الفصل الخامس
لغة الاستعلامات البنيوية (المهيكلة) SQL

1. مقدمة :

تمكنك لغة (SQL) اختصاراً للكلمات (Structured Query Language) من إدارة قواعد البيانات بشكل كامل وإجراء جميع العمليات القياسية كإنشاء الجداول وتعبئتها بالبيانات، أو إجراء الإستعلامات عليها وكذلك الربط بين الجداول المختلفة، ولكن كيف يمكننا عمل ذلك بها؟

كما وضحنا سابقاً فإن قاعدة البيانات عبارة عن مكان أو مستودع كبير لتخزين البيانات المختلفة. تريحك قاعدة البيانات من عناء تخزين بياناتك في ملفات منفصلة وكما أنك تحتاج إلى إجراء عمليات البحث والتصنيف لهذه البيانات عن طريق خوارزميات البحث المعقدة فتسهل لك ذلك، حيث أنها تعطيك واجهة سهلة للتعامل مع البيانات، بحيث لا تحتاج إلى كل هذا وتكون البيانات في قاعدة البيانات مخزنة في عدة جداول Tables وكما نعلم فالجدول يتكون من صفوف Rows وأعمدة Columns هكذا:

الجدول السابق يتكون من ثلاثة صفوف وثلاثة أعمدة، وفي قواعد البيانات فإننا نسمي الصفوف بالسجلات Records ونسمي الأعمدة بالحقول Fields، فما هي الحقول وما هي السجلات ؟

يقوم الحقل الواحد في الجدول بتخزين معلومة معينة، فمثلاً عندما يكون لدينا قاعدة بيانات بأرقام الهواتف فإننا سنحتاج إلى حقلين (أو عامودين)، واحد للإسم والثاني لرقم الهاتف. أما السجلات (أو الصفوف) فيحتوي كل منها على مجموعة من الحقول. لاحظ أن السجل الواحد في مثالنا يحتوي على معلومتين مختلفتين هما الاسم ورقم الهاتف، لذلك يمكننا القول بأن قاعدة البيانات تتكون من الجداول والجداول تتكون من السجلات والسجلات تتكون من الحقول، وكل حقل يحتوي على معلومة واحدة. أي أن الحقل هو أصغر وحدة قاعدة البيانات.

2. عبارات SQL

تنقسم عبارات SQL إلى ثلاث فئات رئيسية وهي كالتالي:

2.1. لغة تعريف البيانات DDL

هي مجموعة من أوامر SQL نستطيع من خلالها إنشاء وتعريف الكائنات في قاعدة البيانات، وتقوم هذه الأوامر بإنشاء أو إسقاط أو تغيير كائن في قاعدة البيانات، وهي اختصاراً للكلمات (Data Definition Language). من أهم أوامرها Create, Alter, Drop, Rename, Truncate.

2.2. لغة معالجة البيانات DML

هي مجموعة من العبارات التي نستطيع من خلالها معالجة البيانات الموجودة في قاعدة البيانات، من إدراج البيانات وتحديثها أو تحديد أو حذف البيانات، وهي اختصاراً للكلمات (Data Manipulation Language)، ومن أهم أوامرها Insert, Update, Delete.

2.3. لغة التحكم والصلاحيات DCL

هي مجموعة من أوامر SQL نستطيع من خلالها منح أو سحب إمتيازات استخدام كائن في قاعدة البيانات، وهي اختصاراً للكلمات (Language Data Control)، ومن أهم أوامرها Grant, Revoke.

وإيجازاً للقول فإن الشكل التالي يوضح الفئات الرئيسية لعبارات SQL:

لغة معالجة البيانات	Insert Update Delete
لغة تعريف البيانات	Create Alter Drop Rename Truncate
لغة التحكم والصلاحيات	Grant Revoke
لغة معالجة البيانات **Transaction control**	Commit Rollback Savepoint

1. جملة Select

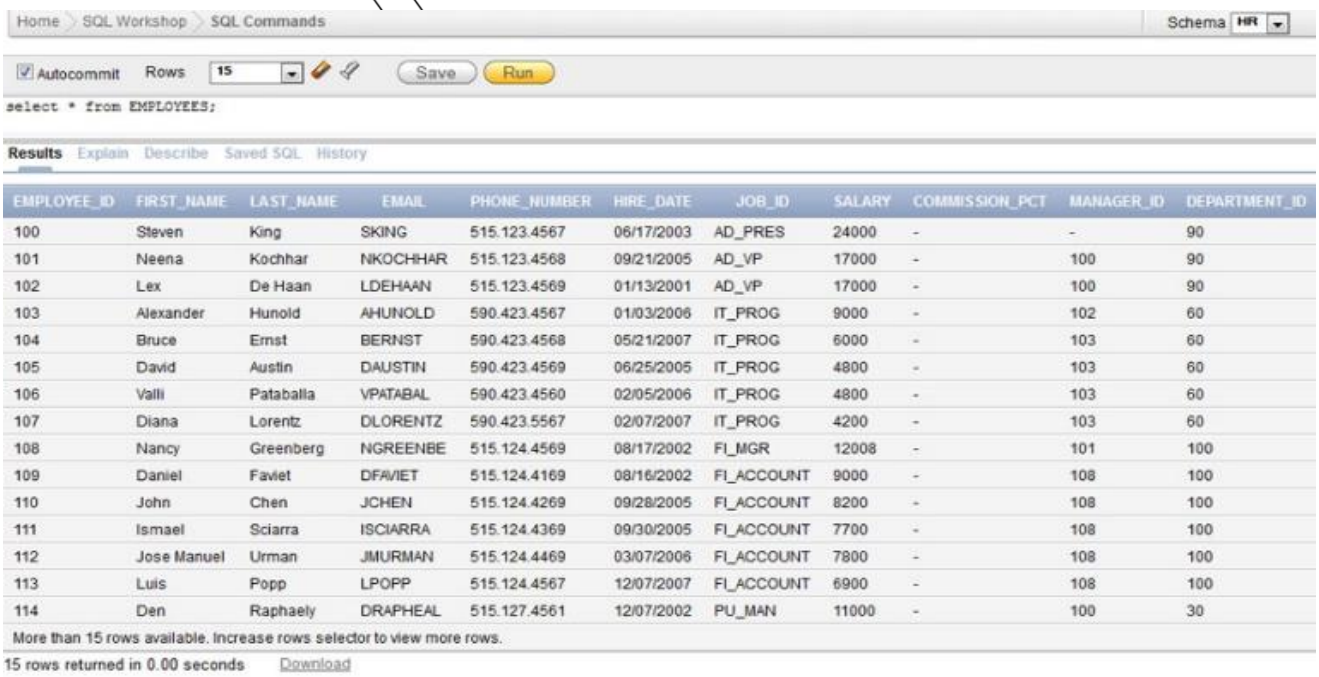
الأمر Select يمكنك من استعادة البيانات من داخل قاعدة البيانات، وتأخذ جملة Select الشكل التالي:

```
SELECT    [DISTINCT] {*, column [alias],...}
FROM      table;
```

1.1. اختيار كافة الأعمدة

تمثل عبارة * Select طريقة مختصرة لإخبار SQL بأننا نريد إظهار جميع أعمدة الجدول، فعلى سبيل المثال:

```
Select * from EMPLOYEES;
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

15 rows returned in 0.00 seconds Download

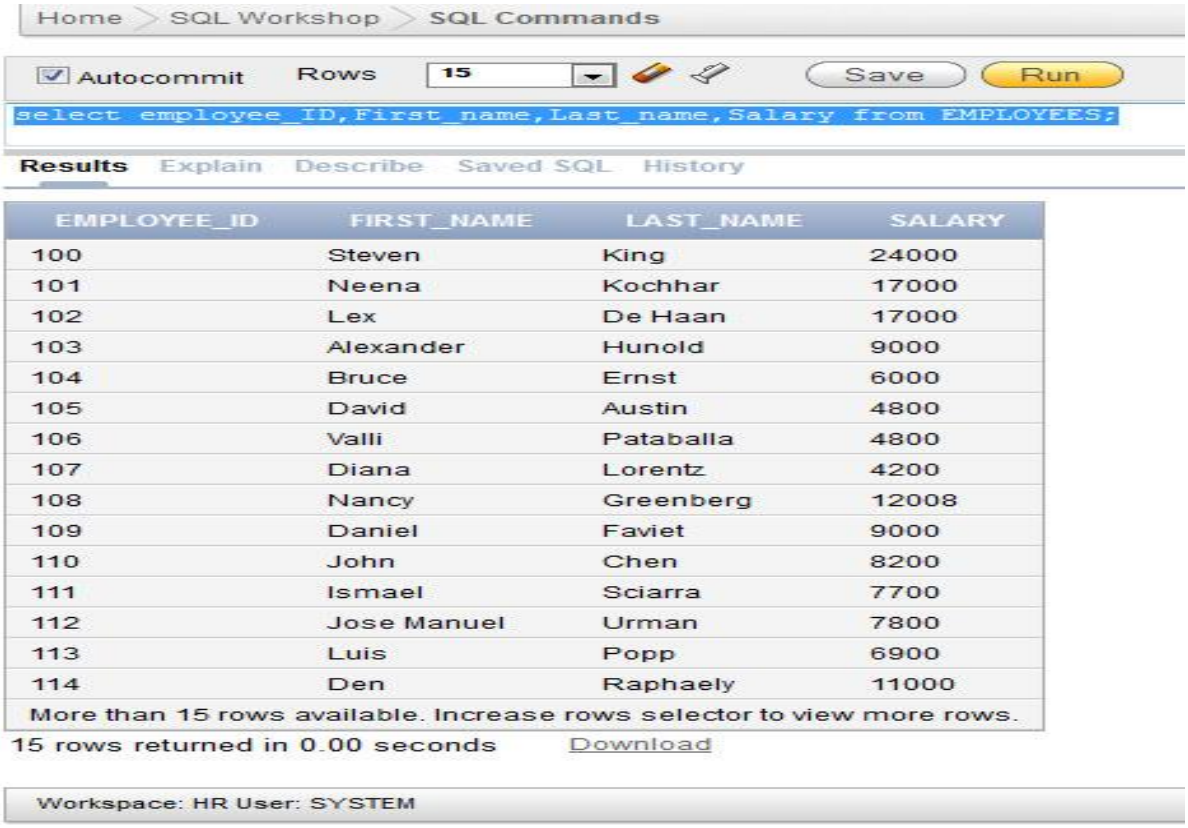
1) Select: تحدد ماهي الأعمدة المراد إظهارها.
2) From: تحدد المكان الذي نريد استعادة البيانات منه (جدول أو أكثر)
3) الفاصلة المنقوطة: تخبر SQL بأن العبارة كاملة وجاهزة للتنفيذ.
4) الأقواس [ ] تشير على أن ما بداخل هذه الأقواس اختياريا.

1.2. اختيار أعمدة محددة

يمكننا اختيار أعمدة محددة،وهو أمر في غاية السهولة حيث تقوم بكتابة أسماء الأعمدة بعد عبارة Select، مع وضع فاصلة بين أسماء الأعمدة، ثم تحدد الجدول بعد From.

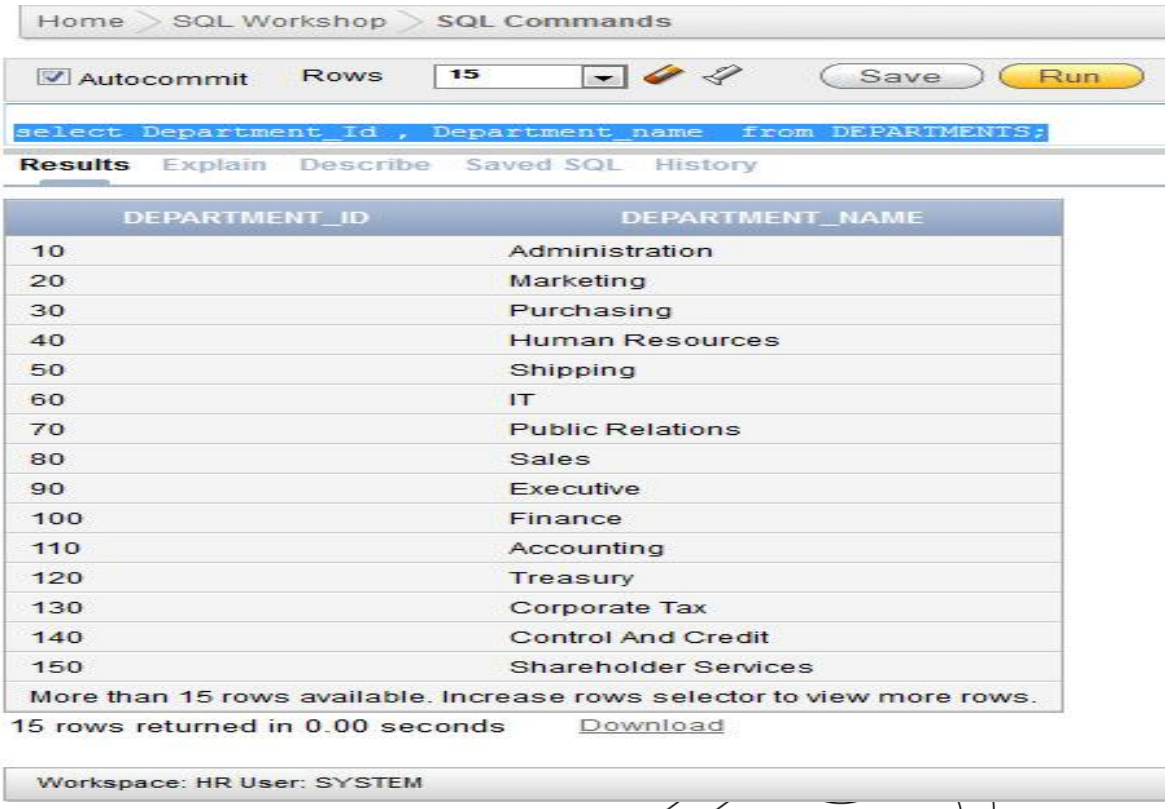
مثال:

```
Select employee_ID, First_name, Last_name, Salary from EMPLOYEES;
```



- ✓ لا تنس إنهاء جملة Select بالفاصلة المنقوطة لكي تخبر SQL بأن الجملة كاملة وجاهزة للتنفيذ.
- ✓ تظهر الأعمدة المطلوبة بدءاً من إليسار إلى إليمين بنفس ترتيب كتابتها في عبارة Select

```
Select Department_Id , Department_name  from DEPARTMENTS;
```



1.3. التعابير الحسابية (Arithmetic Expressions)

ضع التعبيرات الحسابية بعد عبارة Select كأي عمود تريد استعادة بياناته، حيث تستخدم التعبيرات الحسابية مع تلك الأعمدة التي نوع معطياتها عددي مثل الأرقام والتاريخ.

يجب عليك الانتباه لأسبقية المعاملات (Operator Precedence)

في التعبيرات الحسابية وفيما يلي بيان بأسبقية المعاملات في التعبيرات الحسابية

1. ينفذ أولاً الضرب أو القسمة أولا، ومن إليسار إلى إليمين
2. بعد ذلك ينفذ الجمع أو الطرح لاحقاً، ومن إليسار إلى ومن إليسار إلى اليمين.

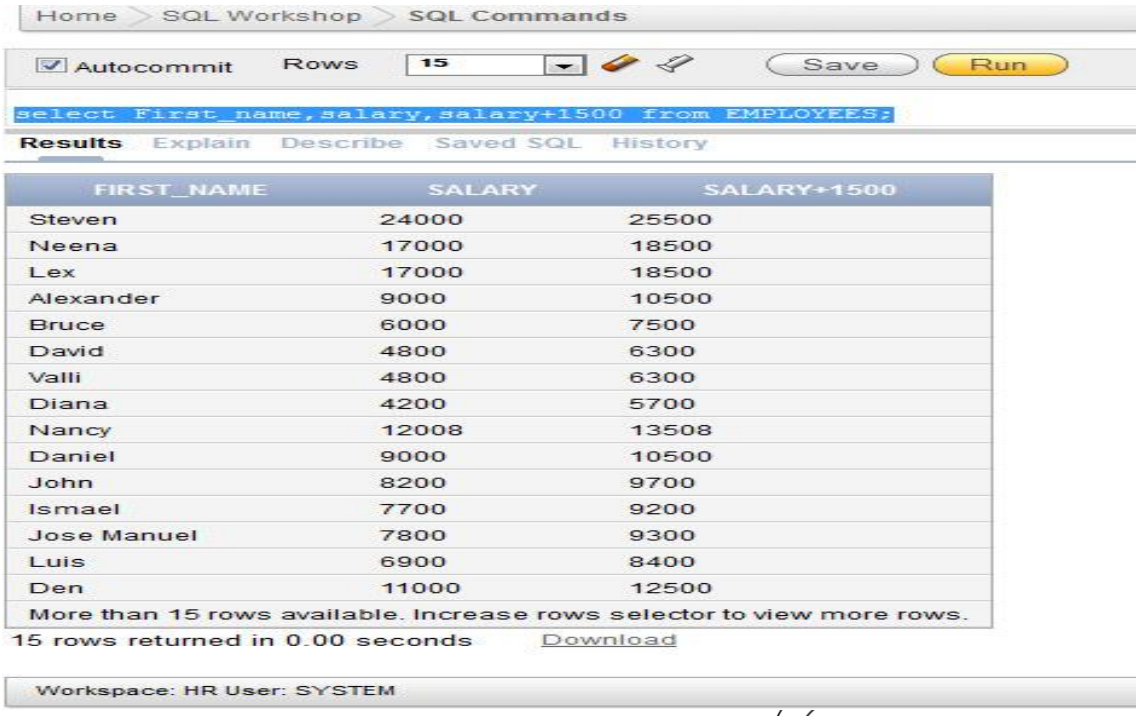
ملاحظة

عند استخدام التعبيرات الحسابية يمكنك تغيير تسلسل أسبقية المعاملات الحسابية باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعبيرات الحسابية.

```
Select First_name, salary, salary+1500 from EMPLOYEES;
```

1.3.1.أمثلة على التعابير الحسابية

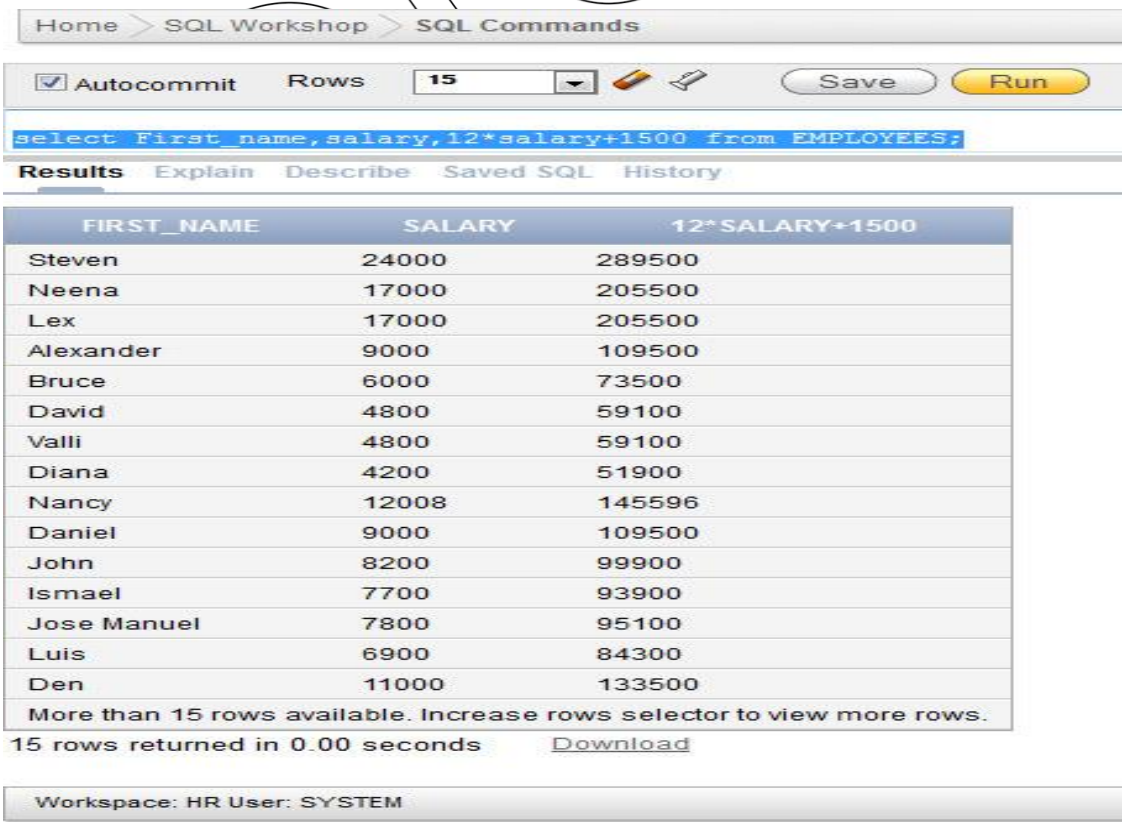
في هذا المثال نقوم باستخدام معامل الجمع لزيادة مرتبات جميع الموظفين بـ 1500 ، و من ثم نقوم بعرض الناتج في عمود جديد.




```
Select First_name, salary, 12*salary+1500 from EMPLOYEES;
```

✓ تتجاهل SQL المسافات Blank Spaces قبل وبعد المعاملات الحسابية.

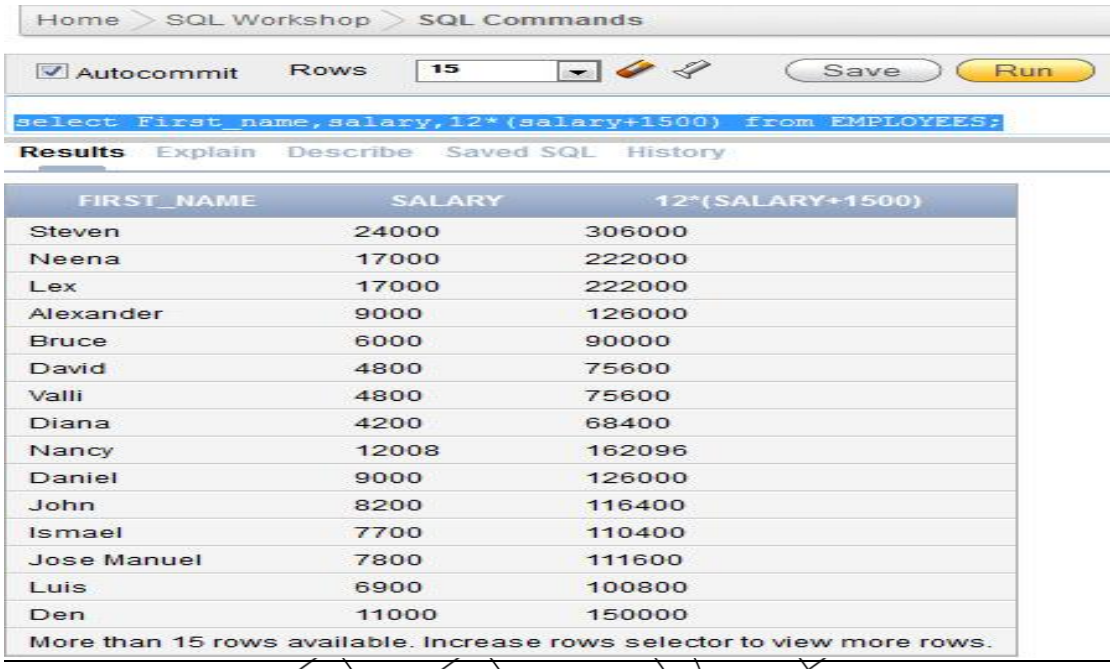
في هذا المثال سنقوم، طبقاً لأسبقية المعاملات، كما وضحنا سابقاً بحساب عملية الضرب أولاً وهي Salary *12، ثم إجراء عملية الجمع ثانياً وهي إضافة 1500.



✓ في المثال السابق كنا نريد زيادة مرتبات جميع الموظفين بـ 1500 ثم حساب المرتب السنوي بعد عملية الزيادة فكيف يمكننا ذلك؟

✓ كما تعلمنا سابقاً يمكنك تغيير تسلسل أسبقية المعاملات الحسابية باستخدام الأقواس (Parentheses)، حيث تأخذ الأقواس الأسبقية في التعابير الحسابية، ويتم ذلك على النحو التالي:

```
Select First_name, salary, 12*(salary+1500) from EMPLOYEES;
```



FIRST_NAME	SALARY	12*(SALARY+1500)
Steven	24000	306000
Neena	17000	222000
Lex	17000	222000
Alexander	9000	126000
Bruce	6000	90000
David	4800	75600
Valli	4800	75600
Diana	4200	68400
Nancy	12008	162096
Daniel	9000	126000
John	8200	116400
Ismael	7700	110400
Jose Manuel	7800	111600
Luis	6900	100800
Den	11000	150000

More than 15 rows available. Increase rows selector to view more rows.

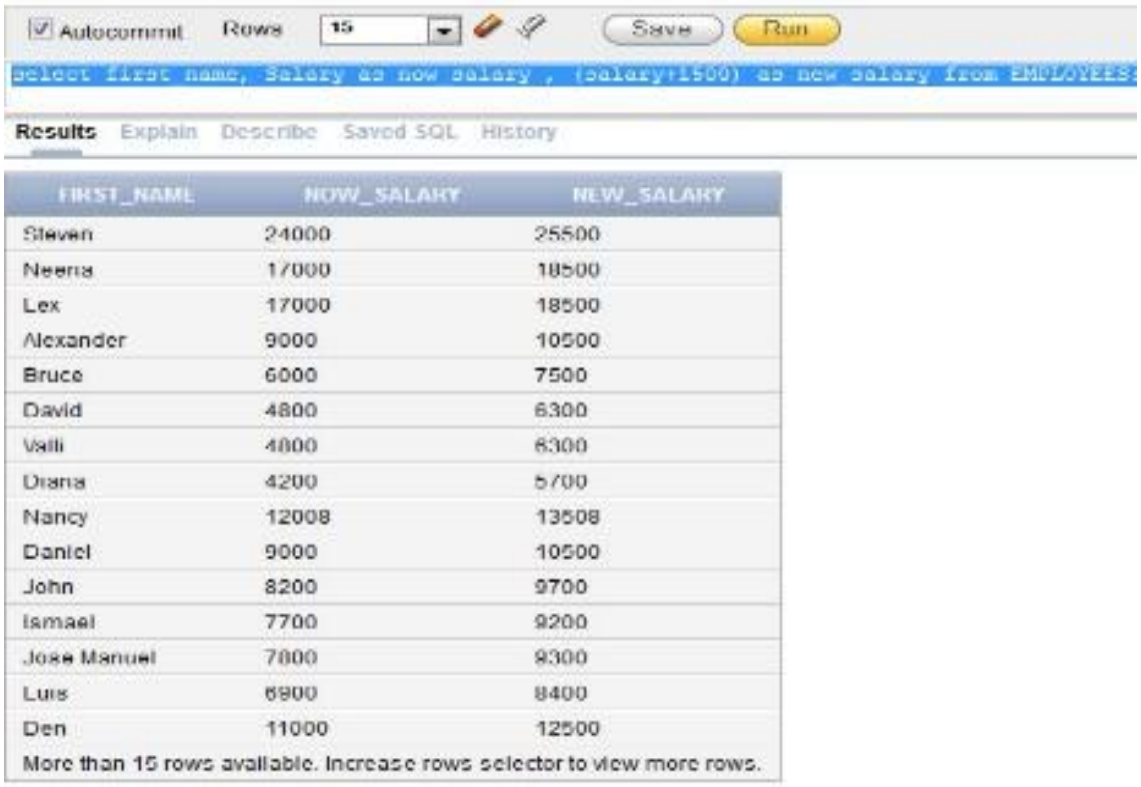
1.4. تعريف A Column Alias

يمكننا إعادة تسمية روؤس الأعمدة باستخدام A Column Alias، بمعنى إبدال الإسم الفعلي للحقل باسم اعتباري يتم تحديده بواسطة المستخدم ويكون ذلك مفيداً وخاصة عند التعامل مع التعبيرات الحسابية، وفيما يلي عدة اعتبارات لابد من مراعاتها عند إعادة تسمية روؤس الأعمدة

1) يتم كتابة Alias في جملة Select يلي اسم العمود أو التعبير الحسابي المراد إعادة تسميته مع ترك مسافة.
2) يتم استخدام As بين اسم العمود وال Alias، ويكون ذلك اختيارياً.
3) إذا احتوي Aliasعلى مسافات أو أحرف وعلامات خاصة مثل ($ or #)

من وضعة بين علامتي تنصيص (" ")Double Quotation.

```
Select first_name, Salary as now_salary , (salary+1500) as new_salary from EMPLOYEES;
```



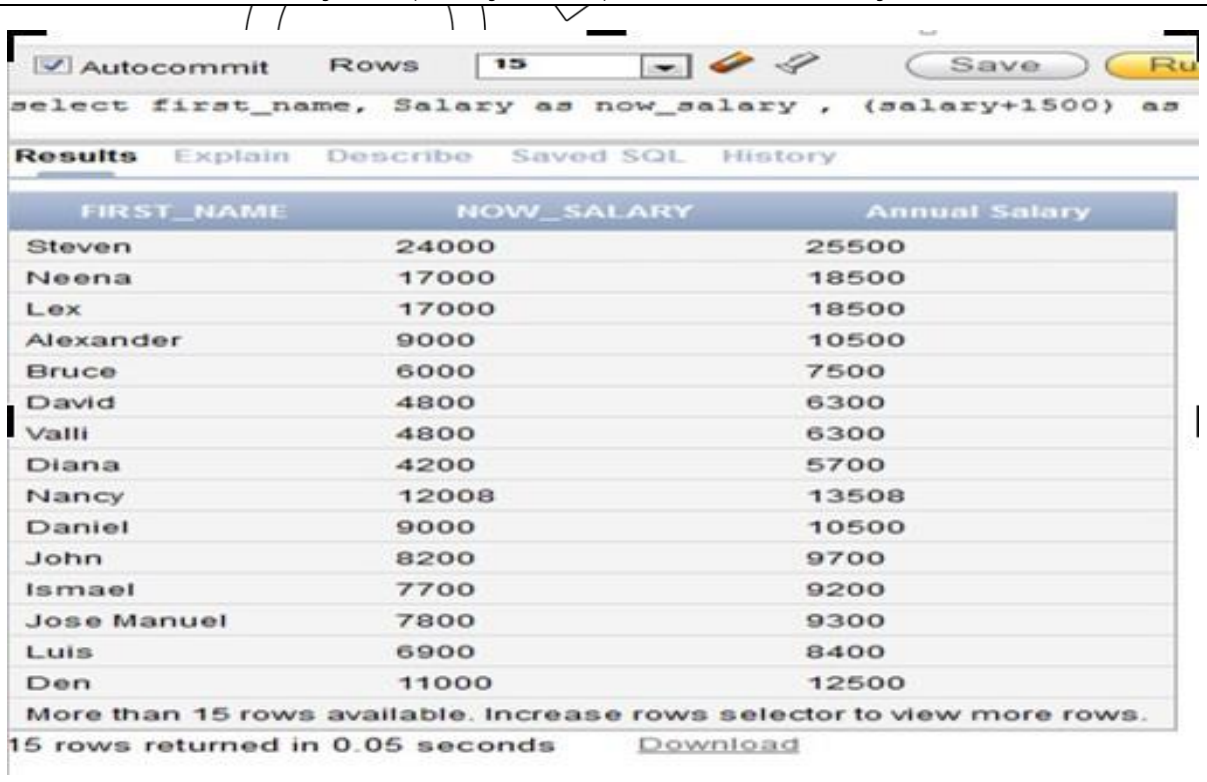
FIRST_NAME	NOW_SALARY	NEW_SALARY
Steven	24000	25500
Neena	17000	18500
Lex	17000	18500
Alexander	9000	10500
Bruce	6000	7500
David	4800	6300
Valli	4800	6300
Diana	4200	5700
Nancy	12008	13508
Daniel	9000	10500
John	8200	9700
Ismael	7700	9200
Jose Manuel	7800	9300
Luis	6900	8400
Den	11000	12500

More than 15 rows available. Increase rows selector to view more rows.

- ✓ في المثال السابق تم إعادة تسمية عمود Salary في جدول Employees بـ Now_Salary.
- م وضع كلمة As قبل Alias Name، وكما تعلمنا أن ذلك اختيارياً.
- ✓ يتم ظهور Alias Name بحالة أحرف كبيرة Uppercase.

مثال2:

```
Select First_name, salary,12*(salary+1500) as "Annual Salary" from EMPLOYEES;
```



FIRST_NAME	NOW_SALARY	Annual Salary
Steven	24000	25500
Neena	17000	18500
Lex	17000	18500
Alexander	9000	10500
Bruce	6000	7500
David	4800	6300
Valli	4800	6300
Diana	4200	5700
Nancy	12008	13508
Daniel	9000	10500
John	8200	9700
Ismael	7700	9200
Jose Manuel	7800	9300
Luis	6900	8400
Den	11000	12500

More than 15 rows available. Increase rows selector to view more rows.

15 rows returned in 0.05 seconds Download

في هذا المثال ايضا تم إعادة تسمية التعبير الحسابي Salary +1500 بـ Annual Salary، و كما نلاحظ تم وضع "Annual Salary" بين علامتي تنصيص لأنه يحتوي مسافات Spaces.

1.5. جملة الشرط Where Clause

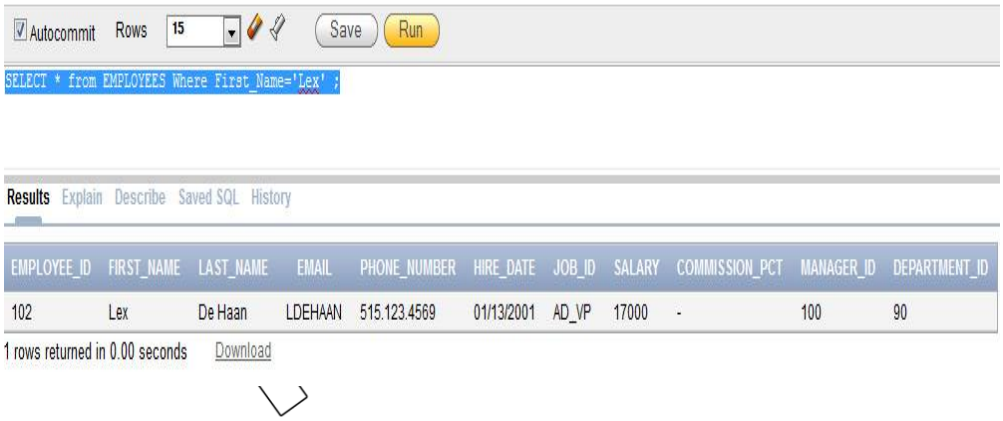
كان اختيارنا سابقاً للأعمدة يجري على كافة الصفوف التي يحتويها الجدول، أردنا إظهار صفوفاً تحتوي على قيماً معينة، كإظهار الموظفين الذين تتجاوز مرتباتهم 3000 ليرة أو الذين يقطنون في مدينة ما، لفعل مثل ذلك سنحتاج إلى

```
SELECT      [DISTINCT] {*, column [alias], ...}
 FROM       table
[WHERE      condition(s)];
```

وضع كلمة Where ضمن عبارة Select، يأخذ الإختيار الشرطي الشكل التالي:

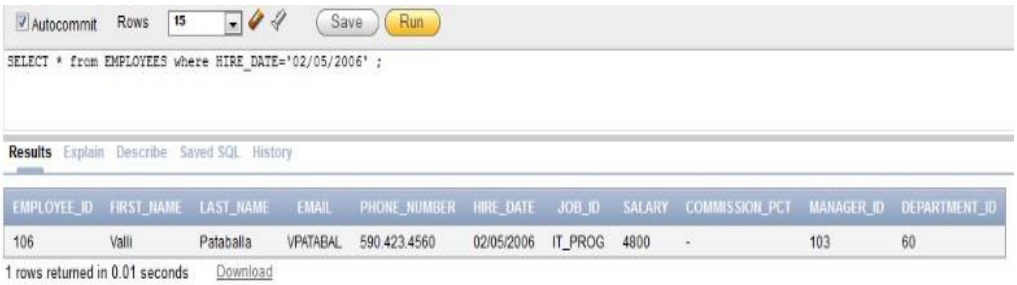
```
SELECT * from EMPLOYEES Where First_Name='Lex' ;
```

- ✓ تأتي جملة Where Clause بعد جملة From Clause.
- ✓ يلي Where شرط أوعدة شروط
- ✓ توجه عبارة Where الأمر إلى نظام ادارة قواعد البيانات أوراكل Oracle مثلا بالبحث في قاعدة البيانات واستعادة الصفوف التي تتوافق مع شرط البحث.




```
SELECT * from EMPLOYEES where HIRE_DATE='02/05/2006' ;
```

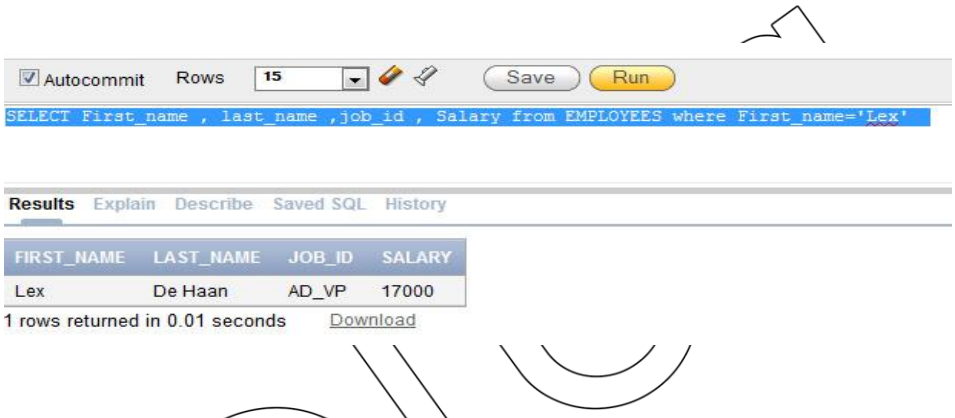
في هذا المثال نريد استرجاع جميع الأعمدة بجدول Employees، ولكن بشرط أن يكون اسم الموظف 'Lex'.



في هذا المثال نريد استرجاع جميع الأعمدة بجدول Employees، ولكن بشرط أن يكون تاريخ تعيين الموظف هو'02/05/2006'

- ✓ الأحرف النصية Character Strings، والتاريخ في جملة Where لابد من وضعها بين علامتى تنصيص Single Quotation (' ').

```
SELECT First_name , last_name, job_id , Salary from EMPLOYEES where First_name='Lex'
```

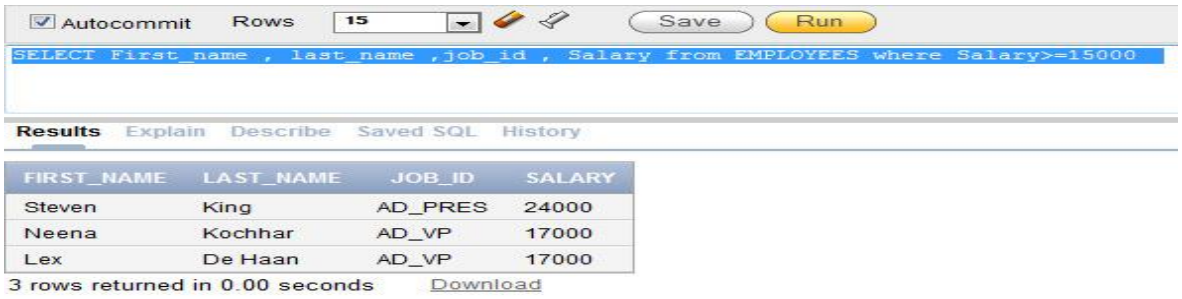


1.6. عوامل المقارنة (Comparison Operators)

تمنحنا SQL ستة عوامل للمقارنة نستطيع استخدامها مع Where، كما هو مبين في الجدول التالي:

Meaning المعنى	Operator المعامل
يســـاوي	=
أكبر من أو أصغر من	< or >
أكبر من أو يساوي	>=
أصغر من أو يساوي	<=
لا يساوي	<> or ! =

```
SELECT First_name , last_name, job_id, Salary from EMPLOYEES where Salary>=15000
```



في هذا المثال استخدمنا أحد عوامل المقارنة في جملة Where، حيث تم استعادة , First_Name Last_Name ,Job_Id , Salary عندما يكون Salary أكبرمن أو يساوي 15000 ليرة .

مثال أخر :

```
SELECT First_name , last_name, job_id , Salary from EMPLOYEES where Salary<>15000
```

1.7. عوامل المقارنة الأخرى (Other Comparison Operator)

تتضمن SQL أربعة معاملات أخرى بالإضافة للمعاملات التي تم الإشارة إليها في الفقرة السابقة و التي يمكن استخدامها مع جملة Where، كما هو مبين في الجدول التالي:

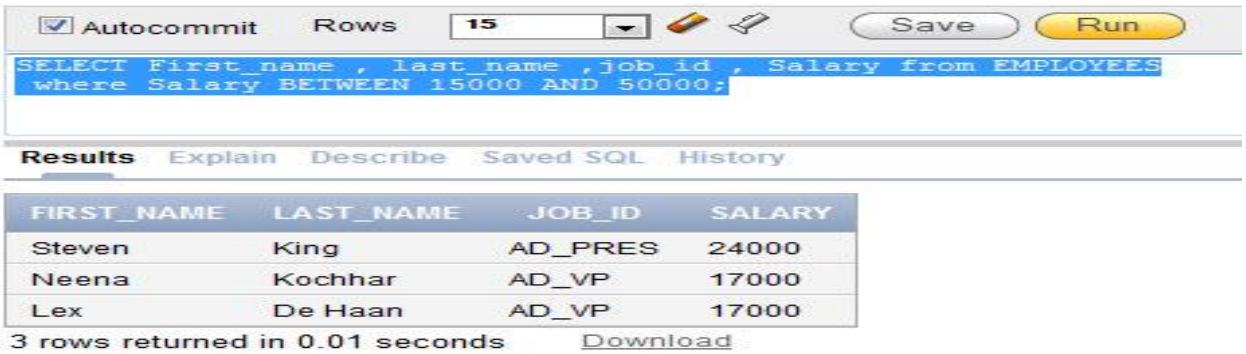
Operator المعامل	Meaning المعنى
Between..And…	بين قيمتين
In(List)	تتيح لك اختيار أي قيمة من List
Like	يشابه أو يماثل
Is Null	فارغاً

1.8. المعامل Between

يمكنك إجراء عمليات البحث داخل نطاق معين من القيم وذلك باستخدام معامل Between مع جملة Where، فعلي سبيل المثل قد ترغب في استعادة بيانات جميع الموظفين الذين يتقاضون راتب مابين 15000، 50000 يمكنك القيام بهذا وذلك باستخدام العبارة التالية:

```
SELECT First_name , last_name ,job_id , Salary from EMPLOYEES where Salary BETWEEN 15000 AND 50000;
```

في هذا المثال تم استخدام معامل Between مع جملة Where لإستعادة السجلات التي تقع ضمن نطاق معين من القيم.



1.9. المعامل In

يمكننا معامل In من القيام بعمليات البحث عن بنود داخل إحدى القوائم، حيث يمكنك استرجاع السجلات التي قيم عمود فيها (محدد في جملة Where) تساوي إحدى القيم المحددة ضمن مجموعة. على سبيل المثال لإظهار كافة السجلات الخاصة بالموظفين ذوي الأرقام 100 أو 110 أو 120 نقوم بكتابة العبارة التالية:

```
SELECT * from EMPLOYEES where EMPLOYEE_ID In (100,110,120);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
120	Matthew	Weiss	MWEISS	650.123.1234	07/18/2004	ST_MAN	8000	-	100	50

3 rows returned in 0.04 seconds Download

في هذا المثال تم استرجاع السجلات التي فيها العمود Employee_Id يساوي إما 100 أو 110 أو 120، فنجد أن In تكافيء سلسلة مقارنات مربوطة بالمعامل Or، حيث يمكننا الحصول على النتيجة نفسها في المثال السابق باستخدام المعامل Or على النحو التالي:

```
SELECT * from EMPLOYEES where EMPLOYEE_ID=100 or EMPLOYEE_ID=110 or EMPLOYEE_ID=120
```

في هذا المثال سيتم استعادة كافة السجلات الخاصة بالموظفين الموجودين في القائمة التي تلي معامل In، حيث يتم استعادة جميع السجلات من جدول Employees عندما يكون First_Name يساوي إما'David' أو 'Nancy' أو 'Lex' أو 'John'.

```
SELECT * from EMPLOYEES where First_name In ('Lex','David','Nancy','John')
```

فيما يلي عدة اعتبارات لابد من مراعاتها عند إعادة استخدام معامل In مع جملة Where:

- ✓ وضع قيم المجموعة بين قوسين.
- ✓ فصل قيم المجموعة عن بعضها بفاصلة.
- ✓ يمكن وضع القيم بأي ترتيب.
- ✓ عندما تكون قيم المجموعة حرفية أو تاريخ لابد من وضعها بين علامتي تنصيص(' ').
- ✓ يمكن استخدام Not In لعكس الاختيار كما في العبارة التالية:

```
SELECT * from EMPLOYEES where First_name not In ('Lex','David','Nancy','John')
```

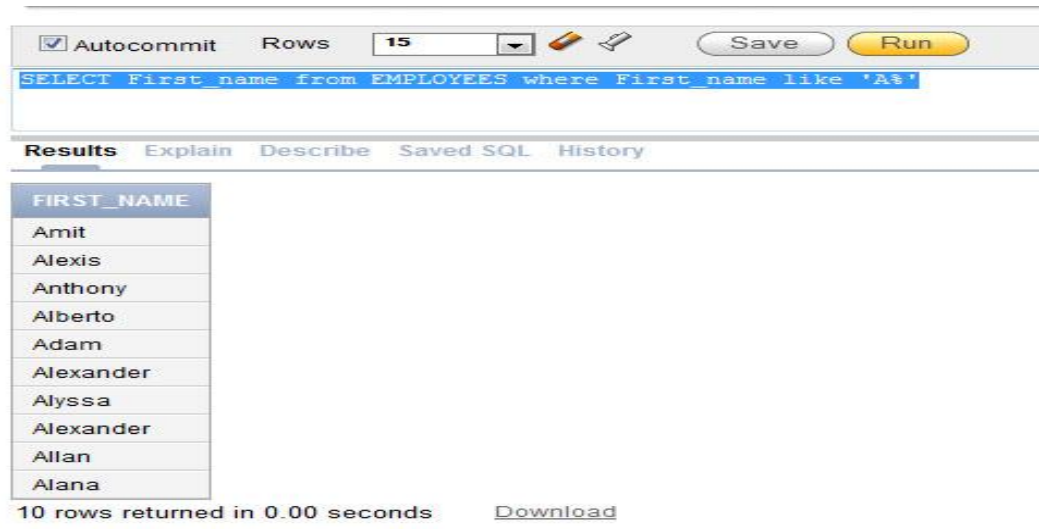
في هذه العبارة تم استعادة جميع السجلات من جدول Employees عندما يكون First_Name لا يساوي 'David' أو 'Nancy' أو 'Lex' أو 'John'.

1.10. المعامل Like

يمكنك استخدام المعامل Like مع جملة Where لاستعادة واسترجاع السجلات التي قيم عمود فيها تشابه سلسلة أحرف نبحث عنها. ليس هذا فحسب بل يمكننا من استرجاع السجلات التي تكون قيم عمود فيها مطابقة جزئياً لمجموعة أحرف نبحث عنها.

لإسترجاع جميع الأسماء التي تبدأ بحرف (A) يتم كتابة العبارة التالية:

```
SELECT First_name from EMPLOYEES where First_name like 'A%'
```



للبحث عن الأسماء التي تشتمل على أحرف Ma نكتب العبارة التالية

```
SELECT * from EMPLOYEES
        where First_name like '%ma%'
```

لاسترجاع بيانات للموظفين الذي تم تعينهم خلال عام 2005 نكتب العبارة:

لإسترجاع جميع الأسماء

```
SELECT * from EMPLOYEES where HIRE_DATE like '%2005'
```

التي يكون الحرف (A) فيها يأخذ الترتيب الثاني يتم كتابة العبارة التالية:

```
SELECT First_name from EMPLOYEES where First_name like '_a%'
```

- ✓ تستخدم Like مع الأعمدة التي يكون نوع بياناتها حرفي.
- ✓ % اختصار لأي تتابع من سلسلة أحرف غير معينة مؤلفة من أي قيمة وبأي طول
- ✓ الرمز _ اختصار لأي حرف وحيد.

Autocommit Rows 15 Save Run

```
SELECT First_name from EMPLOYEES where First_name like '_a%'
```

Results Explain Describe Saved SQL History

FIRST_NAME
David
Sarah
David
Laura
Harrison
Nanette
Karen
Daniel
Pat
Tayler
Nancy
Danielle
Vance
Payam
Janette

1.11. المعامل Is Null

قبل استخدام معامل Is Null مع Where لابد من الإجابة على ماهي القيمة Null ؟ تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات، حيث لا تعني القيمة Null صفر أو مسافةBlank Space.

لإسترجاع جميع بيانات الموظفين الذين لا يتقاضون حوافز Commission يتم كتابة الكود التالي:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is null
```

Autocommit Rows 15 Save Run

```
SELECT * from EMPLOYEES where COMMISSION_PCT is null
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

لاسترجاع جميع بيانات الموظفين الذين يتقاضون حوافز Commission يتم كتابة الكود التالي:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is not null
```

1.12. استخدام العوامل المنطقية (Logical Operators)

بالإضافة للمعاملات التي تم الإشارة إليها سابقاً، تتضمن SQL ثلاثة معاملات منطقية يمكن استخدامها مع جملة Where، كما هو مبين بالجدول التالي:

المعامل Operator	المعنى Meaning
And	و
Or	أو
Not	النفي

1.12.1. استخدام معامل and

يستخدم المعامل And مع Where لإستعادة السجلات التي تحقق كافة الشروط المحددة في جملة Where، حيث يعني And أنه لاستعادة السجلات لابد من تحقق كافة الشروط.

لإسترجاع جميع بيانات الموظفين بالإدارة It_Prog الذين يتقاضون مرتباً أكبر من 6000 نكتب العبارة التالية:

```
SELECT * from EMPLOYEES where Salary > 6000 and JOB_ID='IT_PROG'
```

لإسترجاع جميع بيانات قسم 'It_Prog' الذين يتقاضون مرتباً اقل من 6000 نكتب العبارة التالية:

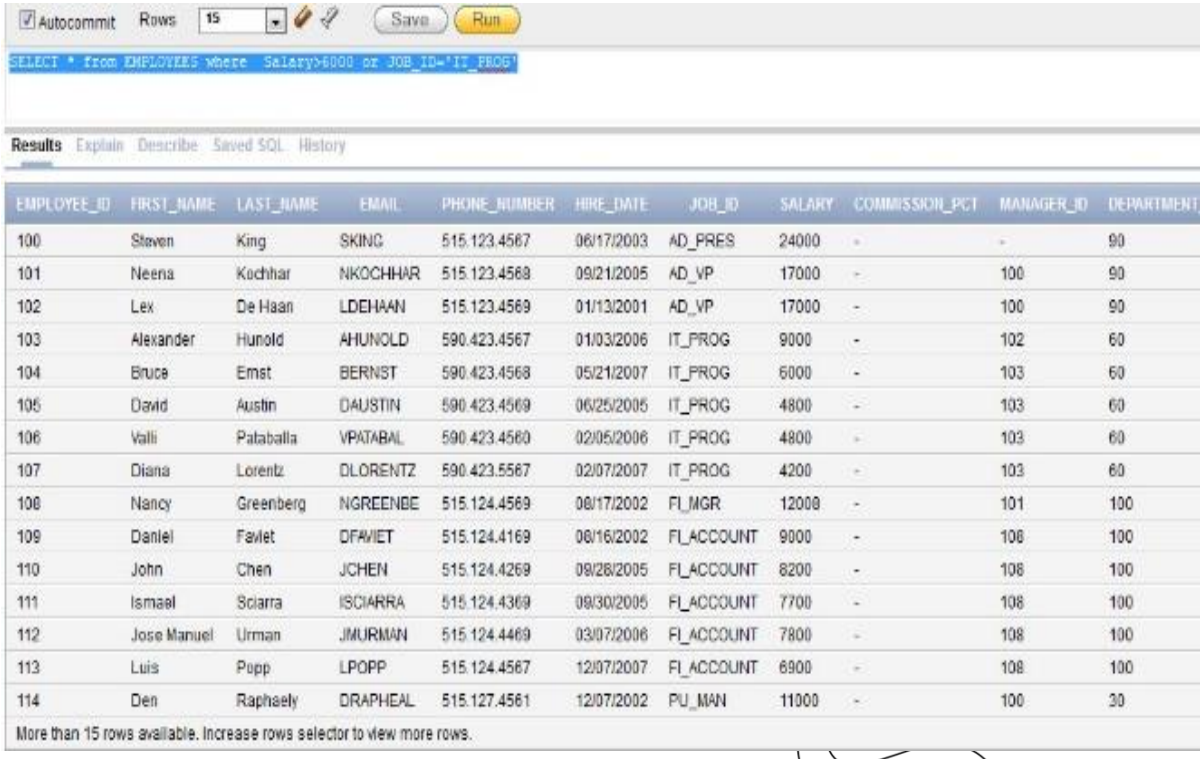
```
SELECT * from EMPLOYEES where  Salary <6000 and JOB_ID='IT_PROG'
```

1.12.2. استخدام معامل or

نستخدم معامل or مع Where لإستعادة السجلات التي تحقق شرطا واحداً من عدة شروط، حيث يعني المعامل or أن استعادة البيانات تتم إذا تحقق أي شرط من الشروط.

لإسترجاع بيانات الموظفين الذين يتقاضون مرتباً أكبر من 6000 ليرة أو يعملون بالوظيفة 'It_Prog'، نكتب العبارة التالية:

```
SELECT * from EMPLOYEES where  Salary>6000 or JOB_ID='IT_PROG'
```



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	09/21/2005	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/2006	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	05/21/2007	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	06/25/2005	IT_PROG	4800	-	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/2006	IT_PROG	4800	-	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/2007	IT_PROG	4200	-	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
110	John	Chen	JCHEN	515.124.4269	09/28/2005	FI_ACCOUNT	8200	-	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	09/30/2005	FI_ACCOUNT	7700	-	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/2006	FI_ACCOUNT	7800	-	108	100
113	Luis	Popp	LPOPP	515.124.4567	12/07/2007	FI_ACCOUNT	6900	-	108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30

More than 15 rows available. Increase rows selector to view more rows.

✓ تم عرض بيانات الموظفين الذين يتقاضون مرتباً أكبر من 6000 أو يعملون في الوظيفة It_Prog.

1.12.3. استخدام معامل Not

استخدام المعامل Not يمكننا من استثناء سجلات معينة من اختيارنا، حيث يمكننا من استرجاع السجلات المعاكسة لاختيارنا، فعلي سبيل المثال لاختيار كافة السجلات من جدول Employees باستثناء التي يكون الـ Commission_Pct لها قيمته Null نكتب العبارة التالية:

```
SELECT * from EMPLOYEES where COMMISSION_PCT is not null
```

لإسترجاع بيانات الموظفين باستثناء الوظيفة Fi_Account, Pu_Man, Ad_Pres نكتب العبارة التالية:

```
SELECT * FROM   EMPLOYEES
WHERE JOB_ID NOT IN (‘AD_PRES’, ‘FI_ACCOUNT’, ‘PU_MAN’);
```

✓ يمكن استخدم معامل Not مع المعاملات الأخرى مثل

(Between, Like, Null)

وإليك التلميح التالي:

```
….. Where SALARY NOT BETWEEN 3000 AND 4000
….. Where FIRST_NAME NOT LIKE '%T%'
….. Where JOB_ID NOT IN('analyst','accountant')
….. Where COMMISSION_PCT IS NOT NULL
```

1.13. قواعد أسبقية المعاملات

Rules of Precedence

الجدول التالي يوضح قواعد أسبقية المعاملات.

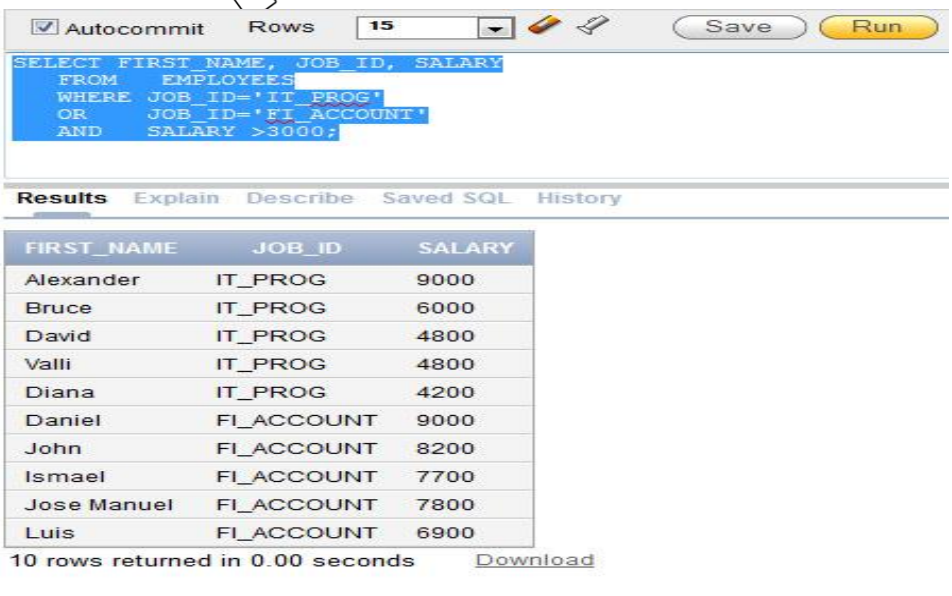
Order الترتيب Evaluated	المعامل Operator
1	جميع عوامل المقارنة
2	Not
3	and
4	Or

✓ كما تعلمنا سابقاً يمكننا تغيير تسلسل أسبقية المعاملات باستخدام الأقواس(Parentheses)، حيث تأخذ الأقواس الأسبقية في التعابير الحسابية

```
SELECT FIRST_NAME, JOB_ID, SALARY FROM   EMPLOYEES
  WHERE JOB_ID='IT_PROG'
  OR    JOB_ID='FI_ACCOUNT'
  AND   SALARY >3000;
```

✓ الشرط الأول هو JOB_ID تكون'IT_PROG' و SALARY أكبر من 3000

✓ الشرط الثاني هو JOB_ID تكون 'FI_ACCOUNT'



1.14. ترتيب النتائج باستخدام Order by

يمكن ترتيب السجلات باستخدام Order By وفقاً لقيم العمود الذي نحدده ترتيباً تصاعدياً أو تنازلياً، ويمكن الترتيب بناءً على قيم عمود أو عدة أعمدة، وعند كتابة جملة Order By يجب إتباع الآتي:

- ✓ Order By تتبع جملة From{Table Name} مباشرةً في حالة عدم وجود Where {Condition(S)}
- ✓ Order By تتبع جملة Where {Condition(S)} مباشرةً.
- ✓ يوجد لدينا خياران عند استخدام Order By:

1) Order By {Column Name} Asc: يسمح هذا الخيار بالترتيب التصاعدي

2) Order By {Column Name} Desc: يسمح هذا الخيار بالترتيب التنازلي.

- ✓ يمكن الترتيب على أساس عدة أعمدة، ويجب الفصل بين أسماء الأعمدة بفاصلة (,).
- ✓ يوجد لدينا طريقتان لكتابة جملة Order By و هما:
  1. عن طريق اسم العمود
  2. عن طريق رقم العمود النسبي (موقعه في جملة Select)

- الترتيب التصاعدي باستخدام Order By

لإستعادة بيانات جميع الموظفين مرتبة على أساس تاريخ التعيين، نكتب العبارة التالية:

```
Select * from EMPLOYEES ORDER BY HIRE_DATE ASC
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
102	Lex	De Haan	LDEHAAN	515.123.4569	01/13/2001	AD_VP	17000	-	100	90
206	William	Gietz	WGIETZ	515.123.8181	06/07/2002	AC_ACCOUNT	8300	-	205	110
204	Hermann	Baer	HBAER	515.123.8888	06/07/2002	PR_REP	10000	-	101	70
203	Susan	Mavris	SMAVRIS	515.123.7777	06/07/2002	HR_REP	6500	-	101	40
205	Shelley	Higgins	SHIGGINS	515.123.8080	06/07/2002	AC_MGR	12008	-	101	110
109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/2002	FI_ACCOUNT	9000	-	108	100
108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/2002	FI_MGR	12008	-	101	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/2002	PU_MAN	11000	-	100	30
122	Payam	Kaufling	PKAUFLIN	650.123.3234	05/01/2003	ST_MAN	7900	-	100	50
115	Alexander	Khoo	AKHOO	515.127.4562	05/18/2003	PU_CLERK	3100	-	114	30
100	Steven	King	SKING	515.123.4567	06/17/2003	AD_PRES	24000	-	-	90
137	Renske	Ladwig	RLADWIG	650.121.1234	07/14/2003	ST_CLERK	3600	-	123	50
200	Jennifer	Whalen	JWHALEN	515.123.4444	09/17/2003	AD_ASST	4400	-	101	10
141	Trenna	Rajs	TRAJS	650.121.8009	10/17/2003	ST_CLERK	3500	-	124	50
184	Nandita	Sarchand	NSARCHAN	650.509.1876	01/27/2004	SH_CLERK	4200	-	121	50

More than 15 rows available. Increase rows selector to view more rows.

- الترتيب التنازلي باستخدام Order By

لإستعادة بيانات جميع الموظفين مرتبة تنازلياً على أساس تاريخ التعيين، نكتب العبارة التالية:

```
Select * from EMPLOYEES Order By HIRE_DATE DESC
```

ملاحظات:

- ✓ الترتيب الإفتراضي في جملة Order By هو الترتيب التصاعدي مالم نصرح عكس ذلك بالكلمة Desc.
- ✓ يتم عرض القيم Null مؤخراً عند الترتيب التصاعدي، والعكس يتم ظهور القيم Null أولاً عند الترتيب التنازلي.
- ✓ يمكنك إجراء الترتيب باستخدام Order By على أساس قيم أعمدة ليست بجملة Select، كما في المثال التالي:

```
Select FIRST_NAME ,SALARY  from EMPLOYEES ORDER BY HIRE_DATE
,SALARY DESC
```

FIRST_NAME	SALARY
Lex	17000
Shelley	12008
Hermann	10000
William	8300
Susan	6500
Daniel	9000
Nancy	12008
Den	11000
Payam	7900
Alexander	3100
Steven	24000
Renske	3600
Jennifer	4400
Trenna	3500
Nandita	4200
More than 15 rows available. Increase rows selector to view more rows.	

- ✓ يمكننا إجراء الترتيب باستخدام Order By على أساس قيم أكثر من عمود.

- الترتيب باستخدام A Column Alias

يمكننا إجراء الترتيب في جملة Order By باستخدام A Column Alias كما في المثال التالي:

```
Select FIRST_NAME ,SALARY*12 as total   from EMPLOYEES Order BY total DESC
```

FIRST_NAME	TOTAL
Steven	288000
Neena	204000
Lex	204000
John	168000
Karen	162000
Michael	156000
Nancy	144096
Shelley	144096
Alberto	144000
Lisa	138000
Den	132000
Gerald	132000
Ellen	132000
Eleni	126000
Clara	126000
More than 15 rows available. Increase rows selector to view more rows.	

1.15. القيمة Null وتأثيراتها

تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات بمعنى وجود قيم فارغة داخل الحقول ومن أشهر عيوب القيمة Null صعوبة إجراء أي عمليات حسابية على سجلات تحتوى بعضها على قيم فارغة.

استخدام القيمة Null

1) تستخدم القيمة Null للإشارة إلى عمود لا يحتوي على بيانات، ولا تعني القيمة Null الصفر أو المسافة الفارغة Blank Space.

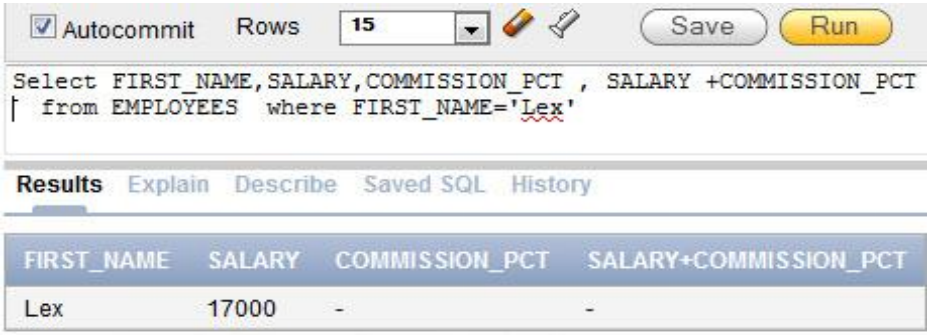
2) عند استخدام القيمة Null في التعابير الحسابية يكون الناتج Null.

```
Select FIRST_NAME,SALARY,COMMISSION_PCT from EMPLOYEES
```

في هذا المثال نجد أن العمود Commission_Pct يحتوي قيماً من النوع Null

FIRST_NAME	SALARY	COMMISSION_PCT
Steven	24000	-
Neena	17000	-
Lex	17000	-
Alexander	9000	-
Bruce	6000	-
David	4800	-
Valli	4800	-
Diana	4200	-
Nancy	12008	-
Daniel	9000	-
John	8200	-
Ismael	7700	-
Jose Manuel	7800	-
Luis	6900	-
Den	11000	-
More than 15 rows available. Increase rows selector to view more rows.		

```
Select FIRST_NAME,SALARY, COMMISSION_PCT , SALARY +COMMISSION_PCT
 from EMPLOYEES where FIRST_NAME='Lex'
```



في هذا المثال تم تناول العمليات الحسابية على القيم Null، حيث أن الـ Commission_Pct الخاص بالموظف 'Lex'يأخذ القيمة Null، وكما تعلمنا أن ناتج أي عملية حسابية طرفها القيمة Null يكون Null.

2. أنواع العلاقات

2.1. الربط بين جدولين أو أكثر

كما مر معنا سابقاً، كثيراً ما نحتاج إلى استرجاع بيانات من جدولين أو أكثر في استعلام واحد Query. إجراء مثل ذلك يتطلب منا إنشاء علاقة بين الجدولين أو الجداول، حيث تكمن القوة الحقيقية لقاعدة بيانات علائقية في قدرتها على استرجاع سجلات من جداول مختلفة في استعلام واحد. توجد لدينا العديد من الأسباب التي تدعونا إلى إنشاء علاقة بين الجداول منها:

1) دمج الأعمدة من جدولين أو أكثر.
2) اختيار أعمدة موجودة في جدول واحد بناءً على شرط يطبق على عمود أخر (Self Join).

2.2. استرجاع بيانات من جدولين أو أكثر

(Retrieving Data from Multiple Tables)

عند استرجاع بيانات من جدولين أو أكثر هناك عدة اعتبارات يجب مراعاتها:

1) وضع أسماء الأعمدة في جملة Select.
2) وضع أسماء الجداول في جملة From، على أن يتم الفصل بينها بالفاصلة.
3) إضافة شرط الربط في جملة Where.
4) الصيغة العامة لإسترجاع بيانات من جدولين

```
Select    table1.column, table2.column
From      table1, table2
Where     table1.column1 = table2.column2;
```

```
Select Emp.Empno, Emp.Ename, Dept.Dname
From Emp, Dept Where Emp. Deptno=Dept.Deptno;
```

في هذا المثال تم استرجاع بيانات من جدولي Dept, Emp بينهما علاقة، حيث تم كتابة اسم الجدول قبل اسم العمود مع الفصل بينهما بنقطة في جملةSelect، و كتابة الجداول مصدر البيانات في جملة From، مع كتابة شرط الربط في جملة Where.

في المثال السابق نجد أن أعمدة الربط لها نفس الاسم Deptno، وهذا يسبب مشاكل مع نظام ادارة قواعد البيانات حيث لا يعرف لأي جدول تنتمي هذه الأعمدة ولذلك تم وضع اسم الجدول قبل اسم العمود، مع الفصل بينهما بنقطة.

2.3. أنواع الروابط

2.3.1.**الرابطة المتكافئة Equijoin:** لابد أن كل سجل في الجدول الأب يقابله سجل أو سجلات في الجدول الإبن المرتبط به.

2.3.2.**الرابطة غير المتكافئة Nonequijoin:** لا تمتلك الجداول أعمدة وقيماً مشتركة.

2.3.3.**الرابطة الخارجية Outerjoin:** تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن السجلات المرتبطة يمكن أن تحتوي قيم مشتركة ويمكن أن لا تحتوي، بمعنى أن الجدول الرئيس الأب يحتوي سجل أو سجلات لا يقابلها سجل أو سجلات في الجدول الإبن المرتبط به.

2.3.4.**الروابط الذاتية Self Join:** يمكننا عمل رابطة ذاتية عند ربط الجدول مع نفسه.

2.3.5.**الرابطة الديكارتية Cartesian Join:** للحصول على الرابطة الديكارتية نقوم بحذف شرط الربط بين الجدولين في جملة Where، عندها سيقوم نظام ادارة قواعد البيانات بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني مع نفسه.

2.3.5.1. الرابطة المتكافئة Equijoin: على سبيل المثال إذا أردنا تحديد اسم الإدارة لكل موظف فما علينا سوى المقارنة بين قيم deptno في كل من الجدولين emp و dept. وفق الصيغة التالية وهنا يكون الربط المتكافئ كون كل سجل من جدول الموظفين emp يرتبط بسجل واحد او اكثر من جدول الاقسام dept.

```
Select emp.empno, emp. ename, emp.deptno, Dept.deptno, dept.loc  From emp, dept
Where emp.deptno=dept.deptno
```

1. في هذا المثال نجد أن جدول Dept لا يوجد به إدارة ليس بها عاملين في جدول Emp، حيث في هذا الاستعلام تم استرجاع البيانات لجميع الإدارات.
2. حيث أن عمود Deptno له نفس الاسم في كلا الجدولين، تم وضع اسم الجدول قبل اسم العمود، مع الفصل بينهما بنقطة.
3. يطلق على العلاقة Equijoin أيضاً Simple Join أو Inner Join

2.4. استخدام أكثر من شرط في جملة Where

عند استرجاع بيانات من أكثر من جدول، يمكننا إضافة أكثر من شرط في جملة الشرط بعد شرط الربط، مثلا لإسترجاع بيانات محللي النظم من رقم، اسم، الوظيفة، رقم الإدارة، موقع الإدارة، لابد من إضافة شرط في عبارة Where كمايلي:

```
Select Empno, Ename, Emp.Job, Emp.Deptno, Loc
From Emp, Dept Where Emp.Deptno=Dept.Deptno
And Job='Analyst';
```

2.5. استخدام Table Alias في جملة Where

كما سبق وتعلمنا أنه يمكن استخدام اسم اعتباري للعمود، يمكننا أيضاً وضع اسم اعتباري للجدول. فعلي سبيل المثال يمكننا إجراء مثال1 السابق في جملة Where باستخدام الأسماء الاعتبارية للجداول Table Aliases كما في العبارة التالية:

```
Select E.Empno, E.Ename, E.Deptno, D.Deptno, D.Loc
From Emp E, Dept D  Where E.Deptno=D.Deptno;
```

هناك عدة اعتبارات يجب مراعاتها عند استخدام الأسماء الاعتبارية:

1) الاسم الاعتباري للجدول لا يزيد عن 30 حرفاً.

2) لابد أن يستخدم الاسم الاعتباري أولاً في جملة Where.

3) يفضل أن يكون الاسم الاعتباري معبراً.

2.6. استرجاع بيانات من عدة جداول (More than Two Tables)

في بعض الأحيان نريد استرجاع بيانات من أكثر من جدولين. على سبيل المثال نريد استرجاع بيانات الإسم ورقم الطلبية والأصناف وإجمإلي الأصناف، إجمإلي الطلبيات للعميل Tarek. يمكن استرجاع تلك البيانات من ثلاث جداول هي Customer, Ord, Item بكتابة العبارة التالية:

```
Select C.Name, O.Ord, I.Item, I.Itemtot, O.Total
From Customer C, Ord O, Item I Where C.Custid=O.Custid and O.Ordid=I.Ordid and
  C.Name='Tarek';
```

الرابطة غير المتكافئة Nonequijoin

نادراً ما نحتاج إلى ربط سجلات من جدولين لا يمتلكان أعمدة ولاقيماً مشتركة. إن مثل تلك الرابطة تسمى رابطة غير متكافئة، كما في المثال التالي:

```
Emp          Salgrade
EMPNO ENAME      SAL
------ ------- ------
1      tarek       4000
2      ayman       2850
3      said        2450
4      ali         2998
5      ahmed       1350
6      helal       1700
7      ebrahim     1600
...
25 rows selected.
```

الراتب في جدول emp يقع بين أقل راتب و أكبر راتب في جدول SALGRADE

```
RADE      LOSAL
          HISAL
----- ----- ------
    1   800 1300
2   1301        1400
3   1401        2500
4   2500        9999
```

- ✓ العلاقة بين الجدولين في عمود Sal في جدول Emp ينحصر بين عمودي Losal و Hisal في جدول Salgrade.
- ✓ في الروابط غير المتكافئة لا تستخدم معامل المساواة.

```
Select Emp.Ename, Emp.Sal, Salgrade.Grade From      Emp, Salgrade Where  Emp.Sal
Between  Salgrade.Losal and Salgrade.Hisal;
```

الرابطة الخارجية Outerjoin

تمتلك الجداول المرتبطة أعمدة مشتركة، ولكن السجلات في الجداول المرتبطة يمكن أن تحتوي قيماً مشتركة ويمكن أن لا تحتوي. على سبيل المثال إذا احتوى جدول Departments على إدارة لا يوجد فيها عاملين، معنى ذلك أن تلك الإدارة لن يتم اختيارها في استعلام يربط بين الجدولين. في الشكل التوضيحي التالي نجد أن لا يوجد موظفين في الإدارة Control And Credit

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance
110	Accounting
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing
180	Construction
190	Contracting
200	Operations

FIRST_NAME	DEPARTMENT_ID
Steven	90
Neena	90
Lex	90
Alexander	60
Bruce	60
David	60
Valli	60
Diana	60
Nancy	100
Daniel	100
John	100
Ismael	100
Jose Manuel	100
Luis	100
Den	30
Alexander	30
Shelli	30
Sigal	30
Guy	30
Karen	30

فإذا قمنا بكتابة العبارة التالية:

```
Select  E.First_Name, D.Department_Id, D.Department_Name  From  Employees E,
Departments D
Where E.Department_Id = D.Department_Id
```

FIRST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Jennifer	10	Administration
Pat	20	Marketing
Michael	20	Marketing
Sigal	30	Purchasing
Karen	30	Purchasing
Shelli	30	Purchasing
Den	30	Purchasing
Alexander	30	Purchasing
Guy	30	Purchasing
Susan	40	Human Resources
Kevin	50	Shipping
Jean	50	Shipping
Adam	50	Shipping
Timothy	50	Shipping
Ki	50	Shipping
Girard	50	Shipping
Douglas	50	Shipping

نجد أن هناك سجلات لا تحقق شرط الربط :

(Where E.Department_Id = D.Department_Id)

بالتالي لن يتم استرجاعها في نتيجة الاستعلام، و حيث لا يوجد موظفين بالإدارة Credit Control And، وبالتالي لن تظهر في نتيجة الاستعلام طبقاً لـ Equijoin. لكي نعالج ذلك لابد من إنشاء رابطة خارجية و ذلك بوضع إشارة الجمع (+) محاطة بقوسين وذلك بعد اسم محدد شرط الربط في جملة Where وذلك للجدول الذي يحتوي السجلات التي لا يقابلها سجل أو سجلات في الجدول المرتبط.

```
Select  E.First_Name,
D.Department_Id,
D.Department_NameFrom
Employees E, Departments D
 Where E.Department_Id(+) =
D.Department_Id;
```

في هذا المثال الإدارة Control And Credit سيتم عرضها في نتيجة هذا الاستعلام، وذلك من خلال إنشاء رابطة خارجية وذلك بوضع إشارة الجمع (+) محاطة بقوسين بعد اسم محدد شرط الربط في جملة Where وذلك للجدول الذي يحتوي السجلات التي لا يقابلها سجل أو سجلات في الجدول المرتبط وهو في مثالنا جدول Employees.

الرابطة الذاتية Selfjoin

يمكننا عمل الرابطة الذاتية عند ربط جدول ما مع نفسه، فعندما نريد ربط جدولاً مع نفسه نكتب اسم نفس الجدول عدة مرات في جملة Where، و بالتالي يتوجب علينا استخدام أسماء اعتبارية

في المثال الموضح بالشكل التالي الجدول Employees يحتوي على عمود (Manager_Id) الذي يحدد رقم المدير. الرقم Employee_Id في عمود المفتاح الرئيسي بجدول Employees، لذلك توجد علاقة من Selfjoin بين عمود Manager_Id وعمود Employee_Id.

مثال:

```
Select  Worker.First_Name,Manager.First_Name
From  Employees  Worker,  Employees  Manager
 Where Worker.Manager_Id=
Manager.Employee_Id
```

FIRST_NAME	FIRST_NAME
William	Gerald
Lisa	Gerald
Sundita	Gerald
Tayler	Gerald
Harrison	Gerald
Elizabeth	Gerald
Alexander	Lex
Clara	Alberto
Mattea	Alberto
David	Alberto
Danielle	Alberto
Amit	Alberto
Sundar	Alberto
Nandita	Adam
TJ	Adam
James	Adam

الرابطة الديكارتية Cartesian Joins

للحصول على الرابطة الديكارتية نقوم بحذف شرط الربط بين الجدولين في جملة Where، عندها سيقوم SQL بربط كل سجل في الجدول الأول مع كل سجل في الجدول الثاني. هذا يجعل SQL يقوم بإنشاء ومعالجة كمية كبيرة جداً من البيانات لا قيمة لها، فعلى سبيل المثال إذا كان جدول Departments يحتوي 10 سجلات و جدول Employees يحتوي 30 سجل ،فإذا تم حذف شرط الربط من جملةWhere في نتيجة الاستعلام يكون لدينا (10*30=300) سجل. لحساب السجلات المسترجعة من رابطة ديكارتية للجدولين Employees, Departments يتم كتابة الاستعلام التالي:

```
Select Count(*) From Departments, Employees;
```

تكون النتيجة Count(*)=300

3. استخدام الدوال Functions

3.1. استخدام الدوال لأكثر من سطر Multi Row Functions

تزودنا SQL بعدة دوال تجميع Group Functions يمكن أن تنفذ على مجموعة من السجلات وهي:

الدالة	الاستخدام
Count	حساب عدد السجلات في جدول ما
Max	تحديد القيمة العظمى لعمود ما
Min	تحديد القيمة الدنيا لعمود ما
Avg	حساب المتوسط الحسابي لعمود ما
Stddev	حساب الانحراف المعياري لعمود ما
Sum	حساب المجموع الكلي لعمود ما
Variance	حساب التغاير لعمود ما

تقوم الدوال التجميعية Group Functions بمعالجة قيم العمود المختار من الجدول وتقدم النتيجة في شكل قيمة وحيدة تخص العمود المختار. يجب علينا إتباع التعليمات التالية عند إستخدام Group Function :

1) وضع اسم العمود بين قوسين ( ) بعد الدالة مباشرة

```
Select Max(Salary)From Employees;
```

2) يمكن لجميع الأعمدة في جملة Select أن تكون كلها دوال تجميعية

```
Select Max(Salary), Min(Salary), Sum(Salary);
```

3) استخدام دالة تجميعية أخرى (أي تركيب الدوال التجميعية) غير مسموح به في SQL.

```
Select Max(Avg(Salary)) From Employees;
```

4) الدوال التجميعية تتجاهل القيم الفارغة Nullvalues

5) يمكن استخدام الدوال التجميعية ضمن التعابير الحسابية.

3.1.1. استخدام Count

نستخدم الدالة Count لحساب عدد السجلات التي تحتوي قيماً غير فارغة (Not Null) في العمود المحدد في المجموعة. فمثلاً لحساب عدد السجلات في جدول Employees الذي يمتلك قيماً غير فارغة في العمود First_Name نكتب العبارة التالية:

```
Select Count (First_Name) From Employees;
```

- ✓ تم وضع العمود المراد حساب عدد سجلاته بين قوسين بعد الدالة Count.
- ✓ لا يتم عد السجلات التي تحتوي قيم فارغة Null.

3.1.2. استخدام Count(*)

عندما نستبدل اسم العمود بالنجمة (*) تقوم الدالة Count بحساب عدد السجلات بما فيها السجلات التي تحتوي على قيم فارغة، فعلي سبيل المثال:

```
SelectCount (*)
From  Employees
Where       Department_Id = 10;
```

3.1.2.1. استخدام Distinct مع Count

يمكننا استخدام كلمة Distinct مع Count، حيث تقوم باستبعاد السجلات المكررة. على سبيل المثال لحساب عدد السجلات في جدول Employees مع عدم استبعاد القيم الفارغة في العمود First_Name نكتب العبارة التالية:

```
Select Count (Distinct (First_Name)) From Employees ;
```

3.1.3. استخدام Max

نستخدم الدالة Max لحساب أكبر قيمة في مجموعة ما، حيث يمكن استخدام الدالة Max مع القيم العددية أو النصية أو التاريخ والوقت. على سبيل المثال:

- ✓ يمكن استخدام Max مع أي نوع من البيانات.
- ✓ عند استخدام Max مع البيانات المحرفية فإنها تعيد أعلى قيمة Ascii، وعند استخدامها مع الأعمدة العددية تعيد أعلى قيمة جبرية، وعند استخدامها مع التاريخ والوقت تعيد القيمة الأحدث في العمود.

```
Select Max(Salary), Max(Hire_Date), Max(First_Name)
From Employees ;
```

- ✓ تهمل الدالة Max القيم Null.

3.1.4. استخدام الدالة Min

```
SELECT     MIN(SALARY)
FROM       EMPLOYEES
WHERE DEPARTMENT_ID =10;
```

تُعيد الدالة Min أصغر قيمة لا تساوي Null، كما أنها تستخدم مع أي نوع من البيانات. على سبيل المثال نستخدم الدالة Min لتحديد أقل مرتب في الإدارة رقم 10.

3.1.5. استخدام الدالة Avg

نستخدم الدالة Avg لحساب المتوسط الحسابي لعمود ما، حيث يعيد Avg القيمة الوسطية لجميع القيم التي لا تساوي Null في العمود. هناك عدة اعتبارات يجب مراعاتها عند استخدام Avg :

```
SELECT    AVG(SALARY)   FROM        EMPLOYEES;
```

- ✓ يستخدم Avg مع القيم العددية فقط.
- ✓ يهمل Avg القيمة Null عند حساب99ه للقيمة الوسطي.
- ✓ يهمل Avg القيم المتكررة في العمود عند حسابه للقيمة الوسطية وذلك إذا صرحنا بكتابة Distinct قبل اسم العمود.

يمكننا حساب المتوسط الحسابي لعمود Sal في جدول Emp، وذلك بكتابة العبارة التالية:

```
SELECT    MAX(SALARY), MIN(SALARY), AVG(SALARY) FROM     EMPLOYEES;
```

يمكننا حساب أعلى قيمة و أدنى قيمة والمتوسط الحسابي لعمود Sal بكتابة العبارة التالية:

يمكننا حساب المتوسط الحسابي لعمود Commission_Pct في جدول Emp، وذلك بكتابة العبارة التالية:

```
Select Avg(Commission_Pct)
From  Employees
```

نلاحظ في هذا المثال أن Avg تتجاهل القيم Null.

3.1.6. استخدام الدالة Sum

نستخدم الدالة Sum لحساب المجموع الكلي لقيم عمود ما. هناك عدة اعتبارات يجب مراعاتها عند استخدام Sum:

- ✓ تُستخدم Sum مع القيم العددية فقط..
- ✓ تهمل Sum القيمة Null عند حسابه المجموع الكلي.
- ✓ تهمل Sum القيم المتكررة في العمود عند حسابه المجموع الكلي وذلك إذا صرحنا بكتابة Distinct قبل اسم العمود.

يمكننا حساب المجموع الكلي لعمود Salary في جدول Employees ، وذلك بكتابة العبارة التالية:

```
Select Sum(Salary) From  Employees;
```

3.1.7. استخدام الدالة Nvl مع الدوال التجميعية Group Functions

```
SELECT AVG(NVL(COMMISSION_PCT,0))
FROM   EMPLOYEES;
```

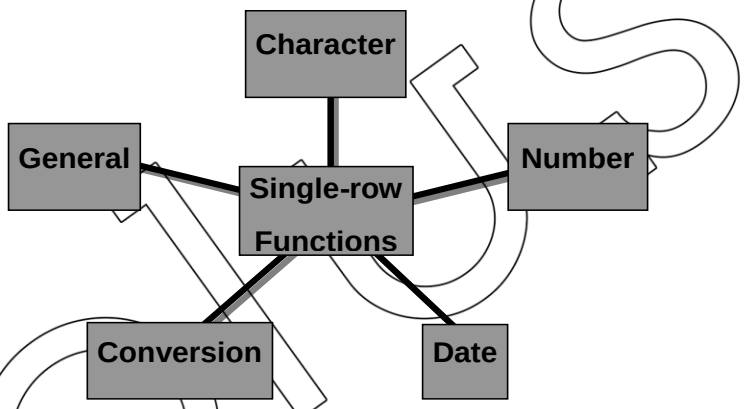
نستخدم الدالة Nvl مع الأعمدة التي تحتوي القيم الفارغة Null، حيث تقوم بمعالجة مشاكل القيم الفارغة باستبدالها بالقيمة الصفرية وبالتالي يمكننا استخدام الدوال التجميعية مع الأعمدة التي تحتوي قيم فارغة لكي تشمل جميع السجلات.

3.2. استخدام الدوال لسطر واحد

(Single Row Functions)

تعالج Single Row Functions قيم العمود المختار من الجدول وتقدم النتيجة على شكل قيمة وحيدة لكل سجل، في حين تُستخدم Group Function في معالجة قيم العمود المختار من الجدول وتقدم النتيجة على شكل قيمة وحيدة تخص العمود المختار

3.2.1. أنواع Single Row Functions



3.2.2. الدوال النصية Character Functions

يمتلك SQL عدد من الدوال التي تستخدم لتعديل أنماط البيانات النصية(Char, Varchar)، وسوف نتعرض لأهم هذه الدوال النصية و هي كما في الجدول التالي:

الدالة النصية	الاستخدام
Initcap	تغيير حالة الحرف الأول من كل سلسلة نصية إلى حرف كبير و تحويل حالة الأحرف التالية للحرف الكبير إلى حالة الأحرف الصغيرة.
Length	إعادة عدد الأحرف المكونة لسلسلة حرفية
Lower	تحويل كل حرف ضمن سلسلة حرفية إلى أحرف صغيرة
Upper	تحويل كل حرف ضمن سلسلة حرفية إلى أحرف كبيرة
Concat	دمج سلسلة حرفية مع سلسلة حرفية أخري
Replace	استبدال سلسلة حرفية بسلسلة حرفية أخري
Soundex	إيجاد الكلمات المتشابهة من حيث اللفظ الصوتي
Substr	نسخ الجزء المحدد من الكلمة ابتداء من حرف محدد
Instr	إعادة قيمة عددية تمثل موضع مجموعة أحرف ضمن سلسلة حرفية

سنقوم باستعراض أمثلة على الدوال النصية كما هو موضح في الجدول التالي:

Function الدالة	Result النتيجة
Lower ('Jicc Center')	('Jicc Center')
Upper ('Jicc Center')	('Jicc Center')
Initcap ('Jicc Center')	('Jicc Center')
Concat('Jicc'),('Center')	Jicccenter
Substr('String',1,3)	Str
Length('String')	6
Instr('String','T')	2

```
SELECT  FIRST_NAME, CONCAT
(FIRST_NAME, JOB_ID),
LENGTH
(FIRST_NAME),INSTR(FIRST_NAME,
'A')
FROM   EMPLOYEES;
```

3.2.3. الدوال الرقمية Number Functions

يمتلك Sql عدد من الدوال الرقمية التي تُستخدم مع أنواع البيانات الرقمية Numeric Data، وسوف نتعرض لأهم هذه الدوال الرقمية و هي موضحة في الجدول التالي:

الدالة الرقمية	الاستخدام
Round	تقريب قيمة رقمية إلى عدد أرقام محدد بعد الفاصلة
Trunc	قطع أو قص القيم إلى دقة محددة
Mod	تحديد باقي القسمة الناتج عن عملية ما

سنقوم باستعراض أمثلة على الدوال الرقمية كما هو موضح فيما يلي:

- Round (42.926,2) 42.93
- Trunc (44.926,2) 44.92
- Mod (1600,300) 100

مثال أخر:

```
SELECT TRUNC(35.832,2),TRUNC(65.323), TRUNC (65.843,-1)
FROM   DUAL;
```

3.2.4.التاريخ ودوال التاريخ

Date and Date Functions

يمتلك Sql عدداً من الدوال التي تستخدم مع أعمدة التاريخ، حيث يمكن إظهار التاريخ بشكل حرفي. يمكننا استخدام التاريخ في عمليات الجمع والطرح. كما يمكن إضافة قيمة ما لتاريخ ما ونحصل بذلك على قيمة تاريخ جديدة. سوف نتعرض لأهم هذه الدوال وهي موضحة في الجدول التالي:

الدالة	الإستخدام
Months_Between(Date1,Date2)	يحدد عدد الأشهر بين تاريخين
Next_Day(Date,'Day')	يحدد إليوم التالي من الأسبوع الذي يلي التاريخ المحدد
Add_Months(Date,Numberof Months)	يستخدم لحساب تاريخ جديد بناء على عدد محدد من الأشهر
Last_Day(Date1)	يحدد آخر يوم في الشهر الحالي أو لتاريخ محدد

سنقوم في المثال التالي باستعراض أمثلة على الدوال التاريخية كما هو موضح فيما يلي:

Months_Between ('1-Sep-93','11-Jan-44') 19.6774194

Add_Months('11-Jan-96',6) '11-Jul-96'

Next_Day('01-Sep-95','Friday') '8-Sep-95'

Last_Day('01-Sep-95') '30-Sep-95'

✓ Sysdate دالة SQL Function تُستخدم لاسترجاع التاريخ و الوقت الحاليين.

```
Select Sysdate from   Dual;
```

3.3. الدوال التحويلية Conversion Functions

يستخدم Sql عدة دوال تحويلية. من أهم هذه الدوال التحويلية نذكر:

✓ To_Number : يُستخدم لتحويل نوع البيانات إلى نوع البيانات الرقمية. مثلاً يمكننا استخدام To_Number لتحويل السلسلة الحرفية السنة إلى حقل رقمي وذلك باستخدام أحد الدوال الرقمية على هذا الحقل مثل Mod.

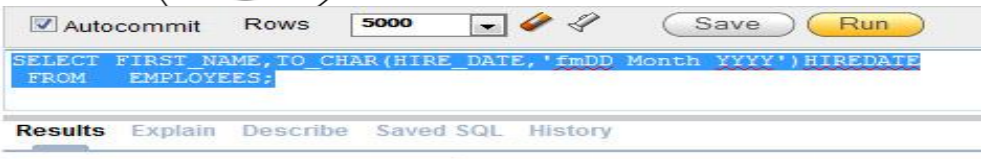
✓ To_Char : يُستخدم لتحويل نوع البيانات إلى نوع البيانات النصية. مثلاً يمكننا استخدامه لتغيير تاريخ ميلاد موظف إلى حقل حرفي وذلك لتنسيقه ليظهر بتنسيق خاص. تعتبر هذه الدالة من أهم الدوال التحويلية في نظام لإدارة قواعد البيانات أوراكل Oracle مثلا ، ومن أهم استخداماتها هو تحويل نوع البيانات الرقمية إلى نوع البيانات النصية وذلك لإضفاء عليها تنسيق خاص مثل إظهار علامة بجوار الرقم والتحكم في رمز العلامة العشرية وعدد الأرقام بعد العلامة العشرية وهناك عدة رموز يمكننا استخدامها عند تحويل البيانات الرقمية إلى نوع البيانات النصية

كما هو موضح بالجدول التالي :

الرمز	الاستخدام
9	يمثل رقم Number
0	يمثل الصفر Zero
$	يمثل علامة الدولار Dollar Sign
L	يمثل علامة العملة على النظام Local Currency Symbol
.	يمثل رمز العلامة العشرية Decimal Point
,	يمثل رمز تجميع الأرقام A Thousand Indicator

✓ To_Date : تُستخدم لتحويل نوع البيانات إلى نوع البيانات التاريخية.

في هذا المثال سنقوم بتحويل نوع بيانات عمود Hire_Date من نوع Date إلى نوع البيانات الحرفية وذلك باستخدام الدالة To_Char وذلك لعمل تنسيق خاص لحقل التاريخ.



FIRST_NAME	HIREDATE
Steven	17 June 2003
Neena	21 September 2005
Lex	13 January 2001
Alexander	3 January 2006
Bruce	21 May 2007
David	25 June 2005
Valli	5 February 2006
Diana	7 February 2007
Nancy	17 August 2002
Daniel	16 August 2002
John	28 September 2005
Ismael	30 September 2005
Jose Manuel	7 March 2006
Luis	7 December 2007
Den	7 December 2002
Alexander	18 May 2003
Shelli	24 December 2005
Sigal	24 July 2005

ملاحظات:

- ✓ تنسيق التاريخ لابد أن يوضع بين علامتي تنصيص مفردة ' '.
- ✓ نفصل بين التاريخ والتنسيق بفاصلة.

في هذا المثال سنقوم بتحويل نوع بيانات عمود Salary من نوع Number إلى نوع البيانات الحرفية وذلك باسخدام To_Char وذلك لعمل تنسيق خاص لحقل المرتب مع استخدام الرموز التي سبق الإشارة إليها:

```
SELECT TO_CHAR(SALARY,'$99,999') SALARY
FROM        EMPLOYEES;
```

SALARY
$24,000
$17,000
$17,000
$9,000
$6,000
$4,800
$4,800
$4,200
$12,008
$9,000
$8,200
$7,700
$7,800
$6,900
$11,000

3.4. الدوال العامة General Functions

من أهم الدوال العامة في Single Row Function دالتي Nvl،Decod. فيما يلي توضيح لاستخدامات كلتا الدالتين في SQL.

3.4.1. الدلة NVL

تُستخدم الدالة Nvl لاستبدال القيمة الفارغة بقيمة أخرى، ويمكن استخدام Nvl مع نوع البيانات الرقمية وغير الرقمية، و من أهم استخدامات تلك الدالة التغلب على مشكلة القيم الصفرية في البيانات الرقمية إذ تقوم Nvl باستبدال القيم الفارغة بقيم صفرية. هناك عدة اعتبارات يجب مراعاتها عند استخدام الدالة Nvl:

- ✓ تستخدم الدالة Nvl في تحويل القيم الفارغة إلى قيم فعلية.
- ✓ يمكنها معالجة أنواع البيانات الرقميةNumeric Data، والنصية Character String، والتاريخية Date.
- ✓ عند تحويل القيم الفارغة عليك بالالتزام بنفس نوع البيانات Data Types في العمود الذي يحتوي القيم الفارغة

فعلي سبيل المثال:

- Nvl(Commission_Pct,0)
- Nvl(Job_Id,'No Job Yet')
- Nvl(Hire_Date,'01-Jan-98')

```
SELECT FIRST_NAME, SALARY, COMMISSION_PCT,(
SALARY*12)+NVL(COMMISSION_PCT,0)
 FROM   EMPLOYEES;
```

FIRST_NAME	SALARY	COMMISSION_PCT	(SALARY*12)+NVL(COMMISSION_PCT,0)
Steven	24000	-	288000
Neena	17000	-	204000
Lex	17000	-	204000
Alexander	9000	-	108000
Bruce	6000	-	72000
David	4800	-	57600
Valli	4800	-	57600
Diana	4200	-	50400
Nancy	12008	-	144096
Daniel	9000	-	108000
John	8200	-	98400
Ismael	7700	-	92400
Jose Manuel	7800	-	93600
Luis	6900	-	82800
Den	11000	-	132000
Alexander	3100	-	37200
Shelli	2900	-	34800

4. استخدام Group by

فيما سبق كانت SQL تعالج الجدول أو الجداول بالكامل أو السجلات المحققة لشرط معين في جملة Where، إلا أننا في بعض الأحيان نريد إجراء عمليات منطقية على سجلات معينة مثل إظهار القيمة العظمى والدنيا والمتوسط الحسابي لكل إدارة من الإدارات على حدة. لإجراء مثل هذا نحتاج إلى استخدام Group By، حيث تستخدم جملة Group By لتحديد الأعمدة التي يتم التجميع بناءً عليها أو عنما نريد تطبيق الدوال التجميعية مثل Avg, Sum, Count, Max, Min على سجلات من الجدول معزولة على شكل مجموعات فرعية، حيث تميز كل مجموعة بنفس قيم المعطيات، بحيث تقسم Group By سجلات الجدول إلى مجموعات صغيرة متشابهة، ثم يمكننا استخدام Group Functions لاسترجاع معلومات تلخيصية لكل مجموعة من هذه المجموعات على حدة، والصيغة العامة لاستخدام Group By تأخذ الشكل التالي:

```
SELECT    column, group_function(column)
  FROM    table
  [WHERE  condition]
  [GROUP BY     group_by_expression]
  [ORDER BY     column];
```

- ✓ تقع جملة Group by بعد جملة Where أو بعد جملة From في حالة عدم وجود جملة Where.
- ✓ تقسم Group by سجلات الجدول إلى مجموعات صغيرة متشابهة، ثم يمكننا استخدام Group Functions لاسترجاع معلومات تلخيصية لكل مجموعة من هذه المجموعات على حدة.
- ✓ تتعامل Group by مع القيم الفارغة Null كمجموعة مميزة عن غيرها.
- ✓ تمكننا Group by من إنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى مثل: Group by Deptno,Sal

هناك عدة اعتبارات يجب مراعاتها عند استخدام Group by :

- ✓ يمكن أن تحتوي عدة أعمدة وتعابير.
- ✓ يتم وضع فواصل بين أسماء الأعمدة.
- ✓ يجب أن تحتوي على كل الأعمدة المكتوبة بجملة Select و التي لا تستخدمها دوال التجميع.

في هذا المثال نريد حساب المجموع الكلي لمرتبات كل إدارة على حدة. لفعل ذلك يجب علينا استخدام عبارة Group By في جملة Select:

```
SELECT deptno, sum(sal)
FROM  emp
GROUP BY deptno;
```

- ✓ يجب أن تحتوي جملة Group by على كل الأعمدة المكتوبة بجملة Select والتي لا تستخدمها دوال التجميع (وهو في المثال السابق حقل Deptno الذي يراد التجميع بناءً عليه).

يمكننا إجراء المثال السابق و حساب المجموع الكلي لمرتبات كل إدارة على حدة باستخدام عمود Deptno بدون كتابة عمود Deptno في جملة Select.

```
SELECT sum(sal)
FROM emp
GROUP BY deptno;
```

- ✓ يمكننا إنشاء مجموعات من السجلات باستخدام Group by بناءً على أعمدة ليست موجودة بجملة Select.

في هذا المثال سنوضح إمكانية قيام Group by بإنشاء مجموعات فرعية ضمن مجموعات فرعية أخرى.

```
SELECT deptno,job,sum(sal)
FROM    emp
GROUP BY deptno, job;
```

- ✓ أولاً: يتم إنشاء مجموعات متشابهة بناءً على رقم الإدارة Deptno

(First,The Rows Are Grouped By Department Number)

- ✓ ثانياً: في مجموعات الإدارة يتم عمل مجموعات متشابهة من السجلات بناءً على الوظيفة.

(Second, Within The Department Number Groups, The Rows Are Grouped By Job Title.)

عند إجراء المثال التالي تظهر لنا رسالة الخطأ التالية: (Ora-00934 : وظيفة المجموعة غير مسموحة هنا) والتي تعني أننا لا نستطيع استخدام الدوال التجميعية Group Function مع جملة Where.

```
SELECT deptno, MAX(sal)
  FROM         emp
 WHERE MAX (sal) > 2000
 GROUP BY    deptno;
```

5. استخدام Having (تقييد الاختيار ضمن المجموعة (Restriction Group Selection)

قد لا نريد استرجاع دائماً جميع المجموعات الفرعية المختلفة في الجدول ضمن استعلام واحد. تقدم لنا SQL جملة Having لحذف المجموعات الفرعية من التي لا نريد استرجاعها في الاستعلام أو التقرير. تعتبر جملة Having مشابهة لجملة Where من حيث أنها تحدد فيما إذا كانت السجلات قابلة للاختيار أو لا، وتحتوي جملة Having على محددات تستخدم لتقييم السجلات، حيث يمكننا وضع عدة شروط في جملة Having. كما يمكننا استخدام المعاملات المنطقية معها مثل and, or، وقد يتبادر إلى الذهن لماذا لا نستخدم عوضاً عنها جملة Where. لا يمكننا ذلك حيث يكمن الاختلاف بين جملة Having و جملة Where في أن محددات الجملة Having يجب أن تكون تابع تجميع، ويتم استخدام Having فقط عندما يتم حساب قيمة مجمعة بواسطة Select، بينما عمل Where هو ترشيح السجلات التي سيطبق عليها لاحقاً جملةGroup by، و ليس ترشيح المجموعات. الصيغة العامة لاستخدام Having موضحة كالتالي:

```
SELECT column, group_function(column)
FROM       table
[WHERE     condition]
[GROUP BY       group_by_expression]
[HAVING    group_condition]
[ORDER BY       column];
```

هناك عدة اعتبارات يجب مراعاتها عند استخدام Having:

- ✓ تقع جملة Having بعد فقرة Group by.
- ✓ تستخدم Having فقط مع Group By ولا يمكن استخدامها بدونها.
- ✓ نستخدم Having مع الدوال التجميعية التالية فقط (Max,Min,Avg,Count,Sum).
- ✓ يمكن تحديد عدة شروط في جملة Having وذلك باستخدام المعاملات المنطقية and, or.
- ✓ يمكننا استخدام كلاً من Having و Where في جملة Select واحدة.

في المثال السابق نقوم باسترجاع الوظيفة و المجموع الكلي للمرتبات الشهرية لكل وظيفة على حده بشرط أن يكون

```
SELECT job, SUM(sal) PAYROLL
FROM     emp
WHERE    job NOT LIKE 'SALES%'
GROUP BY job
HAVING   SUM (sal)<6000
ORDER BY SUM (sal);
```

المجموع الكلي للمرتبات الشهرية لكل وظيفة أقل من 6000 باستثناء وظيفة Salesman، ثم الترتيب التصاعدي بناءً على المجموع الكلي للمرتبات الشهرية.

6. استخدام و عمل استعلام فرعي (Sub Query)

تمكننا SQL من جعل الاستعلامات متداخلة مع بعضها البعض بشكل نموذجي، بحيث ينتج عن الاستعلام الداخلي (Inner Query) قيماً تفحص في قسم الشرط للإستعلام الخارجي (Outer Query) لمعرفة فيما إذا كان الشرط سيتحقق عندها أم لا.

6.1. كيف تعمل الاستعلامات الفرعية

لنفترض أننا نريد كتابة استعلاماً لاسترجاع من يتقاضى مرتباً أكثر من محمد ؟ نجد أننا نحتاج إلى استعلامين:

الأول: ماهو مرتب محمد؟

الثاني: البحث عن من يتقاضى مرتباً أعلى من مرتب محمد؟

لحل مثل تلك المشكلة لابد من دمج الاستعلامين معاً، بمعنى استخدام استعلام داخل استعلام أخر، حيث يعيد الاستعلام الداخلي (أو الفرعي) قيمة يستخدمها الاستعلام الرئيسي (أو الخارجي) لمعرفة فيما إذا كان سيتحقق الشرط عندها أم لا.

- ✓ ينفذ الاستعلام الفرعي أو الداخلي أولاً ثم ينفذ الاستعلام الرئيسي أو الخارجي.
- ✓ يولد أو يعيد الاستعلام الداخلي نتيجة يستخدمها الاستعلام الخارجي لمعرفة ما إذا سيتحقق الشرط عندها أم لا.
- ✓ الصيغة العامة لعمل واستخدام الاستعلامات الفرعية Sub Queries كالتالي:

في المثال السابق نجد أن الاستعلام الداخلي يحدد مرتب الموظف الذي رقمه 1، يأخذ الاستعلام الخارجي (أو الرئيسي)

```
SELECT select_list
FROM table
WHERE expr operator
                (SELECT   select_list
               FROM       table);
```

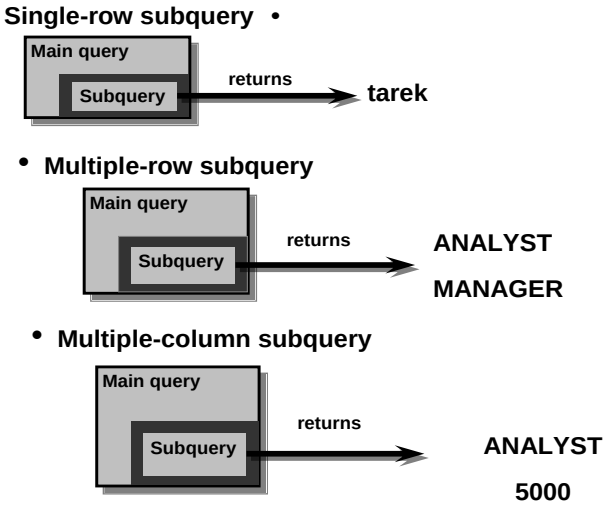
هذه النتيجة ويستخدمها لاسترجاع الموظفين الذين يتقاضون مرتباً أعلى من هذا المرتب.

```
SELECT ename
FROM   emp      5000
WHERE  sal >
                (SELECT sal
FROM   emp
WHERE  empno=1);
```

6.2. أنواع الاستعلامات الفرعية

Types of Sub Queries

1. الإستعلامات الفرعية التي تعيد سجل واحد (Single-Row Sub Queries)
2. الاستعلامات الفرعية التي تعيد أكثر من سجل واحد (Multiple-Row Sub Queries)
3. الاستعلامات الفرعية التي تعيد أكثر من عمود واحد (Multiple-Column Sub Queries)



6.2.1.الاستعلامات الفرعية التي تعيد سجل واحد

كما سبق وتعلمنا أن (Single–Row Sub Queries) تعيد سجل واحد فقط، ويستخدم ذلك النوع من الاستعلامات مع عوامل المقارنة الموضحة بالجدول التالي :

المعنى Meaning	المعامل Operator
يســـاوي	=
أكبرمن	>
أكبرمن أو يساوي	=<
أصغر من	<
أصغر من أو يساوي	<=
لا يساوي	! = or <>

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) لل9موظفين الذين لهم نفس الوظيفة للموظف رقم 555.

```
SELECT ename,job
FROM emp
WHERE job=
                (SELECT    job
                FROM emp
        Where empno=555);
```

في هذا المثال سنقوم باسترجاع بيانات الموظفين (الاسم، الوظيفة) للموظفين الذين لهم نفس الوظيفة للموظف رقم 7698 ومرتباتهم أكبر من راتب الموظف رقم 7934.

```
SELECT   ename, job
FROM     emp          MANAGER
WHERE    job =
              (SELECT  job
                   FROM              emp
                     WHERE           empno = 7698)
  AND      sal >       1300
                   (SELECT           sal
                   FROM    emp
                   WHERE  empno = 7934);
```

- ✓ لابد من وضع الاستعلام الفرعي بين قوسين.
- ✓ لا تستخدم جملة Order By داخل الاستعلام الفرعي.

6.2.1.1. استخدام الدوال التجميعية في Sub Query

تتميز الدوال التجميعية بأنها تولد أو تعيد قيمة وحيدة مهما كان عدد السجلات المختارة. ينيغي علينا الانتباه إلى أن الدوال التجميعية التي هي دوال تجميع معرفة في جملة Group By ستولد وتعيد عدة قيم. لهذا السبب لا يسمح باستخدامها مع هذا النوع من الاستعلامات الفرعية التي تعيد سجل واحد، حتى لو استخدامناGroup By و Having لتوليد مجموعة وحيدة لخرج الاستعلام الفرعي. لهذا يجب علينا استخدام الدوال التجميعية المرغوبة في جملة Where و التي تحذف المجموعات غير المرغوبة.

في هذا المثال سنقوم باسترجاع بيانات الموظف (الاسم، الوظيفة، المرتب) الذي يتقاضى أعلى مرتب، حيث يمكننا تنفيذ هذا الاستعلام باستخدام Group Function داخل الاستعلام الفرعي كما يلي:

```
SELECT ename, job, sal
FROM   emp
WHERE  sal =          5000
                      (SELECTMAX(sal)
               FROM     emp);
```

6.2.1.2. استخدام (Group by و Having) مع Sub Query

يمكن أن نستخدم الاستعلامات الفرعية في جملة Having كما يمكن استخدام الدوال التجميعية أو Group by و Havingكما في المثال التالي:

```
SELECT deptno, MIN(sal)
FROM   emp
GROUP BY        deptno
HAVINGMIN(sal) >
                (SELECTMIN(sal)
                FROM    emp
                WHERE  deptno = 10);
```

في هذا المثال تم إنشاء مجموعات أصغر مرتبات بناءً على عمود رقم الإدارة، مع تقييد ذلك بشرط وهو أن يكون أقل مرتب في تلك الإدارات أكبر من أقل مرتب في الإدارة رقم 10.

- ✓ يجب علينا التأكد عند استخدام الاستعلامات الفرعية في القسم الشرطي (الذي فيه عامل المساواة أو عدم المساواة) من أن الاستعلام الفرعي سيعيد سجلاً واحداً فقط، وإلا سيصدر خطأ إذا أعاد أكثرمن قيمة واحدة. لن يفشل الأمر إذا لم يعيد الاستعلام الفرعي أية قيمة ولكننا عندها لن نحصل على خرج من الاستعلام الفرعي.

6.2.2.الاستعلامات الفرعية التي تعيد أكثر من سجل واحد

تعيد الاستعلامات الفرعية متعددة السجلات أكثر من سجل، وتتطلب هذه الأنواع من الاستعلامات الفرعية معاملاً يمكن استخدامه لتقييم عدة قيم. على سبيل المثال لا الحصر يمكننا استخدام المعامل In مع الاستعلامات الفرعية متعددة الأسطر لأنه يقييم مصفوفة من القيم. الجدول التالي يوضح المعاملات التي يمكننا استخدامها لتقييم الاستعلامات متعددة السجلات.

الوصف	المعامل Operator
يكون المحدد مساوياً لأي من القيم التي يعيدها الإستعلام الفرعي لهذا الشرط حتى يصبح محققاً (True)	In
يقارن المحدد بكل قيمة يعيدها الإستعلام الفرعي ،ويكون الشرط محققاً (True) إذا كان أي عنصر من المجموعة يحقق الشرط	Any
يقارن المحدد بكل قيمة يعيدها الإستعلام الفرعي، ويكون الشرط محققاً	All

في هذا المثال نريد استرجاع بيانات الموظفين (رقم الموظف، الاسم، الوظيفة) ذوي المرتبات أقل من أي من الموظفين الذين يعملون بوظيفة Clerk والذين لا يعملون في وظيفة Clerk

```
SELECT empno, ename, job    1300
FROM    emp              1100
WHERE   sal < ANY        1300
                    (SELECT    sal    950
                    FROM       emp
                    WHERE      job = ' CLERK ')
AND     job <> 'CLERK';
```

- ✓ Any <: تعني أقل من Minimum.
- ✓ Any >: تعني أقل من Maximum.
- ✓ Any =: تعادل تكافيء معامل In.

6.2.3. الاستعلامات الفرعية التي تعيد أكثر من عمود واحد

(Multiple-Column Sub Queries)

رأينا في الأمثلة السابقة استعلامات تستخدم أعمدة وحيدة كمحددات تقييم، وأحياناً لتنفيذ نوع من الشروط يجب علينا استعراض عدة أعمدة وكأنها محدد واحد، ويجب إحاطة تلك الأعمدة بين قوسين ووضع فواصل بين الأعمدة.

```
SELECT     ename, deptno, sal, comm
FROM       emp
WHERE   (sal, NVL (comm,0)) IN
                        (SELECT sal, NVL (comm,0)
                    FROM   emp
                    WHERE deptno = 30);
```

في المثال السابق نريد استرجاع بيانات الموظف(الاسم، رقم الإدارة، المرتب، الحوافز)، لأي موظف يتماثل في المرتب والحوافز مع أي موظف في الإدارة رقم 30.

7. عبارة Insert كيفية إضافة سجلات جديدة

تمتلك معظم برامج أنظمة قواعد البيانات أدوات ممتازة لإدخال وتعديل وحذف البيانات من الجداول، وهي بلا شك أفضل وأسرع بكثير من إدخال البيانات بعبارات SQL، إلا أن ذلك يمكن المبرمج من معالجة البيانات ضمن برامجه التي يكتبها. نستطيع إضافة سجلات جديدة إلى الجدول باستخدام عبارة Insert، ويوجد لدينا صيغتين من عبارة Insert: الأولى تسمح بإضافة سجل واحد في نفس الوقت إلى الجدول، والثانية تسمح باختيار مجموعة سجلات من جدول وإدراجها في جدول آخر.

7.1. إضافة سجل واحد إلى الجدول

كما وضحنا سابقاً أن عبارة Insert لها صيغتين الأولى والتي نحن بصددها تمكننا من إضافة سجل واحد إلى الجدول في نفس الوقت، والصيغة العامة لها تأخذ الشكل التالي:

```
INSERT INTO table [(column [, column...])]
VALUES      (value [, value...]);
```

مما سبق نجد أن الصيغة العامة لعبارة Insert تتألف مما يلي:

1. Insert,Into,Values هي كلمات محجوزة في SQL.
2. Table ويشير إلى اسم جدول وحيد في قاعدة البيانات، وبالتحديد الجدول الذي نريد إدراج السجلات فيه.
3. Column: وتشير إلى أعمدة الجدول.
4. لا ئحة من القيم محاطة بين قوسين تقوم SQL بإدراجها في الأعمدة المقابلة لها على الترتيب.

لإدراج سجل من البيانات إلى جدول Dept، نقوم بكتابة العبارة التالية:

```
INSERT INTO   dept (deptno, dname, loc)
  VALUES      (50, 'DEVELOPMENT', 'cairo');
1 row created.
```

لإدراج سجل من البيانات إلى جدول Emp، نقوم بكتابة العبارة التالية:

```
INSERT INTO   EMP (empno, ename, job,
              mgr, hiredate, sal, comm,
              deptno)
VALUES        (7369, 'ayman', 'programmer',
              7372, SYSDATE, 4000, NULL,
              50);
1 row created.
```

- ✓ يجب أن تتطابق الأعمدة والقيم المقابلة لها في نوع البيانات Data Types.
- ✓ لائحة القيم لابد أن تأخذ نفس ترتيب الأعمدة في الجدول
- ✓ المراد إضافة السجلات إليه.
- ✓ لابد من وجود علاقة واحد إلى واحد بين الأعمدة ولائحة القيم، بمعنى أنه توجد قيمة واحدة من أجل كل عمود وبنفس الترتيب.
- ✓ قد نجهل قيمة أحد الحقول، وذلك بكتابة كلمة Null، بشرط ألا يكون العمود المقابل للقيمة معرفاً بالخاصية Not Null.
- ✓ قد تسبب Insert مشكلة في التجانس و التكامل المرجعي (وسيتم تناول ذلك عند إنشاء العلاقات بين الجداول لاحقاً في هذا الفصل).
- ✓ لابد من وضع الأحرف النصية والتاريخ بين علامتي تنصيص مفردة' '.

```
INSERT INTO table([(column [, column...])
Select ([(column [, column...])])
[where conditions]);
```

للتأكد من تنفيذ العبارة السابقة وإضافة السجل إلى جدول Emp قم بتنفيذ العبارة التالية:

```
Select *    From Emp;
```

7.2. اضافة عدة سجلات إلى الجدول:

نلاحظ أنه ليس من المنطقي أن تقوم بإدخال 20,000 سجلاً مثلاً في أحد الجداول سجلاً سجلاً باستخدام الصيغة الأولى من عبارة Insert. لإضافة عدة سجلات في نفس الوقت تمكننا الصيغة الثانية من عبارة Insert بإضافة عدة سجلات إلى جدولنا مختارة من جدول آخر، والصيغة العامة لها هي:

مما سبق نجد أن الصيغة العامة لعبارة Insert تتألف مما يلي:

1. Insert, Into هي كلمات محجوزة في SQL.
2. Table ويشير إلى اسم جدول وحيد في قاعدة البيانات، وبالتحديد الجدول الذي نريد إدراج السجلات فيه.
3. Column: وتشير إلى أعمدة الجدول.
4. عبارة Select: هي عبارة Select عادية تستخدم لانتخاب الأعمدة التي ستنسخ محتوياتها إلى الجدول المراد إضافة السجلات إليه. على سبيل المثال نريد إضافة سجلات أمان إلى الجدول Secemp من أجل كل سجل من سجلات جدول Emp كما يلي:

```
INSERT INTO secemp (empno,name,job,mgr,sal,comm,deptno)
(Select (mpno,name,job,mgr,sal,comm,deptno)
 From emp;
```

8. حذف بيانات موجودة في قاعدة البيانات

(Deleting Record)

نستخدم عبارة Delete لحذف سجلات من جدول ما، وتأخذ الصيغة العامة لعبارة Delete الشكل التالي:

```
DELETE [FROM]  table
[WHERE      condition];
```

مما سبق نجد أن الصيغة العامة لعبارة Delete تتألف مما يلي:

1. Delete: هي أحد الكلمات المحجوزة في SQL.
2. From: هي أحد الكلمات المحجوزة في SQL ويليها اسم الجدول المراد حذف السجلات منه.
3. Where: هي أحد الكلمات المحجوزة في SQL ويليها شرط حذف السجلات، وهي عبارة اختيارية للحد من عدد السجلات المحذوفة، فإذا تم حذف عبارة Where من جملة Delete فإن هذا يعني حذف كافة السجلات.

8.1. حذف سجلات محددة

(Deleting Sepecific Record)

لحذف السجلات الموظفين ذوي المرتبات أكبر من 5000 من جدول Emp يتم كتابة العبارة التالية:

```
Delete From      Emp Where Sal > 5000;
```

8.2. حذف كافة السجلات (Deleting all Record)

يمكننا حذف كافة السجلات من جدول ما بحذف عبارة Where من جملة Delete. هذا يعني حذف كافة السجلات ، والصيغة العامة لها هي:

```
DELETE [FROM]  table;
```

لحذف كافة السجلات من جدول Emp يتم كتابة العبارة التالية:

```
Delete From Emp;
```

8.3. استخدام الاستعلام الفرعي في جملة Delete (Sub Queries In Delete Statements)

يمكننا استخدام الاستعلام الفرعي في جملة Delete، وذلك لحذف سجلات من جدول بناء على قيم من جدول آخر، فعلي سبيل المثال إذا أردنا حذف سجلات الموظفين التابعين لإدارة المبيعات يتم ذلك على النحو التالي

```
DELETE FROM   employee
WHERE         deptno =
                  (SELECT   deptno
                   FROM     dept
                   WHERE    dname='sales');
5 rows deleted.
```

باستخدام Delete نستطيع حذف بيانات جدول واحد فقط في نفس الوقت.

- ✓ تحذف Delete جميع السجلات التي تحقق الشرط في جملة Where.
- ✓ لا تحذف Delete الجدول.
- ✓ تحذف السجلات من الجدول بشكل مستمر، إلا أن الحذف الحقيقي للسجلات يعتمد على معالجة Commit المستخدمة التي سيتم تناولها في هذا الفصل
- ✓ قد تسبب Deleteمشكلة في التجانس والتكامل المرجعي (وسيتم تناول ذلك عند إنشاء العلاقات بين الجداول لاحقا في هذا الفصل).
- ✓ تحذف Delete السجل أو السجلات بالكامل فلا حاجة للإشارة إلى أي عمود إلا في جملة Whereعند استخدامها.

9. تعديل البيانات الموجودة في قاعدة البيانات(Using The Update Command)

يمكننا استخدام Update لتعديل القيم الموجودة في جدول ما، والصيغة العامة لعبارة Update تأخذ الشكل التالي:
مما سبق نجد أن الصيغة العامة لعبارة Update تتألف مما يلي:

```
UPDATE        table
SET column = value [, column = value]
[WHERE        condition];
```

1. Update: من الكلمات المحجوزة في, وهو أمر بتحديث البيانات فيSQL.

2. Table: ويشير إلى اسم الجدول المراد التعديل في بياناته.
3. Set: وتحتوي على سلسلة من عمليات التحديث، حيث نقوم بوضع اسم العمود الذي نقوم بتعديل بياناته كمعامل أول ، ويحتوي المعامل الثاني على القيمة الجديدة ، ويفصل بين المعاملين إشارة المساواة.
4. Where: وهي عبارة اختيارية يتم استخدامها لتحديد السجلات التي يراد التعديل بها ، فإذا تم حذف عبارة Where فإن التعديل سيتم على كل سجلات الجدول.

9.1. تحديث كافة السجلات (Update All Record)

يمكننا تحديث كافة السجلات من جدول ما بحذف عبارة Where من جملة Update فإن هذا يعني تحديث كافة السجلات، والصيغة العامة لها هي:

لتحديث كافة السجلات من جدول Emp وذلك بوضع المرتب 3000 لكل الموظفين يتم كتابة العبارة التالية:

```
UPDATE      table
SET         column = value [, column = value];
```

```
UPDATE    employee
SET       sal = 3000;
14 rows updated.
```

9.2. تحديث سجلات محددة (Update Specific Record)

يمكننا تحديث سجلات محددة والتعديل فيها بناءً على شرط أو عدة شروط، وذلك بتضمين عبارة Update فقرة Where. على سبيل المثال لتعديل مرتب الموظف رقم 555 من 3000 إلى 6000 يتم ذلك على النحو التالي:

```
UPDATE      emp
SET    sal = 6000
WHERE       empno = 555;
```

9.3. استخدام الاستعلام الفرعي في جملة Update (Sub Queries In Update Statements)

يمكننا استخدام الاستعلام الفرعي في جملة Update ، على سبيل المثال إذا أردنا تعديل أو تحديث كلاً من الوظيفة والإدارة للموظف رقم 555 لتماثل الوظيفة والإدارة للموظف رقم 7489، سوف نحتاج إلى استخدام استعلام فرعي، ويتم كتابة عبارة Update على النحو التالي:

```
UPDATE emp
SET    (job, deptno) =
                    (SELECT job, deptno
               FROM    emp
               WHERE   empno =7489)
WHERE   empno = 555;
1 row updated.
```

10. استخدام تعليمات التثبيت Commit والتراجع Rollback

يتطلب تنفيذ التغييرات في بيانات الجداول خطوتين:

1) تنفيذ إحدى العبارات التالية Insert, Update, Delete.
2) تأكيد التغيير باستخدام Commit.

عندما يتم تنفيذ عبارة من عبارات DML، يقوم نظام قواعد البيانات أوراكل Oracle مثلا بتغيير الجدول وحفظ نسخة من السجلات قبل إجراء التعديل، وسيبدو الأمر للمستخدم وكأن التغييرات قد تمت بشكل نهائي ودائم. إذا قام المستخدم بالاستعلام في قاعدة البيانات، فإنه سيرى التعديلات موجودة. وإذا قام مستخدم آخر بالاستعلام في قاعدة البيانات فلن يرى تلك التعديلات، وذلك لأن التعديلات على البيانات لا تصبح مرئية من قبل المستخدمين الذين يسمح لهم بالوصول إلى تلك الجداول، ولا تصبح دائمة حتى يتم تنفيذ تعليمة Commit، بمعنى أن تلك الخاصية تسمح للمستخدم بالتراجع عن أي تغيير قام به. يتم التراجع عن التعديلات باستخدام تعليمة Rollback، حيث تعيد هذه التعليمة قاعدة البيانات إلى الحالة التي كانت عليها بعد تنفيذ آخر تعليمة Commit.

- ✓ يتم تنفيذ تعليمة Commit بشكل ضمني إذا قطع المطور الاتصال مع نظام ادارة قواعد البيانات بشكل نظامي أو نفذ تعليمة من تعليمات DDL، حيث يؤدي ذلك إلى حفظ التغييرات.
- ✓ إذا قُطع الاتصال مع نظام قاعدة البيانات بشكل غير نظامي قبل تنفيذ تعليمة Commit لن يتم حفظ التغييرات،

```
SAVEPOINT save_point_name
ROLLBACK save_point_name
```

وستعود قاعدة البيانات إلى النقطة التي تم عندها تنفيذ آخر تعليمة Commit.

- ✓ يمكن التراجع عن التغييرات بشكل جزئي، حيث تسمح تعليمة Savepoint
- ✓ للمستخدم حفظ التغييرات حتى جزء محدد من التغييرات. على سبيل المثال نفترض أننا قمنا بمجموعة من التعديلات على مجموعة من السجلات، ولكننا غير متأكدين من أننا أجرينا التعديلات بشكل صحيح. في حال رغبنا في رؤية السجلات قبل تثبيت هذه التغييرات، يمكننا استخدام عدة تعليمات Savepoint، وبالتالي يمكننا التراجع عن التغييرات حتى أي نقطة من النقاط العلام الموضوعة كما هو موضح بالشكل التالي، والصيغة العامة لكلاًمن نقطة العلام Savepoint والتراجع Rollback إلى نقطة علام تأخذ الشكل التالي:

```
UPDATE...
SAVEPOINT jicc_ update;
Savepoint created.
INSERT...
ROLLBACK TO jicc_ update;
Rollback complete.
```

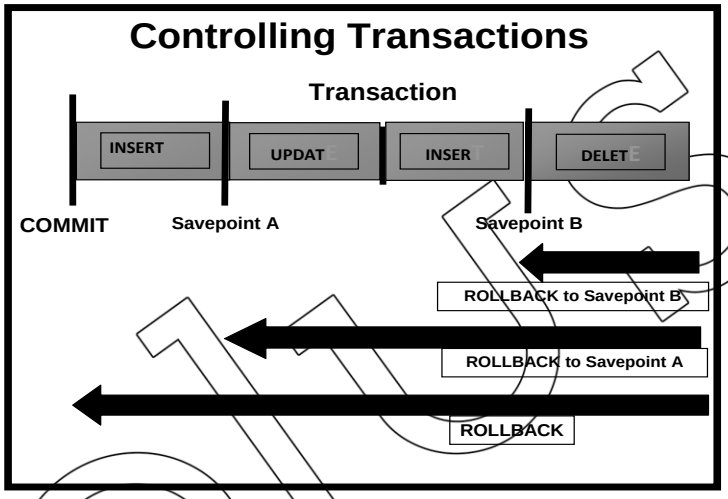
لتحديث كافة السجلات من جدول Emp وذلك بوضع المرتب 3000 لكل الموظفين يتم كتابة العبارة التالية:

```
UPDATE    employee
SET       sal = 3000;
14 rows updated.
```

لحفظ التعديلات في قاعدة البيانات يتم كتابة العبارة التاليةCOMMIT

لحذف كافة السجلات من جدول Emp ثم التراجع عن ذلك يتم بكتابة العبارات التالية:

```
Delete From emp;
14 rows deleted.
ROLLBACK;
Rollback complete.
```



11. إنشاء وتعديل وإسقاط الجداول

كيفية إنشاء جداول وإضافة محددات من نوع (Primary Key, Foreign Key)

مما سبق نجد أنه يوجد في لغة SQL ثلاث عبارات تسمي بعبارات تعريف البيانات وهي تسمح لنا بتعريف ومحي وتعديل الجداول في قاعدة البيانات، وهذه العبارات هي:

1. Create: تستخدم لتعريف جدول جديد في قاعدة البيانات.
2. Drop: تستخدم لحذف جدول موجود من قاعدة البيانات.
3. Alter: تستخدم لتغيير بنية الجدول الموجود.

قواعد البيانات العلائقية قادرة على تنفيذ كافة عبارات SQL في أي وقت بشرط أن يكون له الحق في تنفيذ تلك الأوامر والعبارات، فلكي تقوم بإنشاء وحذف وتعديل الجداول لابد من أن تتأكد من أن مدير قاعدة البيانات يمنحك حق تعريف وحذف وتعديل الجداول.

11.1. إنشاء الجداول (Create Table)

تمكننا عبارة Create من إنشاء الجداول في SQL، وتأخذ عبارة Create الشكل التالي:

```
CREATE TABLE [schema.] Table
        (column datatype [DEFAULT expr]);
```

مما سبق نجد أن الصيغة العامة لعبارة Create تتألف مما يلي:

1- Create: من الكلمات المحجوزة في SQL،وتخبر DBMS بإنشاء كائن في قاعدة البيانات.

2- Table: من الكلمات المحجوزة في SQL، وتخبر DBMS بنوع الكائن المراد إنشائه في قاعدة البيانات.

3- Table Name: ويستخدم لتحديد اسم الجدول، ويجب أن يبدأ اسم الجدول بحرف نصي ويمكن أن يحتوي أحرف وأرقام والشرطة ،ويجب ألا يتجاوز طوله 30 حرفاً ،ولايسمح بالفراغات، ولكن يسمح بالشرطة السفلية (Underscore).

4- Column Names: وتستخدم لتحديد أسماء الأعمدة المكونة للجدول المراد إنشائه، وتنطبق نفس القواعد الخاصة بالتسمية على اسم الجدول على أسماء الأعمدة، ويجب أن تحاط مجموعة تعريف الأعمدة بين قوسين.

5- Column Data Type: يحتوي تعريف الأعمدة على نوع البيانات، وطولها ودقتها كما هي موضح بالجدول اللاحق.

6- Constraints: وهي إعدادات اختيارية يمكن استخدامها لتمثل قيود للمحافظة على تكامل قاعدة البيانات.

هناك عدة اعتبارات يجب مراعاتها عند إنشاء الجداول وهي كالتالي:

1- يجب وضع أقواس تضم أسماء الأعمدة ونوع بياناتها.

2- يتم وضع فاصلة (,) بين تعريف أي عمود وأخر.

3- يجب أن يكون اسم العمود فريداً داخل الجدول.

4- لايمكن استخدام الكلمات المحجوزة كأسماء أعمدة في الجدول.

5- يتم استخدام الفاصلة المنقوطة لإخبار SQL أن الجملة انتهت وجاهزة للتنفيذ..

6- الجدول التالي يوضح لنا أهم أنواع البيانات التي يمكن أن نستخدمها عند إنشاء الجداول في SQL.

نوع البيانات	الوصف
Char(N)	يعرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (N) حرف، وجميع الحقول متساوية في الطول، حيث N=<255.
Varchar(N)	يعرف هذا النوع من البيانات عمود من نوع البيانات الحرفية بطول (N) حرف، وتختلف الحقول في الطول وتتجاوز 255.
Number(N)	يعرف عمود من نوع بيانات من نوع رقم، مع حد أقصى لعدد الأرقام (N)، ويمكن أن تأخذ (N) كحد أقصى 105.
Number(N,D)	يعرف عمود من نوع البيانات Number، ويحتوي عدد أرقام قدره (N)، والتي تتضمن القيمة (D) التي توضح عدد الأرقام العشرية بعد الفاصلة.
Integer	يشبه نوع البيانات Number، ولكنه لا يأخذ غير القيم الرقمية الصحيحة، ولا يقبل أي أرقام عشرية بعد الفاصلة.
Long	يعرف عمود من نوع البيانات الحرفية، وأقصى طول له 65.535، ولايمكن تعريف سوى عمود واحد من هذا النوع في الجدول، ولايمكن استخدام هذا العمود ضمن عبارات Where أو في الاستعلامات الفرعية أو الدوال أوالتعابير أو ضمن الفهارس.
Raw(N)	يعرف هذا النوع عمود يحتوي سجل بيانات بطول قدره N
Smallint	يعرف عمود يخزن أرقاماً موجبة وسالبة صغيرة تقع في النطاق (-32, 767 ،+32, 767)
Date	يعرف عمود نوع بياناته تاريخ.

في العبارة التالية نقوم بإنشاء جدول Department، يوضح أسماء أعمدته ونوع بياناتها.

```
CREATE TABLE department
      (deptno      NUMBER (2) not null,
            dname       VARCHAR2 (15),
            loc     VARCHAR2 (20));
Table created.
```

لعرض مواصفات وتعريفات الأعمدة نستخدم عبارة Describe أو اختصاراً Desc وتأخذ الشكل التالي:

```
Describe Table Name;
```

11.2. إنشاء الجداول باستخدام الاستعلام الفرعي

يمكنك إنشاء جدول باستخدام الاستعلام الفرعي كما في المثال التالي:

نلاحظ وجود كلمة As بين اسم الجدول الجديد المراد إنشائه والاستعلام الفرعي. للتأكد من مواصفات وتعريفات الأعمدة Column Definitions نستخدم الأمر Describe كمايلي:

```
CREATE TABLE dept10
AS
SELECT      empno, ename, sal*12 ANNSAL,hiredate
FROM        emp
WHERE       deptno = 10;
Table created.
```

```
Describe Dept10 ;
```

11.3. تعديل الجدول (Altering Table)

يُستخدم Alter Table في تعديل بنية جدول موجود وتغيير محددات الجدول بعد إنشائه. فيما يلي الخيارات التي يمكن استخدامها مع الأمر Alter:

```
ALTER TABLE table
   ADD      (column datatype [DEFAULT expr]
              [, column datatype]...)
```

1) Add: يستخدم هذا الخيار لإضافة أعمدة جديدة أو شروط إلى الجدول و يأخذ الشكل التالي:

```
ALTER TABLE dept10
ADD         (job VARCHAR2(9));
Table altered.
```

2) Modify: يستخدم لتغيير محددات موجودة ويأخذ الشكل التالي:

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
              [, column datatype]...);
```

3) Disable: يُستخدم لإلغاء تفعيل شرط ما في الجدول.
4) Enable: يُستخدم لإعادة تفعيل شرط كنا قد قمنا بإلغاء تفعيله.
5) Drop: يُستخدم لإزالة شرط ما بشكل دائم من الجدول.
6) Set Unused: يُستخدم لتحديد عمود أو أكثر كأعمدة غير مستخدمة، بحيث يمكن حذف تلك الأعمدة من الجدول عندما تصبح مصادر النظام ملائمة لذلك.
7) Drop Column: يُستخدم لإزالة عمود ما من الجدول.
8) للتعديل في نوع البيانات Dala Type الخاصة بعمود الـ Ename في جدول Dept30 نقوم بكتابة العبارة التالية:

```
ALTER TABLE    dept30
MODIFY         (ename VARCHAR2(15));
Table altered.
```

11.4. حذف جدول (Dropping Table)

```
DROP TABLE table_name;
```

تستخدم SQL عبارة Drop لحذف الجداول من النظام. كل ما نحتاجه لحذف جدول هو أن نكتب اسمه بعد Drop Table فيتم حذف الجدول نهائياً بجميع سجلاته وفهارسه، ولايمنحنا النظام فرصة لإستعادة الجدول. تأخذ جملة Drop الشكل التالي:

لحذف جدول Dept10 نقوم بكتابة العبارة التالية:

```
Drop Table Dept10;
```

12. المحافظة على تكامل قاعدة البيانات

12.1. إضافة محددات إلى الجدول من نوع (Primary Key, Foreign Key)

يوجد في نظام ادارة قواعد البيانات بعض الأدوات لحماية قاعدة البيانات. تسمى هذه الأدوات بالشروط Constraints. تقوم هذه الشروط بعدة وظائف هامة منها:

1) التأكد من أن المفتاح الأساسي أو الرئيسي وحيد Unique.
2) فرض التكامل المرجعي: بمعنى التأكد من أن السجلات الأبناء الموجودة في جداول مرتبطة، تمتلك سجل أب.
3) التأكد من أن الأعمدة المرتبطة بالشروط تحتوي دائما قيمة بداخلها Not Null.
4) التأكد من وضع القيمة الافتراضية Default Value في عمود ما.
5) التأكد من أن عموداً ما يحتوي على قيمة ما، وأن هذه القيمة موجودة ضمن نطاق محدد من القيم The Check Constraint.

12.2. الشرط Check (فحص وجود قيمة)

يستخدم هذا الشرط للتأكد من أن قيمة ما لعمود ما تقع بين قيم مجموعة محددة. عندما نستخدم الشرط Check سيقوم نظام ادارة قواعد البيانات بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في المجموعة.

في المثال التالي تم استخدام الشرط Check في تعريف جدول Employee. كما تم إعطاء اسم للشرط Check وهو (Gender_Validation).

تسمية الشرط يكون اختيارياً، وإذا لم يتم تسمية الشرط فإن نظام ادارة قواعد البيانات سيولد إسماً خاصاً للشرط.

نلاحظ أن نظام قواعد البيانات أوراكل Oracle مثلاً سيقوم بمقارنة أية قيمة يتم إدخالها مع مجموعة القيم المحددة في المجموعة ('M','F') فقط في العمودGender بمعنى ألا يسمح إلا بهاتين القيمتين فقط.

```
CREATE TABLE employee
        (emp_id         NUMBER (2) not null,
                name    VARCHAR2 (15),
                gender char (1)
Constraint gender_validation check (gender in('M','F')));
Table created.
```

12.2.1. خيار القيمة الافتراضية (Default Option)

يستخدم هذا الخيار Default لإعطاء قيمة افتراضية لعمود ما عندما لايتم وضع قيمة في هذا العمود، ولا يمنع هذا الخيار من جعل القيمة Null في هذا العمود.

في هذا المثال تم استخدام الخيار Default للتأكد من أن العمود Comm تم إعطائه القيمة صفر عند عدم وضع أي قيمة له. يكون ذلك مفيداً في معالجة القيم الفارغة Null عند إجراء العمليات الحسابية.

```
CREATE TABLE employee
        (emp_id         NUMBER (2) not null,
                name    VARCHAR2 (15),
                gender          char (1)
         sal    number
         comm   default (0),
Constraint gender_validation check (gender in('M','F')));
Table created.
```

12.2.2. الشرط Not Null (ليس فارغاً)

نستخدم الشرط Not Null للتأكد من أن العمود دوماً سيحتوي قيمة بداخله، وخاصة ما يتم وضع هذا الشرط على عمود أو أعمدة المفتاح الأساسي للجدول ، لكي تقوم بوضع الشرط Not Null على عمود ما ،قم بوضع العبارة Not Null بعد نوع بيانات العمود كما في المثال التالي:

```
CREATE TABLE employee
        (emp_id         NUMBER (2) not null,
                name    VARCHAR2 (15),
                gender char (1)
        sal     number
        comm    default (0),
Constraint gender_validation check (gender in('M','F')));
Table created.
```

5-2-1-4 الشرط Unique (وحيد)

يستخدم الشرط Unique للتأكد من أن القيمة الموجودة بالعمود المشروط هي قيمة وحيدة في جميع سجلات الجدول. هذا يعني عدم السماح بتكرار القيم، حيث يقوم هذا الشرط بعمله عبر إنشاء فهرس وحيد (Unique Index) على هذا العمود، يعتبر الشرط Unique أداة جيدة للمحافظة على هذه الخاصية في قاعدة البيانات. على سبيل المثال يمكن اعتبار عمود الضمان الإجتماعي "Number_Social_Security" الموجود في جدول Employee لا يحتوي سوى قيمة وحيدة، مع العلم بأن العمود " Emp_Id " هو المفتاح الأساسي في الجدول. أي أنه لابد أن يحتوي قيماً فريدة ووحيدة ولايسمح بالتكرار كما هو موضح كالتالي:

```
CREATE TABLE employee
        (emp_id         NUMBER (2) not null
constraint unique_emp_id unique,
                name    VARCHAR2 (15),
                gender char (1)
Constraint gender_validation check (gender in('M','F')));
Table created.
```

12.3. الشرط Primary Key (المفتاح الأساسي)

يستخدم الشرط Primary Key للمحافظة على تكامل عمود أو أعمدة المفتاح الأولي، حيث يجعل هذا الشرط القيم الموجودة في المفتاح في عمود مشروط به تتمتع بالشرطين Unique, Not Null معاً. عند تعريف هذا الشرط يتم إنشاء فهرس وحيد Unique Index ضمني وإنشاء شرط Not Null ضمني أيضاً على العمود أو الأعمدة المشروطة بشرط المفتاح الأساسي. يمكن اعتبار تعريف هذا الشرط كجزء من تعريف العمود أو جزء من تعريف الجدول، حيث إذا تم تعريف هذا الشرط على عدة أعمدة (حالة مفتاح أساسي مركب) فإن تعريف هذا الشرط سيكون جزءاً من تعريف الجدول.

لكي نعرف الشرط Primary Key كجزء من تعريف عمود ما،

نقوم بكتابة العبارة Primary Key بعد نوع بيانات الجدول كما في العبارة التالية:

```
CREATE TABLE employee
        (emp_id         NUMBER primary key,
    name        VARCHAR2 (15),
        Gender          char (1)
 Constraint gender_validation check (gender      in ('M','F')));
```

لكي نعرف الشرط Primary Key كجزء من

تعريف الجدول، نقوم بكتابة العبارة Primary Key في أسفل تعريف الجدول، ويتم تعريفه بعد تعريف آخر عمود. يجب وضع فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الأساسي كما في العبارة التالية:

```
CREATE TABLE employee
        (emp_id         NUMBER not null,
    F_name     VARCHAR2 (25),
         L_name        VARCHAR2 (15),
   Primary key(emp_id));
```

12.4. الشرط Foreign Key (المفتاح الخارجي)

يُستخدم الشرط Foreign Key كأداة للتأكيد من أن الجداول المرتبطة لا تحتوي على سجلات غير منتمية إلى سجل أب. يحتوي الشرط Foreign Key على خيار يسمح بحذف كل السجلات الأبناء المرتبطة مع سجل الأب عند حذف سجل الأب وهذا الخيار هو On Delete Cascade. يمكننا إنشاء مفتاح الربط الخارجي كجزء من تعريف العمود عند إنشاء الجدول، حيث يتم استخدام كلمة References ويليها اسم الجدول الذي يحتوي السجل الأب وذلك يكون بعد تعريف العمود. لسنا بحاجة لذكر اسم العمود من الجدول الأب، حيث أنه إذا لم تقوم بكتابة اسم العمود الموافق من الجدول الأب يفترض نظام ادارة قواعد البيانات أنه هو المفتاح الأساسي.

في هذا المثال سنقوم بتعرف الشرط Foreign Key كجزء من تعريف عمود ما كما يلي:

```
CREATE TABLE consultant_projects
  (emp_id number references consultant,
    project_name     varchar (25),
    complete_date     date);
Table created
```

لكي نعرف الشرط Foreign Key كجزء من تعريف الجدول، نقوم بكتابة العبارة Foreign Key في أسفل تعريف الجدول متبوعة باسم العمود المطبق علية الشرط ثم كلمة Reference ثم عمود المفتاح الرئيسي بالجدول الأب، ويتم تعريفه بعد تعريف آخر عمود.

يجب وضع فاصلة بعد تعريف آخر عمود وقبل تعريف شرط المفتاح الخارجي كما في العبارة التالية:

```
CREATE TABLE emp(
        empno NUMBER(4),
        ename  VARCHAR2(10) NOT NULL,
        job      VARCHAR2(9),
        mgr     NUMBER(4),
        hiredate          DATE,
        sal      NUMBER(7,2),
        comm  NUMBER(7,2),
        deptno NUMBER(7,2) NOT NULL,
        CONSTRAINT emp_deptno_fk FOREIGN KEY(deptno)
        REFERENCES dept (deptno));
```

12.5. تعديل تعريف شرط ما (Modifying Constraints)

يمكننا إضافة شرط للجدول بعد إنشائه. كما يمكننا إزالة تلك الشروط أو إلغاء تفعيلها أو تفعيلها وذلك باستخدام Alter Table. يمتلك هذا الأمر عدة خيارات يمكن استخدامها مع الشروط وهي موضحة كالتالي:

Add (إضافة): يستخدم لإضافة شرط جديد إلى الجدول.

Modify (تعديل): يستخدم لإضافة شرط إلى عمود موجود.

Drop (حذف) : يستخدم لحذف الشرط من الجدول.

Disable (إلغاء تفعيل): يسمح بإدخال البيانات دون النظر فيما إذا كانت تحقق الشرط أم لا، ولكن مع بقاء الشرط كما هو في قاموس البيانات.

Enable (تفعيل): يقوم بالتحقق من أن جميع البيانات التي يتم إدخالها والموجودة مسبقاً تحقق الشرط.

```
ALTER TABLE     emp
  enable primary key;
 Table altered.
```

```
ALTER TABLE     EMP
ADD CONSTRAINT emp_mgr_fk
FOREIGN KEY (mgr) REFERENCES emp
(empno);
Table altered.
```

```
Alter table consultant_projects
Add primary key
(emp_id,project_name);
```

```
ALTER TABLE     emp
  drop primary key;
 Table altered.
```

```
ALTER TABLE     emp
 2 disable primary key;
 Table altered.
```

كل الشكر للدكتور ((د.فراس الحلبي)) **على المجهود الرائع في إعطاء هذا المقرر وتقديمه هذا الملخص كنوطة مساعدة في دراسة قواعد البيانات ولغة الـ** SQL

وأرجو من الله أن يكون الملخص خالي من الأخطاء التنسيقية ومراعياً من حيث الخط والحجم وعدد الأوراق تكاليف الطباعة لتناسب الجميع

Mohammad Malas