذه القيمة أي إنها ليست عليها قيود مثل (primary key) كما في المثال التالي :

مثال (٢) : إضافة سجل جديد إلى جدول الإدارات يحتوي هذا السجل على رقم الإدارة واسمها فقط .

```
SQL> INSERT  INTO  dept (deptno , dname )
  2   VALUES              ( 60 , 'MIS' ) ;
```

جدول الإدارات (DEPT) بعد الإضافة

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	

قيمة فارغة NULL

في المثال السابق تم إضافة إدارة جديدة برقم (60) واسمها (MIS) ، ولكن لم يتم إضافة موقعها ولذلك تم تسجيل القيم NULL داخل العمود (LOC) ، ومن الممكن أن نكتب هذا المثال كتالي :

```
SQL> INSERT  INTO  dept (deptno , dname , loc )
  2   VALUES              ( 60 , 'MIS' , NULL ) ;
```

إضافة قيم خاصة داخل الأعمدة:

إذا أردنا إضافة تاريخ اليوم الحالي (SYSDATE) إلى العمود (HIREDATE) الموجود داخل جدول الموظفين وذلك عند إضافة سجل جديد أي إضافة بيانات موظف جديد فيمكن عمل ذلك كما هو واضح من المثال التالي :

مثال (٣) : إضافة بيانات موظف جديد إلى جدول الموظفين .

```
SQL> INSERT INTO emp
  2                 (empno , ename , job, mgr , hiredate , sal , comm , deptno)
  3  VALUES
  4   (7196 , 'AHMED' , 'SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10) ;
```

في المثال السابق تم إضافة بيانات موظف جديد إلى جدول الموظفين وتم وضع تاريخ التعيين له بحيث يكون تاريخ اليوم الحالي SYSDATE .

المثال السابق يمكن كتابته بحيث لا نذكر أسماء الأعمدة في الجزء INSERT ، وهنا لابد من كتابة كل القيم حسب ترتيب الأعمدة داخل الجدول كالآتي :

```
SQL> INSERT INTO emp
       VALUES
         (7196 , 'AHMED' , 'SALESMAN' , 7782 , SYSDATE , 2000 , NULL , 10) ;
```

إضافة سجلات عن طريق المتغيرات البديلة Substitution Variables .

تمكنا لغة SQL من عمل متغيرات تسمى المتغيرات البديلة ، وهي عبارة عن مخزن مؤقت للبيانات ومن خلالها يتم تخزين قيم معينة داخل هذه المتغيرات وأثناء تنفيذ جملة SQL يتم استبدال هذه المتغيرات بقيمتها ، ويتم تعريفها أثناء كتابة جملة SQL وذلك بوضع العامة (&) قبل أسم المتغير ، كما تظهر رسالة تسأل عن قيمة هذه المتغيرات أثناء التنفيذ . والمثال التالي يوضح كيفية إضافة سجل جديد باستخدام المتغيرات البديلة substitution Variables .

مثال (٤) : إضافة إدارة جديدة إلى جدول الإدارات باستخدام المتغيرات البديلة .

```
SQL> INSERT INTO dept ( deptno , dname , loc )
  2  VALUES             (&dept_id , '&dept_name' , '&dept_loc' ) ;

Enter value for dept_id : 80
Enter value for dept_name : EDUCATION
Enter value for dept_loc : ATLANTA
```

المتغيرات البديلة

في المثال السابق تم إضافة بيانات إدارة جديدة عن طريق استخدام المتغيرات البديلة التالية :

- **(&dept_id)** متغير يمثل رقم الإدارة وعند التنفيذ تم تخزين رقم (80) داخله .
- **(&dept_name)** متغير يمثل اسم الإدارة وعند التنفيذ تم تخزين (EDUCATION) داخله .
- **(&dept_loc)** متغير يمثل موقع الإدارة وعند التنفيذ تم تخزين (ATLANTA) داخله .

ويمكن تكرار هذا الأمر عدة مرات لإضافة أكثر من إدارة دون الحاجة إلى كتابة الأمر مرة آخرى .

وبهذا قد أضفنا إدارة جديدة إلى جدول الإدارات وكأننا كتبنا الأمر التالي :

```
SQL> INSERT INTO dept ( deptno , dname , loc )
  2  VALUES             ( 80 , 'EDUCATION' , 'ATLANTA' ) ;
```

يمكن استخدام المتغيرات البديلة بدلاً من أسماء الجداول والأعمدة كما في المثال التالي .

مثال (٥) : إضافة إدارة جديدة إلى جدول الإدارات باستخدام المتغيرات البديلة بدلاً من أسم جدول أو أسم عمود .

```
SQL> INSERT INTO &dept_table ( &dept_id , dname , loc )
  2  VALUES             ( 80 , 'EDUCATION' , 'ATLANTA' ) ;

Enter value for dept_table : dept
Enter value for dept_id : deptno
```

في المثال السابق تم استخدام المتغير البديل (&dept_table) بدلاً من أسم الجدول (dept) وأيضاً المتغير البديل (&dept_id) بدلاً من أسم العمود (deptno) .

إضافة سجلات جديدة عن طريق نسخها من جدول آخر :

افرض أن لدينا جدولاً اسمه (managers) به بيانات المديرين ، وجدول آخر (emp) به بيانات الموظفين ونريد إضافة سجلات الموظفين الذين يعملون على وظيفة مدير(manager) إلى جدول المديرين ، يتم عمل ذلك كما في المثال التالي :

مثال (٦) : إضافة سجلات الموظفين الذين يعملون على وظيفة مدير من جدول الموظفين إلى جدول المدراء (managers) .

```
SQL> INSERT INTO managers (id , name , salary , hiredate)
  2                        select   empno , ename , sal , hiredate
  3                        from    emp
  4                        where  job='MANAGER'  ;
```

في المثال السابق تم استخدام الاستعلام الجزئي التالي :

```
2                        select empno , ename , sal , hiredate
3                        from emp
4                        where job='MANAGER'  ;
```

وذلك لإيجاد الموظفين الذين يعملون بوظيفة مدير ومن ثم إضافتهم إلى جدول المديرين . لاحظ هنا عدم كتابة الجزء (values) .

التعديل في بيانات سجل أو سجلات معينة داخل جدول (UPDATE).

التعديل في سجل أو سجلات معينة داخل جدول هي عملية تعديل بيانات عمود أو عدة أعمدة عن طريق استخدام جملة التعديل (UPDATE) .

الصيغة العامة لتعديل البيانات داخل الجدول .

```
SQL > UPDATE جدول
      SET   عمود١ = قيمة١ , عمود٢ = قيمة٢ .....
      WHERE   شرط   ;
```

شرح الشكل العام:

- جدول : اسم الجدول المطلوب تعديل سجلات فيه .
- عمود١ ، عمود٢ : أسماء الأعمدة المطلوب التعديل فيها .
- قيمة ١ ، قيمة ٢ : القيم الجديدة المراد وضعها بدلاً من القيم القديمة .
- شرط : شرط لاختيار سجلات (صفوف) معينة للتعديل فيها ، وبدون هذا الشرط فسوف يتم التعديل في جميع السجلات .

القواعد التي يجب التقيد بها عند التعديل :

- يجب أن يكون نوع البيانات الجديدة من نفس نوع بيانات الأعمدة المطلوب التعديل فيها.
- عند تعديل قيم التاريخ أو النصوص يجب وضع القيم الجديدة بين علامتي التنصيص الفردية (' ') .
- يجب أخذ الحذر عند كتابة الجزء WHERE في جملة التعديل لتحديد أي الصفوف التي سوف يتم التعديل فيها .

مثال (٧) : تعديل أسم الإدارة رقم (30) ليصبح (EDUCATION) بدلاً من (SALES)

```
SQL> UPDATE   dept
  2  SET dname='EDUCATION' ◄——— القيمة الجديدة لاسم الإدارة رقم (30)
  3  WHERE deptno=30 ;

1 row updated ◄——— رسالة تدل على أن التعديل تم في الإدارة رقم (30) فقط
```

في المثال السابق تم تعديل أسم الإدارة رقم (30) من (SALES) إلى (EDUCATION) باستخدام أمر التعديل (UPDATE dept) في جدول الإدارات ، ولاحظ اختيار رقم الإدارة (30) باستخدام الجزء WHERE ، ماذا يحدث لو لم نحدد الإدارة رقم (30) ؟ أي ماذا يحدث لو الغينا جملة الشرط WHERE من المثال السابق ؟

عند عدم كتابة الشرط في عملية التعديل أي لم يتم تحديد الصف المراد التعديل فيه فإنه يتم التعديل في جميع الصفوف كما في المثال التالي :

مثال (٨) : تعديل أسم الإدارة رقم (30) ليصبح (EDUCATION) بدلاً من (SALES)

```
SQL> UPDATE   dept
  2   SET dname='EDUCATION'  ;

4 row updated ←——— رسالة تدل على أن التعديل تم في جميع الصفوف
```

في المثال السابق تم تعديل أسم جميع الإدارات إلى (EDUCATION) ، ولذلك لابد أن من الحذر عند التعديل في صف معين فلابد من تحديد الصف المراد التعديل فيه باستخدام الجزء (WHERE) .

التعديل في أكثر من عمود :

في الأمثلة السابق تم التعديل في عمود واحد فقط وهو عمود أسم الإدارات ، فإذا أردنا التعديل في أكثر من عمود ، فمثلاً لتعديل رقم الإدارة والوظيفة للموظف (BLAKE) ليصبح مثل الموظف (WARD) ، فماذا نفعل ؟ انظر المثال التالي :

مثال (٩) :

```
SQL> UPDATE   emp
  2   SET (job , deptno) = (select  job , deptno  from emp  where ename='WARD')
  3   WHERE ename='BLAKE' ;
                                        (MANAGER    30)  نتيجة الاستعلام الفرعي
1 row updated
```

في المثال السابق تم عمل استعلام فرعي متعدد الأعمدة لإيجاد وظيفة ورقم الإدارة للموظف (WARD) ، ومن ثم استخدامهما في الجزء SET لتعديل وظيفة ورقم الإدارة للموظف (BLAKE) وبهذا فقد تم التعديل في العمودين (job , deptno) للموظف (BLAKE) .

حذف سجل أو سجلات معينة داخل جدول (DELETE FROM).

حذف سجل أو سجلات معينة داخل جدول هي عملية إلغاء بيانات عمود أو عدة أعمدة عن طريق استخدام جملة الحذف (DELETE FROM) .

الصيغة العامة لحذف البيانات من الجدول .

```
SQL > DELETE  FROM  جدول
      WHERE   شرط    ;
```

شرح الشكل العام :

- جدول : اسم الجدول المطلوب حذف سجل أو سجلات منه .
- شرط : شرط لاختيار سجلات (صفوف) معينة لحذفها ، وبدون هذا الشرط فسوف يتم حذف جميع السجلات (الصفوف) .

القواعد التي يجب التقيد بها عند الحذف :

- يجب الحذر عند كتابة الجزء WHERE في جملة الحذف لتحديد أي الصفوف التي سوف يتم حذفها ، عندما لا نكتب جزء الشرط في جملة الحذف فإنه يتم حذف جميع صفوف الجدول كاملة .

مثال (١٠) : حذف الإدارة رقم (40) من جدول الإدارات .

```
SQL> DELETE FROM dept
  2    WHERE deptno = 40  ;

1 row deleted.
```

في المثال السابق تم حذف الإدارة رقم (40) من جدول الإدارات ، أي إنه تم حذف صف كامل من الجدول .

مثال (١١) : حذف جميع الموظفين من جدول الموظفين .

```
SQL> DELETE FROM emp  ;

14 row deleted.
```

في المثال السابق تم حذف جميع الصفوف من جدول الموظفين وذلك لعدم تحديد الصف أو الصفوف المراد حذفها أي إننا لم نكتب جملة الشرط WHERE في المثال السابق .

(تنبيه : الرجاء عدم تنفيذ الأمر السابق للمحافظة على بيانات الجدول ، وإذا كنت قد فعلت فاكتب الأمر (ROLLBACK) لاسترجاع البيانات) .

مثال (١٢) : حذف جميع الموظفين المسجلين في الإدارة التي لها اسم (SALES) .

```
SQL> DELETE FROM  emp
  2    WHERE deptno=( select deptno  from dept  where  dname='SALES' )  ;

6  row deleted.
```

نتيجة الاستعلام الفرعي هو الإدارة رقم (30)

في المثال السابق تم عمل استعلام فرعي لإيجاد رقم الإدارة التي لها أسم (SALES) من جدول الإدارات وناتج الاستعلام هو (30) ، ومن ثم استخدام هذا الرقم لحذف جميع الموظفين المسجلين في الإدارة رقم (30) ، وبالتالي فقد تم حذف أكثر من صف من جدول الموظفين .

ماذا يحدث لو أردت حذف الإدارة رقم (10) من جدول الإدارات ؟ للإجابة على هذا السؤال دعنا نرى هذا المثال .

مثال (١٣) :

```
SQL> DELETE FROM  dept
  2    WHERE deptno= 10   ;

ERROR at line 1 :
ORA-02292 : integrity constraint ( USR.EMP_DEPTNO_FK)
Vaiolated –child record found
```

عند حذف الإدارة رقم (10) من جدول الإدارات ظهرت رسالة خطأ تبين أنه تم انتهاك قيد أو شرط ربط جدول الإدارات بجدول الموظفين ، ومعنى ذلك أنه لا يمكن حذف الإدارة رقم (10) وذلك بسبب أنه يوجد موظفون مسجلون في هذه الإدارة داخل الجدول (emp) . وذلك لوجود ربط

بين الجدولين عن طريق العمود (deptno) فهو بالنسبة لجدول الإدارات يعتبر مفتاح أساسي (Primary key) وبالنسبة لجدول الموظفين فيعتبر مفتاح ربط (Foreign Key) .

عمليات قواعد البيانات (Database Transactions)

يوجد عملية تبدأ عند قيامنا بالإضافة أو التعديل أو الحذف في قاعدة البيانات هذه العملية تسمى Database transactions وهي عملية انتقال البيانات من مرحلة إلى آخرى ، فمثلاً عندما نقوم بعمل إضافة سجل إلى جدول ما فإن هذه الإضافة لا تتم بشكل نهائي إلا إذا قمنا بإصدار أمر من أوامر Database transactions وهو الأمر (COMMIT) وهذا الأمر يعني تثبيت أو حفظ البيانات بشكل نهائي سواء كنا نقوم بعملية إضافة أو تعديل أو حذف بيانات ، فهو يعادل تماماً أمر حفظ (SAVE) . وإليك بعض أوامر Database transactions وهي كالتالي :

- COMMIT يقوم هذا الأمر بحفظ البيانات التي تم إجراء عمليات الإضافة أو التعديل أو الحذف عليها .
- ROLLBACK يقوم هذا الأمر بالتراجع عن عملية الإضافة أو التعديل أو الحذف .

أمر حفظ البيانات بشكل نهائي COMMIT .

هذا الأمر ينفذ بطريقتين :

الأولى : ينفذ هذا الأمر بمجرد كتابته مباشرةً بعد عمليات الإضافة والتعديل والحذف كالتالي :

```
SQL> DELETE FROM  emp
  2    WHERE deptno =  30 ;

6  row deleted.

SQL> COMMIT ;
```

في المثال السابق تم حذف الموظفين الذين يعملون بالإدارة رقم (30) ولكن ليست بصورة نهائية وحتى يتم الحذف بشكل نهائي لابد لنا من إصدار أمر COMMIT كما هو واضح أعلاه .

الطريقة الثانية : يتم إصدار هذا الأمر آلياً عند الخروج من قاعدة البيانات بكتابة الأمر : (SQL > EXIT) أو عندما نقوم بإصدار أمر من أوامر تعريف البيانات (DDL) أو أمر من أوامر التحكم في البيانات (DCL) مثل :

(CREATE VIEW, CREATE TABLE , DROP TABLE , CONNCET .....)

والتي سوف نقوم بدراستها لاحقاً إن شاء الله .

في هذا الفصل سوف نتعرف على قسم مهم جداً من أقسام لغة SQL وهو لغة تعريف البيانات (Data Definition Language) والتي عادةً ما يرمز لها بـ (DDL) ، وهذه اللغة هي التي تمكننا من إنشاء وتعديل وإلغاء أي كائن داخل قاعدة البيانات ، وكما هو معروف أن قاعدة البيانات تتكون من كائنات مختلفة وأهم هذه الكائنات هي الجداول (Tables) والتي سوف نركز في هذا الفصل على كيفية إنشائها والتعديل فيها وإلغائها . وإليك بعض الكائنات التي تتكون منها قاعدة البيانات :

الكائن	وصف الكائن
Table	هو الوحدة الأساسية لمكونات قاعدة البيانات والتي نستخدمها في حفظ البيانات ويتكون من عدة صفوف وأعمدة .
View	المناظير : عبارة عن جزء مؤقت من جدول معين يتكون من عدة صفوف وأعمدة ويستخدم لغرض معين بشكل مؤقت .
Sequence	سلسلة : عبارة عن سلسلة تستخدم لتوليد أرقام متتالية بشكل معين دون تكرار لذلك يفضل استخدامها لتسجيل بيانات المفتاح الأساسي داخل جدول
Index	فهرس : ويستخدم في عملية فهرست بعض الأعمدة لتسهيل عملية البحث فيها عن معلومة معينة ، وأيضاً لتقليل وقت الاستفسارات من الجداول .
Synonym	مرادفات : تستخدم لإعطاء أكثر من أسم على كائن معين .

وكما سبق ذكره فإننا سوف نركز في هذا الفصل على كائن واحد وهو الجداول (Tables) .

أنواع البيانات DATATYPES .

يوجد أنواع للبيانات التي تخزن داخل الجدول وهذه البيانات إما أن تكون بيانات حرفية أو عددية أو بيانات تاريخ أو بيانات آخرى والجدول التالي يبين أنواع البيانات المختلفة :

نوع البيانات	الوصف
Varchar2(الحجم)	تستخدم مع البيانات الحرفية المتغيرة الطول
Char(الحجم)	تستخدم مع البيانات الحرفية الثابتة الطول لابد من تحديد طول البيانات الحرفية
Number(p,s)	تستخدم مع البيانات الرقمية ويمثل الحرف (p) الجزء الصحيح قبل العلامة العشرية ، والحرف (s) يمثل الجزء العشري بعد العامة العشرية .
Date	تستخدم مع بيانات التاريخ والوقت
Long	تستخدم لتمثيل البيانات الكبيرة الحجم التي تصل إلى (2) جيجا بايت .
CLOB – BLOB	تستخدم لتمثيل البيانات الكبيرة مثل الصور والرسومات والتي تصل حجمها إلى أكثر من (4) جيجا بايت
Bfile	تستخدم لتخزين الملفات الكبيرة والخارجية والتي يصل حجمها إلى أكثر من (4) جيجا بايت .

تستخدم الأنواع السابقة في تحديد نوع البيانات لكل عمود عند إنشاء الجدول .

الشروط الواجب توافرها عند اختيار أسم الجداول أو أسماء الأعمدة :

- يجب أن يبدأ اسم الجدول أو أسم العمود بحرف .
- يجب أن لا يزيد طول الاسم عن (30) حرفاً .
- من الممكن أن يتكون من حروف كبيرة وصغيرة وأرقام ورموز خاصة مثل (# , $, _) .
- يجب أن لا يتكرر أسم الجدول أكثر من مرة داخل قاعدة البيانات الواحدة .

- يجب أن لا يتكرر أسم عمود أكثر من مرة داخل الجدول الواحد.
- يجب أن لا يكون من الاسماء المحجوزة لأوراكل مثل (... SELECT , FROM) .
- يفضل أن يكون أسم الجدول له معنى بحيث يُعبر عن نوع بيانات الجدول.

إنشاء الجداول CREATE TABLES .

الصيغة العامة لإنشاء الجداول .

```
SQL > CREATE table  أسم الجدول  (
                        , نوع البيانات  العمود١
                        , نوع البيانات  العمود٢
                        ; (  نوع البيانات  العمود٣
```

مثال (١) : إنشاء جدول الإدارات (dept2) .

```
SQL > CREATE TABLE dept2 (                          نوع البيانات
  2                    deptno  NUMBER(2) ,
  3   أسماء الأعمدة   dname  VARCHAR2(14) ,
  4                    loc     VARCHAR2(13)  ) ;
Table created .
```

في المثال السابق تم إنشاء جدول الإدارات (dept2) والذي يتكون من ثلاثة أعمدة ، العمود الأول نوعه رقمي وطوله (رقمان) ، والعمود الثاني نوعه حرفي وطوله (١٤ حرفاً) ، وكذلك العمود الثالث نوعه حرفي وطوله (١٣ حرفاً) ، وعندما نريد عرض البناء الداخلي للجدول الذي تم إنشاؤه نقوم بكتابة الأمر التالي :

```
SQL > DECRIBE  dept2 ;

Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
DEPTNO                                             NUMBER(2)
DNAME                                              VARCHAR2(14)
LOC                                                VARCHAR2(13)
```

إنشاء الجداول باستخدام الاستعلامات الفرعية :

يمكن إنشاء جدول باستخدام جدول آخر موجود مسبقاً في قاعدة البيانات ، فمثلاً لو أردنا إنشاء جدول يحتوي على بعض الأعمدة الموجودة أصلاً في جدول آخر دون الحاجة إلى تعريف أسماء الأعمدة من جديد أو تحديد نوع البيانات فيها ، فمثلاً إذا أردنا إنشاء جدول خاص يحتوي على بيانات الموظفين للإدارة رقم (30) فقط عن طريق جدول الموظفين (EMP) ، فإننا نقوم بكتابة الأمر كما في المثال التالي :

مثال (٢) : إنشاء جدول للإدارة رقم (30) باستخدام بيانات جدول الموظفين (EMP).

```
SQL > CREATE TABLE dept30
  2   AS
  3   SELECT empno , ename , sal*12 annsal , hiredate
  4   FROM   emp
  5   WHERE deptno = 30 ;
Table created .
```

الاستعلام الفرعي

في المثال السابق تم إنشاء جدول خاص بالإدارة رقم (30) يسمى (dept30) باستخدام جملة select في الاستعلام الفرعي المبين في المثال ، ونلاحظ أننا استخدمنا الجزء (AS) في جملة إنشاء الجدول ، والجدول المسمى بـ (dept30) يتكون من أربعة أعمدة هي (empno,ename,annsal,hiredate)، والجدول الجديد (dept30) يحتوي على بيانات الموظفين في الإدارة (30) أي إننا عندما ننشئ جدولاً باستخدام الاستعلام الفرعي يتم نقل البيانات من الجدول المستخدم في جملة الاستعلام . وعندما نقوم بعرض البناء الداخلي للجدول (dept30) تكون النتيجة كالتالي :

```
SQL > DECRIBE  dept30 ;
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
ANNSAL		NUMBER
HIREDATE		DATE

وعندما نقوم باستعراض البيانات من الجدول (dept30) تكون النتيجة كالتالي :

```
SQL> SELECT * FROM DEPT30 ;
```

EMPNO	ENAME	ANNSAL	HIREDATE
7499	ALLEN	19200	20-FEB-81
7521	WARD	15000	22-FEB-81
7654	MARTIN	15000	28-SEP-81
7698	BLAKE	34200	01-MAY-81
7844	TURNER	18000	08-SEP-81
7900	JAMES	11400	03-DEC-81

وأيضاً في المثال السابق نجد أن أسماء الأعمدة هي نفس أسماء الأعمدة المذكورة في جملة الاستعلام ، وعندما نريد إنشاء نفس الجدول أو أي جدول آخر باسماء محددة غير الاسماء الموجودة في جملة الاستعلام فإننا نكتب الأمر كما في المثال التالي :

مثال (٣) : إنشاء جدول للإدارة رقم (20) باستخدام جدول الموظفين (EMP) يحتوي على أسماء أعمدة مختلفة عن الأعمدة في جدول الموظفين (EMP) .

```
SQL > CREATE TABLE dept20
  6                (emp_id , emp_name , salary , start_date )
  7     AS
  8     SELECT empno , ename , sal , hiredate
  9     FROM   emp
 10     WHERE deptno = 20 ;
Table created .
```

الاستعلام الفرعي

كما هو واضح من المثال السابق فإننا قمنا بإنشاء جدول يسمى (dept20) يحتوي على بيانات الموظفين للإدارة رقم (20) ، ويتكون من الأعمدة التالية :

(emp_id , emp_name , salary , start_date)

التعديل في الجداول باستخدام ALTER TABLE .

توفر لنا لغة الاستعلام أوراكل (SQL) إمكانية مهمة جداً وهي إمكانية التعديل في هيكل (البناء الداخلي) لجدول قد تم أنشاؤه مسبقاً باستخدام الأمر ALTER TABLE ، وعملية التعديل في الجداول تشتمل على ثلاث إمكانيات وهي إما إضافة أعمدة جديدة على الجدول أو التعديل في نوع بيانات عمود معين أو إلغاء عمود معين . كما هو موضح بالجدول التالي :

أوجه التعديل في الجدول باستخدام الأمر ALTER TABLE	
ADD	تستخدم لإضافة أعمدة جديدة إلى الجدول
MODIFY	تستخدم للتعديل في نوع البيانات للجدول
DROP	تستخدم لإلغاء عمود معين من الجدول

مثال (٤) : إضافة عمود جديد يسمى (REGION) إلى جدول الإدارات DEPT2 .

```
SQL> ALTER TABLE   dept2
  2  ADD        ( region VARCHAR2(20) );
Table altered .
```

في المثال السابق قمنا بإضافة عمود جديد يسمى (region) إلى جدول الإدارات وطبعاً هذا العمود لا يحتوي على أية بيانات ويمكن حفظ بيانات المناطق لكل إدارة ، ويظهر هذا العمود كآخر عمود عند الاستعلام ، وللتأكد من إضافة هذا العمود نفذ الأمر التالي :

```
SQL > DECRIBE  dept2 ;

Name                                      Null?     Type
----------------------------------------- -------- ----------------------------
DEPTNO                                              NUMBER(2)
DNAME                                               VARCHAR2(14)
LOC                                                 V ARCHAR2(13)
REGION                                              V ARCHAR2(20)
```

مثال (٥) : التعديل في طول بيانات العمود (DNAME) ليصبح بطول 20 بدلاً من 14 .

```
SQL> ALTER TABLE   dept2
  2  MODIFY        ( dname VARCHAR2(20) );
Table altered .
```

في المثال السابق تم التعديل في نوع بيانات العمود dname ليصبح طوله VARCHAR2(20) بدلاً من VARCHAR2(14) ، ويجب أخذ الملحوظات التالية في الاعتبار عند التعديل في أعمدة الجداول :

- يمكن زيادة حجم (طول) البيانات للأعمدة .
- يمكن تغيير نوع البيانات من نوع إلى آخر بحيث لا يؤثر ذلك في بيانات الأعمدة إذا كانت موجودة .
- لا يمكن تقليل حجم (طول) الأعمدة إذا كانت تحتوي على بيانات .

مثال (٦) : إلغاء العمود المسمى REGION من جدول الإدارات DEPT .

```
SQL> ALTER TABLE   dept2
  2  DROP COLUMN   REGION ;
Table altered .
```

في المثال السابق تم إلغاء العمود (REGION) من جدول الإدارات (DEPT2) ، ويجب أن نأخذ في الاعتبار الملحوظات التالية عندما نريد إلغاء أعمدة من جدول معين :

- يجب أن يكون العمود المراد إلغاؤه فارغاً من البيانات .
- لا يمكن إلغاء أكثر من عمود واحد فقط في الأمر الواحد .
- يجب أن يتبقى عمود واحد على الأقل بعد عملية الإلغاء داخل الجدول .
- لا يمكن استعادة العمود بعد إلغائه .

إلغاء جدول باستخدام الأمر DROP .

عملية إلغاء الجدول هي عبارة عن إلغاء الجدول تماماً من قاعدة البيانات وحذف كل بياناته وكل القيود المتعلقة به ، وبالتأكيد لا يمكن استعادته مرة أخرى .

مثال (٧) : إلغاء الجدول المسمى بـ (DEPT30) .

```
SQL> DROP TABLE    dept 30 ;
Table dropped .
```

في المثال السابق تم إلغاء جدول موظفي الإدارة رقم (30) والذي قد تم إنشاؤه مسبقاً ويسمى بـ(DEPT30) .

تغيير أسم جدول معين إلى اسم آخر باستخدام الأمر RENAME .

لتغيير أسم الجدول (DEPT2) ليصبح اسمه مثلاً (DEPARTMENT) نقوم بكتابة الأمر كما هو مبين في المثال التالي :

مثال (٨) : تغيير أسم جدول الإدارات (DEPT2) ليصبح باسم (DEPARTMENT) .

```
SQL> RENAME    dept2   TO  deptartment
Table renamed.
```

في المثال السابق تم تغيير أسم جدول الإدارات ليصبح بالاسم (department) ، ويجب أن يكون مالك الجدول هو الذي يقوم بتغيير الاسم ، أي إنه لا يمكن لأحد من المستخدمين تغيير أسم الجدول إلا إذا كان هو مالك هذا الجدول .

أنواع القيود Constraints .

الجدول التالي يبين الأنواع المختلفة من القيود ومعنى كل منها :

القيد(constraint)	معنى القيد (Description)
NOT NULL	يمنع هذا القيد ترك عمود معين فارغ (لابد أن يدخل قيمة للعمود) يطبق على مستوى العمود فقط
UNIQUE	يمنع هذا القيد تكرار القيم داخل العمود (القيم داخل العمود وحيدة) يطبق على مستوى العمود أو الجدول .
PRIMARY KEY	يستخدم لعمل مفتاح أساسي داخل الجدول ، والمفتاح الأساسي يتميز بعدم تكرار القيم ، وعدم ترك القيم فارغة أي إنه عبارة عن القيدين السابقين . يطبق على مستوى العمود أو الجدول .
FOREIGN KEY	يستخدم لعمل مفتاح ربط بين جدولين . يطبق على مستوى العمود أو الجدول .
CHECK	يستخدم لاختبار قيمة عمود بحيث لا يقبل هذا العمود إلا قيم حسب شرط معين. يطبق على مستوى العمود أو الجدول .

إنشاء القيود Create a Constrait .

تُنشأ القيود بطريقتين هما:

- عمل القيود أثناء إنشاء الجدول .
- عمل القيود بعد إنشاء الجدول .

وتطبق القيود على مستوى الأعمدة أو على مستوى الجدول . كما سنوضح ذلك من خلال الأمثلة .

القيد PRIMARY KEY .

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول ، ومعنى هذا القيد هو إنشاء مفتاح أساسي (primary key) داخل الجدول وذلك لتمييز عمود معين بحيث إن هذا العمود يكون له خاصيتان هما :

١ – عدم قبول تكرار القيم داخله .

٢ - عدم السماح بترك قيمته فارغة (NULL) .

فمثلاً إذا أردنا تمييز المتدربين فإننا نميزهم عن طريق الرقم الأكاديمي فهذا الرقم يجب أن يكون رقماً وحيداً لا يتكرر فكل متدرب يحمل رقماً أكاديمياً وحيداً خاصاً به وأيضاً يجب أن يكون لكل متدرب رقم أكاديمي فليس من المعقول أن نسجل بيانات متدرب دون تسجيل رقمه ، ولتحقيق ذلك نقوم بعمل عمود داخل جدول المتدربين ونطبق عليه القيد (primary key) .

مثال (١) : إنشاء القيود أثناء إنشاء جدول الإدارات وتطبيقها على مستوى الأعمدة .

```
SQL > CREATE TABLE dept (
  5                    deptno  NUMBER(2)     PRIMARY KEY,
  6   أسماء الأعمدة   dname   VARCHAR2(14)    NOT NULL ,
  7                    loc     VARCHAR2(13)
  8                    ) ;
  9                                          القيود على مستوى الأعمدة
Table created .
```

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيدين على العمودين (deptno,dname) ، القيد الأول هو (primary key) على العمود deptno لجعل هذا العمود (مفتاح أساسي) داخل الجدول ، حيث إن رقم الإدارة يجب أن لا يتكرر داخل العمود وأيضاً عدم ترك قيمته فارغة .

أما القيد الثاني (not null) على العمود dname يمنع ترك قيمة هذا العمود فارغة بدون بيانات فلابد من إدخال أسم الإدارة عند إضافة إدارة جديدة إلى جدول الإدارات . والجدير بالذكر هنا أن القيد (NOT NULL) يطبق فقط على مستوى العمود أي إننا لا نستطيع تطبيقه على مستوى الجدول .

مثال (٢) : إنشاء القيود أثناء إنشاء جدول الإدارات وتطبيقها على مستوى الجدول .

```
SQL > CREATE TABLE dept (
  2                    deptno NUMBER(2) ,
  3                    dname VARCHAR2(14)  NOT NULL ,
  4                    loc       VARCHAR2(13)  ,
  5    CONSTRAINT  dept_deptno_pk  PRIMARY KEY(deptno)  ) ;

Table created .
```

القيد على مستوى الجدول

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (primary key) على مستوى الجدول كما هو واضح في السطر رقم (5) ، والفرق بين هذه الطريقة والطريقة المستخدمة في المثال رقم (١) هو أننا نستطيع حذف القيد إذا كان على مستوى الجدول ، ونلاحظ في هذا المثال أننا سمينا القيد بـ (dept_deptno_pk) وبهذا يمكن حذفه بسهولة كما سنرى لاحقاً .

القيد UNIQUE KEY.

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول ، ومعنى هذا القيد هو عدم السماح بتكرار القيم داخل العمود .

مثال (٣) : إنشاء القيد (UNIQUE) أثناء إنشاء جدول الإدارات وتطبيقه على مستوى الجدول .

```
SQL > CREATE TABLE dept (
  2                    deptno NUMBER(2) ,
  3                    dname VARCHAR2(14) ,
  4                    loc       VARCHAR2(13)  ,
  5    CONSTRAINT  dept_deptno_uk  UNIQUE(dname)  ) ;

Table created .
```

القيد على مستوى الجدول

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (unique) على العمود(dname) وذلك بغرض عدم تكرار أسم الإدارات داخل الجدول ، ونلاحظ أننا طبقنا القيد على مستوى الجدول ويمكن تطبيقه على مستوى العمود .

القيد FOREIGN KEY .

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول ، ويستخدم هذا القيد عندما نريد ربط جدولين ببعض ، فمثلاً لربط جدول الموظفين بجدول الإدارات بغرض معرفة موظفي إدارة معينة ، فإنه لابد من وجود عمود (primary key) داخل جدول الإدارات ونفس هذا العمود يوجد في جدول الموظفين ويسمى (foreign key) ، كما في الشكل التالي :

EMP table

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

DEPT table

DEPTNO	DNAME	LOC
10	ACCOUNT	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPRATIONS	BOSTON

Foreign Key **Primary Key**

ولتنفيذ هذا الربط لابد من تطبيق القيد (foreign key) على العمود (deptno) داخل جدول الموظفين كما هو واضح من المثالين التاليين :

مثال (٤) : إنشاء القيد **(foreign key)** على العمود **deptno** في جدول الموظفين وتطبيقه على مستوى العمود .

```
SQL > CREATE TABLE emp (
  2         empno    NUMBER(4) ,
  3         ename    VARCHAR2(10)  NOT NULL ,
  4         job      VARCHAR2(9) ,
  5         mgr      NUMBER(4) ,
  6         hiredate DATE ,
  7         sal      NUMBER(7,2) ,
  8         comm     NUMBER(7,2) ,
  9         deptno   NUMBER(2) REFERENCES dept(deptno) ) ;

Table created .
```

القيد على مستوى العمود

في المثال السابق تم إنشاء جدول الإدارات (emp) كما تم إنشاء قيد (foreign key) على العمود (deptno) ، لعلك تتساءل أين لفظ (foreign key) هنا ؟ نقول لك أن كلمة (REFERENCES) تشير إلى أنه تم تطبيق القيد (foreign key) على العمود (deptno) الذي يشير إلى العمود (deptno) داخل الجدول DEPT . وبهذه الطريقة تم تطبيق القيد على مستوى العمود .

أما تطبيق القيد على مستوى الجدول فسوف يبين في المثال التالي :

مثال (٥) : إنشاء القيد (foreign key) على العمود deptno في جدول الموظفين وتطبيقه على مستوى الجدول .

```
SQL > CREATE TABLE emp (
  2         empno    NUMBER(4) ,
  3         ename    VARCHAR2(10)  NOT NULL ,
  4         job      VARCHAR2(9) ,
  5         mgr      NUMBER(4) ,
  6         hiredate  DATE ,
  7         sal      NUMBER(7,2) ,
  8         comm     NUMBER(7,2) ,
  9         deptno   NUMBER(2) ,
 10      CONSTRAINT emp  deptno  fk  FOREIGN KEY (deptno)
 11                    REFERENCES dept(deptno)  ) ;

Table created .
```

القيد على مستوى الجدول

القيد CHECK .

هذا القيد يتم إنشاؤه على مستوى العمود أو على مستوى الجدول ، ويستخدم هذا القيد عندما نريد تحديد مجال قيم معينة لعمود في الجدول ، فمثلاً إذا أردنا أن نحدد القيم المدخلة للعمود deptno داخل جدول الإدارات بحيث تكون هذه القيم محصورة بين (99 , 10) فإننا نطبق هذا القيد ، كما هو واضح من المثال التالي :

مثال (٦) : إنشاء القيد (CHECK) على العمود DEPTNO في جدول الإدارات وتطبيقه على مستوى الجدول .

```
SQL > CREATE TABLE dept (
  2                    deptno NUMBER(2) ,
  3                    dname VARCHAR2(14) ,
  4                     loc      VARCHAR2(13) ,
  5    CONSTRAINT  dept_deptno_ck  CHECK(deptno BETWEEN 10 AND 99)  ) ;

Table created .
```

القيد على مستوى الجدول

في المثال السابق تم إنشاء جدول الإدارات (dept) كما تم إنشاء قيد (check) على العمود deptno بشرط أن تكون القيم المدخلة لهذا العمود محصورة بين (99, 10) .

إضافة قيود على الجداول Adding Constrait.

يتم إضافة القيود على الجدول بعد إنشائه باستخدام الأمر Alter Table كما في الصيغة العامة التالية :

```
SQL > ALTER TABLE   أسم الجدول
      ADD  CONSTRAINT  أسم القيد      نوع القيد  ;
```

والأمثلة التالية توضح كيفية إضافة القيود على الجداول التي تم إنشاؤها سابقاً .

مثال (٧) : إضافة القيد (FOREIGN KEY) على العمود MGR في جدول الموظفين .

```
SQL > ALTER TABLE  emp
  2     ADD  CONSTRAINT  emp  mgr  fk
  3                  FOREIGN KEY(mgr)  REFERENCES  emp(empno) ;

Table altered .
```

في المثال السابق تم إضافة القيد (foreign key) إلى العمود (mgr) وذلك لربط رقم المدير لكل موظف برقم الموظف نفسه ونوع الربط هنا (selfjoin) ، راجع الفصل السادس .

مثال (٨) : إضافة القيد (PRIMARY KEY) على العمود DEPTNO في جدول الإدارات .

```
SQL > ALTER TABLE  DEPT
  2              ADD CONSTRAINT  dept_deptno_pk  PRIMARY KEY(deptno) ;

Table altered .
```

في المثال السابق تم إضافة القيد (primary key) إلى العمود (deptno) في جدول الإدارات وذلك في حالة عدم إنشائه أثناء إنشاء الجدول .

إزالة القيود من الجداول DROP.

يتم إزالة القيود من الجدول عن طريق استخدامنا لأمر Alter Table ، كما في المثال التالي :

مثال (٩) : إزالة القيد المسمى بـ (emp_mgr_fk) من جدول الموظفين .

```
SQL > ALTER TABLE  emp
  2     DROP  CONSTRAINT   emp_mgr_fk ;

Table altered .
```

كما هو واضح من المثال السابق فقد تم إزالة القيد (Foreign key) المسمى بـ (emp_mgr_fk) من الجدول (emp) .

وكما نعرف أن جدول الموظفين له علاقة بجدول الإدارات وذلك عن طريق وجود مفتاح أساسي (primary key) على العمود deptno داخل جدول الإدارات ووجود مفتاح (froeign key) على نفس العمود deptno داخل جدول الموظفين ، فماذا نفعل عندما نريد إلغاء أو إزالة قيد المفتاح الأساسي من جدول الإدارات وفي نفس الوقت إزالة العلاقة بينه وبين جدول الموظفين أي نزيل المفتاح (foreign key) من جدول الإدارات ، في هذه الحالة نقوم بإصدار الأمر كما في المثال التالي :

مثال (١٠) : إزالة العلاقة بين جدولي الموظفين والإدارات عن طريق إزالة المفتاح الأساسي من جدول الإدارات وإزالة توابعه .

```
SQL > ALTER TABLE  dept
  2       DROP  primary key  CASCADE ;

Table altered .
```

في المثال السابق تم إزالة المفتاح الأساسي من جدول الإدارات وكل ما يتعلق به من مفتاح ربط في جدول الموظفين وذلك باستخدام اللفظ (CASCADE) ، فهذا اللفظ يقوم بإزالة كل ما يتعلق بالمفتاح الأساسي أي يقوم بإزالة (Foreign key) من جدول الموظفين .

استعراض القيود المطبقة على جدول معين :

كما عرفنا سابقاً أن أوراكل يقوم بإنشاء جداول لتسجيل التغيرات التي تتم في قاعدة البيانات وهذه الجداول تسمى Data Dictionary . ومن خلالها يمكن عرض القيود المطبقة على جدول معين كما هو واضح من المثال التالي :

مثال (١١) : عرض القيود المختلفة المطبقة على جدول الموظفين .

```
SQL > SELECT constraint_name , constraint_type
  2   FROM   user_constraints
  3   WHERE  table_name = 'EMP' ;
```

CONSTRAINT_NAME	C
SYS_C00674	C
SYS_C00675	C
EMP_EMPNO_PK	P
FK_DEPTNO	R

في المثال السابق عرض أسماء القيود وأنواعها المطبقة على جدول الموظفين ، وكما نلاحظ أن ناتج عمود أنواع القيود (Constraint_type) عبارة عن حرف واحد يدل على نوع القيد ومعنى هذه الحروف هي كما يلي :

- الحرف C يعني أن نوع القيد هو CHECK .
- الحرف P يعني أن نوع القيد هو Primary key .
- الحرف R يعني أن نوع القيد هو Foreign key .
- الحرف U يعني أن نوع القيد هو UNIQUE .

أما نوع القيد NOT NULL فيظهر مثل القيد CHECK .

والمثال التالي يوضح كيفية عرض القيود المطبقة على الأعمدة .

مثال (١٢) : عرض أسماء الأعمدة وأسماء القيوم المطبقة عليها .

```
SQL > SELECT  constraint_name , column_name
  2     FROM  user_cons_column
  3     WHERE   table_name = 'EMP '  ;
```