

SQL Review

نظام إدارة قواعد البيانات (DBMS):

نظام إدارة قواعد البيانات (DBMS) هو تطبيق برمجي يتفاعل مع المستخدم والتطبيقات وقاعدة البيانات نفسها لإنشاء البيانات وتحليلها. قاعدة البيانات هي مجموعة منظمة من البيانات.

يوجد نوعين من نظام إدارة قواعد البيانات:

• الـ Relational Database Management

تستخدم قواعد البيانات الجداول في تخزين البيانات و علاقاتها. و ايضاً تعتبر static, or fixed schema

• الـ Non-Relational Database Management System

هنا هي لا تلتزم بالجداول في بياناتها فهي تلجئ الى Document stores و Graph databases و Key-value stores. و تعتبر dynamic schemas

سوف نركز و نتعامل مع Relational Database Management.

ما هو نظام إدارة قواعد البيانات (RDBMS (Relational Database Management System ؟

الـ RDBMS هو أساس SQL, يتم تخزين البيانات في RDBMS على هيئة جداول .

ما هو SQL (Structured Query Language) ؟

لغة الاستعلام الهيكلية Structured Query Language أو SQL هي لغة كمبيوتر لتخزين ومعالجة واسترجاع البيانات المخزنة في قاعدة بيانات, لوصف مجموعات من البيانات.

ملاحظة: يعتبر SQL من اللغات الحساسة (sensitive).

ماذا يمكن أن تفعل SQL؟

- تنفيذ استعلامات على قاعدة بيانات
- استرداد البيانات من قاعدة البيانات
- إدراج السجلات في قاعدة البيانات
- إنشاء قواعد بيانات جديدة , جداول جديدة في قاعدة بيانات
- وغيرها ...

SQL Syntax:

جداول قواعد البيانات :

غالباً تحتوي قاعدة البيانات على واحد أو أكثر من الجداول , يتم تعيين لكل جدول اسم فريد , الجدول يتكون من سجلات (records) وهي الصفوف و المجالات (fields) وهي الأعمدة .

Table				
Field				
id	ISSN-L	ISSNs	PublisherId	Journal_Title
0	2056-9890	2056-9890	1	Acta Crystallographica Section E Crystallographic Communications
1	2077-0472	2077-0472	2	Agriculture
2	2073-4395	2073-4395	2	Agronomy
3	2076-2615	2076-2615	2	Animals
4	2076-3417	2076-3417	2	Applied Sciences
5	2306-5354	2306-5354	2	Bioengineering
6	2079-7737	2079-7737	2	Bioinformatics
7	2079-6374	2079-6374	2	Biochemistry

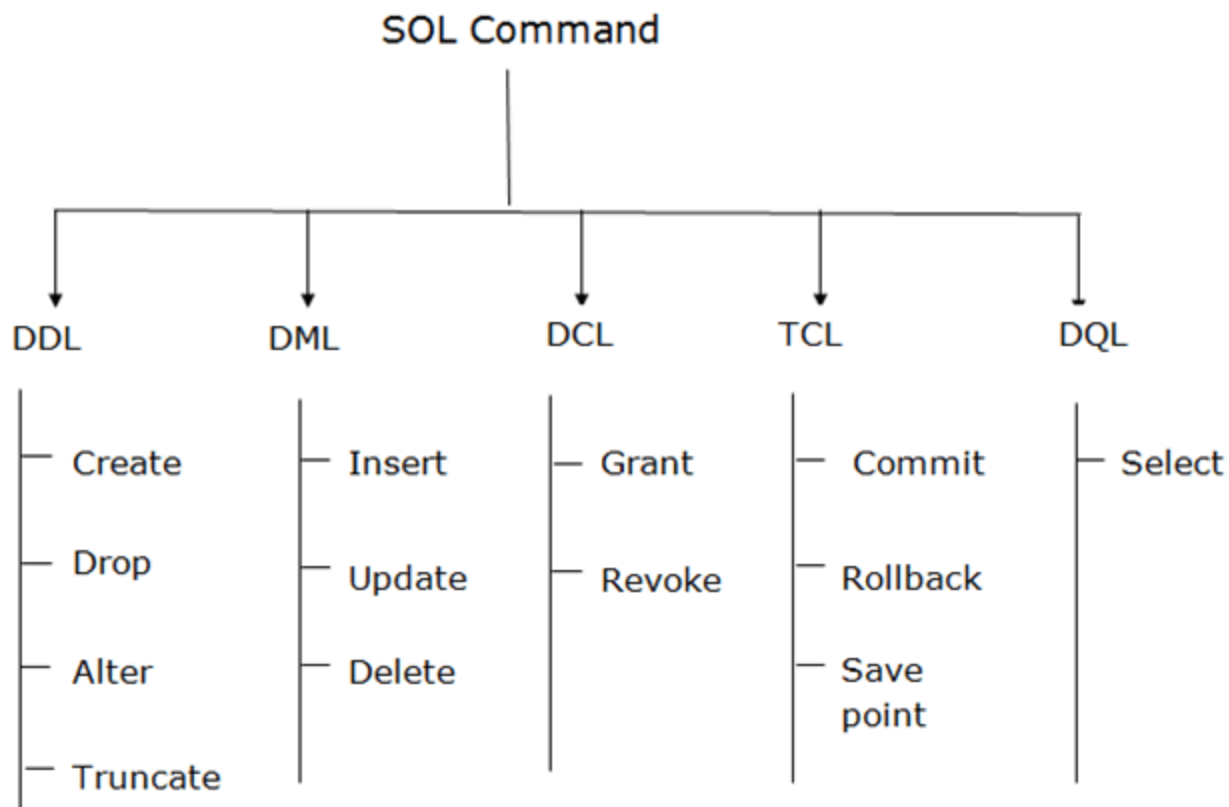
Record

Value

Database design – Library Carpentry: SQL

اهم أوامر SQL:

1. الـ DDL: لغة تعريف البيانات - Data Definition Language
2. الـ DML: لغة معالجة البيانات - Data Manipulation Language
3. الـ DCL: لغة التحكم في البيانات - Data Control Language
4. الـ DQL: استعلام البيانات - Data Query Language
5. الـ TCL: لغة التحكم في المعاملات - Transaction Control Language



SQL Commands: DDL, DML, DCL, TCL, DQL - javatpoint

لغة تعريف - DDL البيانات	لغة معالجة - DML البيانات	لغة التحكم في - DCL البيانات	لغة التحكم في - TCL المعاملات
إنشاء - CREATE جدول جديد في قاعدة بيانات	إضافة - INSERT بيانات جديدة إلى قاعدة بيانات	يتم استخدامه - GRANT لمنح المستخدم امتيازات الوصول إلى قاعدة البيانات	يتم استخدامه: SELECT ما بناء attribute لتحديد على الشرط الموصوف في جملة WHERE جلب بياناتها
يعدل بنية - ALTER جدول موجود	تعديل - UPDATE البيانات الموجودة في قاعدة البيانات	يتم - REVOKE استخدامه لاستعادة الأذونات من المستخدم	
يحذف جدولاً من - DROP قاعدة بيانات	يزيل - DELETE البيانات من قاعدة البيانات		
يحذف - TRUNCATE جميع الصفوف من الجدول			

التعامل مع قواعد البيانات (Database):

- إنشاء قاعدة بيانات

يتم إنشاء قاعدة باستخدام أمر " CREATE DATABASE "

```
CREATE DATABASE Students;
```

- عرض قاعدة البيانات

يتم عرض جميع قواعد البيانات باستخدام أمر "SHOW DATABASES"

```
SHOW DATABASES; #OR SHOW SCHEMAS;
```

- حذف قاعدة البيانات

يتم حذف قاعدة البيانات عن طريق أمر " DROP DATABASE "

```
DROP DATABASE Students;
```

التعامل مع الجداول (Tables):

حين نقوم بتعامل مع الجداول , وإنشائها في قاعدة البيانات لابد ان نتذكر الفرق بين

الـ " PRIMARY KEY, FOREIGN KEY ":

- الـ **PRIMARY KEY**: يحمل قيم فريدة (uniquely) لكل سجل في الجدول , ولا يمكن أن يحتوي على قيم خالية (Null values), يحتوي الجدول الواحد على PRIMARY KEY واحد فقط .
- الـ **FOREIGN KEY**: هو يشير إلى المفتاح الأساسي (PRIMARY KEY) في جدول آخر.

- إنشاء جدول:

يتم إنشاء جدول في قاعدة البيانات عن طريق أمر "CREATE TABLE "

```
CREATE TABLE Authors(  
    ID SERIAL PRIMARY KEY,  
    Name VARCHAR(70) NOT NULL,  
    Country VARCHAR(100) NOT NULL);
```

يتم كتابة الأمر ثم اسم الجدول المراد إنشائه , بداخله يتم إدراج و كتابة الأعمدة لهذا الجدول و تحديد نوع البيانات لكل عمود .

- إضافة عمود:

حين نريد إضافة عمود جديد للجدول بعد إنشائه نستخدم أمر "ALTER TABLE, ADD"

```
ALTER TABLE Authors  
ADD Age int;
```

- **تعيين الـ PK:**

أيضاً حين نريد تعيين المفتاح الرئيسي بعد إنشاء الجدول

```
ALTER TABLE Stds_info  
ADD PRIMARY KEY (Std_ID);
```

- **تعديل نوع بيانات العمود:**

في بعض الأحيان نريد تغيير نوع البيانات لأحد الأعمدة في الجدول , لكن لابد أن نحرص أن هذا التغيير لا يضر البيانات السابقة .

```
ALTER TABLE Authors  
ALTER COLUMN Name TYPE CHAR(255)
```

- **حذف عمود:**

لحذف عمود معين فقط وذلك عن طريق أمر "DROP COLUMN".

```
ALTER TABLE Authors  
DROP COLUMN Age ;
```

- **حذف جدول:**

حذف الجدول بالكامل فيتم باستخدام أمر "DROP TABLE".

```
DROP TABLE Authors;
```

التعامل مع البيانات (Data):

غالباً ما يتم التعامل مع البيانات عن طريق DATA CRUD.

- إضافة البيانات :

يتم إدراج سطر أو أسطر في الجدول عن طريق أمر "INSERT INTO, VALUES"

```
INSERT INTO Authors
(Name, Country, Age)
VALUES
('J.D. Salinger', 'USA',50),
('F. Scott. Fitzgerald', 'USA',60);
```

- عرض / قراءة البيانات :

يتم عرض محتويات الجدول و إستعراض الأعمدة جميعها بإستخدام أمر "SELECT"

```
SELECT * FROM Authors;
```

" * " هنا تعني الكل (All) إي يتم إرجاع جميع الأعمدة.

أيضاً حين نريد إستعراض بعض الأعمدة من جدول معين , نقوم بتحديد أسماء fields.

```
SELECT ID,Name FROM Authors;
```

لكن حين نريد عرض بعض البيانات تحت شرط معين نستخدم أمر "WHERE"

```
SELECT * FROM Authors
WHERE Age > 50;
```

و حين نريد الاستعلام عن البيانات ,لإرجاع قيم مميزة (مختلفة) فقط أي بدون تكرار , نستخدم أمر "DISTINCT"

```
SELECT DISTINCT Country
FROM Authors;
```

- **تعديل البيانات:**

تم تعديل البيانات التي تم تخزينها في الجدول عن طريق أمرين "UPDATE, SET"

```
UPDATE Authors  
SET Age = 45  
WHERE Name = 'Jane Austen';
```

- **حذف البيانات:**

حين نريد حذف جميع البيانات في الجدول نستخدم امر "DELETE".

```
DELETE FROM Authors;
```

لكن حين نريد سطر أو أسطر معينة تحت شرط نستخدم "WHERE"

```
DELETE FROM Authors  
WHERE ID = 5;
```

- **البحث في البيانات:**

يتم البحث عن نمط معين في عمود ما عن طريق أمر "LIKE"

```
SELECT * FROM Authors  
WHERE Name LIKE 'J%' ;
```

"%" هي تمثل جزء أو مقطع من الكلمة .

```
lower(string) LIKE 'o%';  
string LIKE '_n_';  
string LIKE '_n%';  
string LIKE '%';  
string LIKE '%\%';  
string LIKE '%\_%';
```

LIKE and ILIKE for Pattern Matching in PostgreSQL

```
SELECT * FROM Authors  
WHERE Country LIKE '__A' ;
```

"_" هي تمثل حرف فقط من الكلمة .

معاملات البيانات (Operators):

لدينا ثلاث أنواع للمعاملات وهي :

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators

• معاملات رياضية (Arithmetic Operators):

هي العمليات الحسابية مثل:

- الجمع "+"
- الطرح "-"
- الضرب "*",
- القسمة "/"
- باقي القسمة "%"

• معاملات المقارنة (Comparison Operators):

=	تساوي
>	أكبر من
<	أقل من
>=	أكبر من أو تساوي
<=	أقل من أو تساوي
<>	لا يساوي

• معاملات منطقية (Logical Operators):

تكون القيمة TRUE إذا تحققت جميع الشروط	ALL
تكون TRUE إذا كانت جميع الشروط المفصولة بـ AND صحيحة	AND
تكون TRUE إذا تحققت أي قيمة من الشرط	ANY
تكون TRUE إذا كانت القيمة ضمن نطاق المقارنات	BETWEEN
تكون TRUE في حالة إرجاع سجل أو أكثر من قيمة الاستعلام	EXISTS
تكون TRUE إذا كانت قيمة المعامل تساوي إحدى القيم	IN
يكون TRUE إذا كانت القيمة تطابق نمط	LIKE
حين لا تكون قيمة الشرط صحيح يعرض القيم , بمعنى آخر نفي الشرط	NOT
صحيحة OR إذا كانت أي من الشروط المفصولة بـ TRUE	OR
تكون القيمة TRUE في حالة استيفاء أي من قيم الاستعلام الفرعي للشرط	SOME

الدوال التجميعية (Aggregate functions):

تقوم الدالة التجميعية بإجراء عملية حسابية على مجموعة من القيم ، وإرجاع قيمة واحدة.

تقوم الدالة COUNT () بإرجاع عدد الصفوف التي تطابق معيارًا محددًا.	COUNT ()

ترجع الدالة AVG () القيمة المتوسطة لعمود رقمي.	AVG ()
ترجع الدالة SUM () المجموع الإجمالي لعمود رقمي.	SUM ()
تُرجع الدالة MIN () أصغر قيمة للعمود المحدد.	MIN ()
ترجع الدالة MAX () أكبر قيمة للعمود المحدد.	MAX ()
تجمع عبارة GROUP BY الصفوف التي لها نفس القيم من عمود و تُلخصها في صفوف.	GROUP BY
تُستخدم الكلمة الأساسية ORDER BY لفرز مجموعة النتائج بترتيب تصاعدي أو تنازلي.	ORDER BY

سوف نتضح لنا هذه المفاهيم مع الأمثلة و التطبيق:

1. سوف نقوم بإرجاع عدد الـ USA من الـ country:

```
SELECT COUNT(Country)
FROM Authors
WHERE Country = 'USA';
```

2. نريد إرجاع متوسط الـ age:

```
SELECT AVG(Age)
FROM Authors
```

3. نقوم بإرجاع مجموع الـ amount التي أقل من 500:

```
SELECT SUM(amount)
FROM Orders
WHERE amount < 500;
```

4. نريد إرجاع أصغر Author عمراً:

```
SELECT MIN(Age)
FROM Authors ;
```

5. الآن نقوم بالعكس, نريد إرجاع أكبر Author عمراً:

```
SELECT Max(Age)
FROM Authors ;
```

6. نريد عدد الـ Authors في كل بلد:

```
SELECT COUNT(ID), Country
FROM Authors
GROUP BY Country;
```

الـ output لتوضيح الفكرة:

count bigint	country character varying (100)
1	India
1	UK
5	USA

ملاحظة: غالباً ما تُستخدم جملة GROUP BY مع الدالات التجميعية (COUNT () و MAX () و MIN () و SUM () و AVG ()) لتجميع مجموعة نتائج.

7. نريد عرض بيانات الـ Authors و لكن بترتيب تصاعدي:

```
SELECT ID, Name ,Age
FROM Authors
ORDER BY Age ASC;
```

يتم استخدام **ASC** للترتيب من الاصغر الى الاكبر

8. نفس المثال السابق و لكن نريد عرضها بترتيب تنازلي:

```
SELECT ID, Name ,Age
FROM Authors
ORDER BY Age DESC;
```

يتم استخدام **DESC** للترتيب من الاكبر الى الاصغر

ملاحظة: تقوم الكلمة الأساسية ORDER BY بفرز السجلات بترتيب تصاعدي افتراضياً.

Joins in SQL

في علم البيانات تعتبر "JOIN" مهمة جداً , وهي تستخدم لدمج صفوف من جدولين أو أكثر ، بناءً على عمود مرتبط بينهما.

أنواع الـ JOIN:

- Self-join
- Inner join
- Outer join
 - Left join.
 - Right join
 - Full join
- Union
 - Union all
- Intersection
- Minus

لنتعرف على كل نوع و تطبيقه:

سنستخدم الجدولين التاليين لتطبيق أنواع مختلفة من استعلامات.

جدول الـ basket_a

	a [PK] integer		fruit_a character varying (100)
1		1	Apple
2		2	Orange
3		3	Banana
4		4	Cucumber

جدول basket_b

	b [PK] integer		fruit_b character varying (100)
1		1	Orange
2		2	Apple
3		3	watermelon
4		4	pear

• Self join:

هو join من الجدول الى نفسه (يتم ربط الجدول بنفسه), باختصار يمكننا القول إنه join بين نسختين من نفس الجدول.

o لنقوم بعرض الـ fruits بناءً على مقارنة بينهم في الجدول الاول basket_a؟

```
SELECT ba.a, ba.fruit_a, bb.fruit_a
FROM basket_a ba ,basket_a bb
WHERE ba > bb;
```

Result:

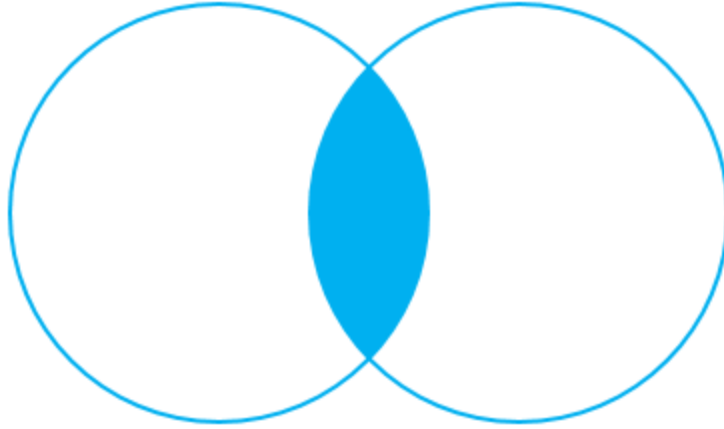
	a [PK] integer	fruit_a character varying (100)	fruit_a character varying (100)
1	2	Orange	Apple
2	3	Banana	Apple
3	3	Banana	Orange
4	4	Cucumber	Apple
5	4	Cucumber	Orange
6	4	Cucumber	Banana

الـ Inner join:

(تسمى أحياناً simple join) هو الـ join لجولين أو أكثر تُرجع فقط تلك الصفوف التي تفي بشرط الصلة (join).

o كيف نقوم بضم الجدول الأول (basket_a) بالجدول الثاني (basket_b) عن طريق مطابقة القيم الموجودة في العمودين Fruit_a و fruit_b

```
SELECT a , fruit_a , b , fruit_b
FROM basket_a
INNER JOIN basket_b
ON fruit_a = fruit_b
```



INNER JOIN

PostgreSQL Join - Inner Join

Result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange

• الـOuter join:

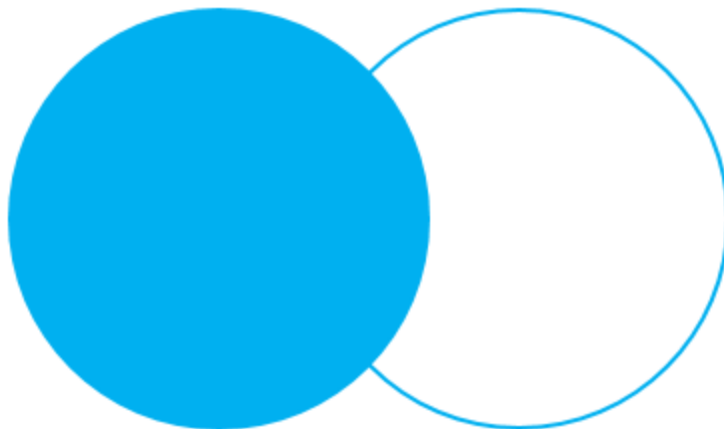
ترجع الـouter join جميع الصفوف التي تفي بشرط الـjoin وتعيد أيضاً بعض أو كل هذه الصفوف من جدول واحد لا تفي فيه بعض الصفوف الشرط.

○ الـLeft join:

إرجاع كافة السجلات من الجدول الأيسر ، والسجلات المتطابقة من الجدول الأيمن.
 ■ لنقوم بإرجاع جميع الفواكة من الجدول الأيسر و الفواكة المتطابقة من الجدول الأيمن ؟

```
SELECT a, fruit_a, b, fruit_b
FROM basket_a
LEFT JOIN basket_b
ON fruit_a = fruit_b;
```

الجدول الأول [basket_a] الجدول الأيسر ويسمى الجدول الثاني [basket_b] الجدول الأيمن.
 في left join, يبدأ في تحديد البيانات من الجدول الأيسر. يقارن القيم الموجودة في العمود Fruit_a بالقيم الموجودة في العمود Fruit_b في الجدول basket_b.



LEFT OUTER JOIN

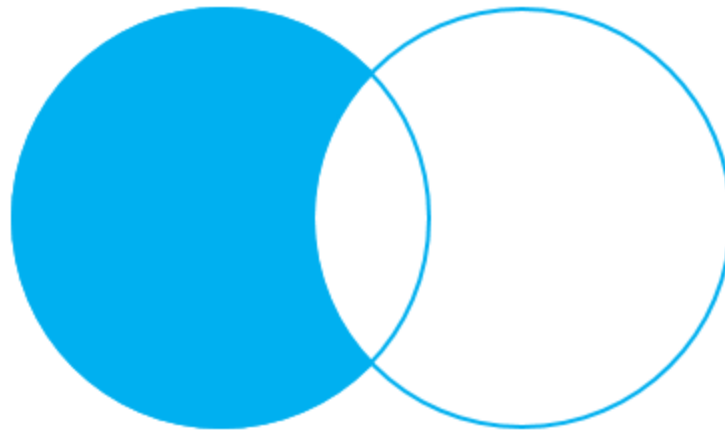
PostgreSQL Join - Left Join

Result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana	[null]	[null]
4	4	Cucumber	[null]	[null]

لنفترض اننا نريد تحديد الصفوف من الجدول الأيسر التي لا تحتوي على صفوف متطابقة في الجدول الأيمن ، ذلك يمكن باستخدام الصلة اليسرى مع عبارة WHERE:


```
SELECT a, fruit_a, b, fruit_b
FROM basket_a
LEFT JOIN basket_b
  ON fruit_a = fruit_b
WHERE b IS NULL;
```



LEFT OUTER JOIN – only
rows from the left table

PostgreSQL Join - Left Join with Where

Result:

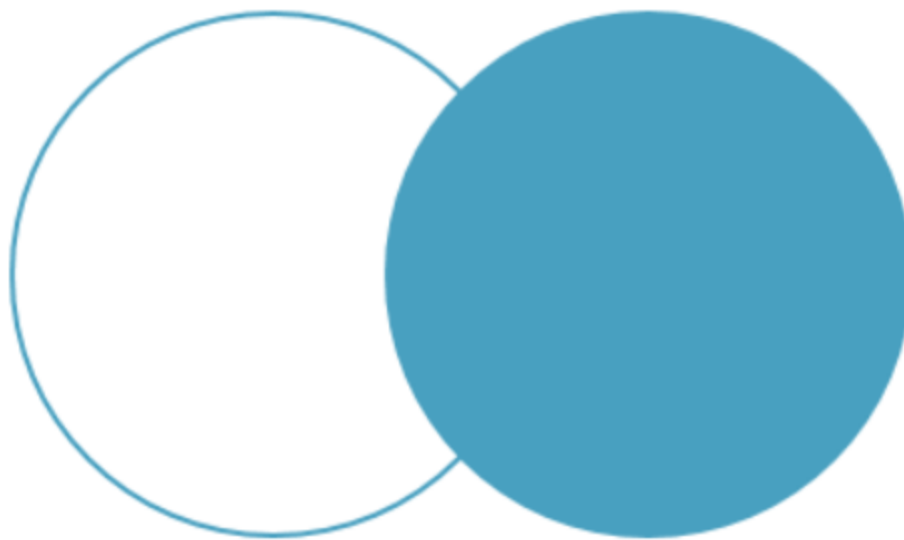
	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	3	Banana	[null]	[null]
2	4	Cucumber	[null]	[null]

○

○ **الـRight join:**

الـright join هي نسخة معكوسة من الـleft join.
إرجاع كافة السجلات من الجدول الأيمن والسجلات المتطابقة من الجدول الأيسر.

```
SELECT a, fruit_a, b, fruit_b
FROM basket_a
RIGHT JOIN basket_b ON fruit_a = fruit_b;
```



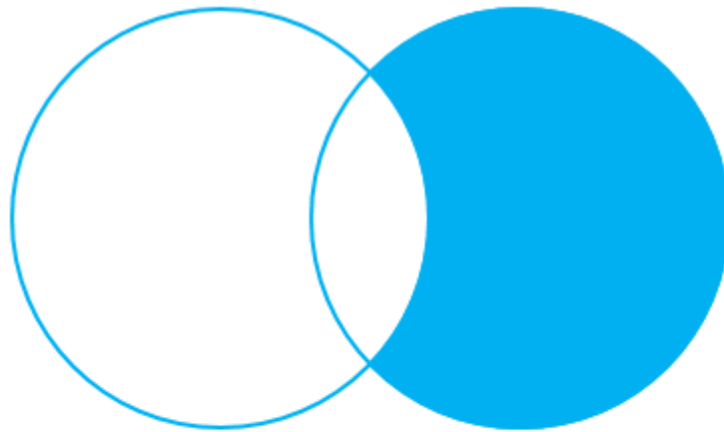
RIGHT OUTER JOIN

Result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	2	Orange	1	Orange
2	1	Apple	2	Apple
3	[null]	[null]	3	Watermelon
4	[null]	[null]	4	Pear

وبالمثل ، يمكننا الحصول على صفوف من الجدول الأيمن لا تحتوي على صفوف متطابقة من الجدول الأيسر عن طريق إضافة عبارة WHERE على النحو التالي:

```
SELECT a, fruit_a, b, fruit_b
FROM basket_a
RIGHT JOIN basket_b
    ON fruit_a = fruit_b
WHERE a IS NULL;
```



**RIGHT OUTER JOIN – only
rows from the right table**

PostgreSQL Join - Right Join with Where

Result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	[null]	[null]	3	Watermelon
2	[null]	[null]	4	Pear

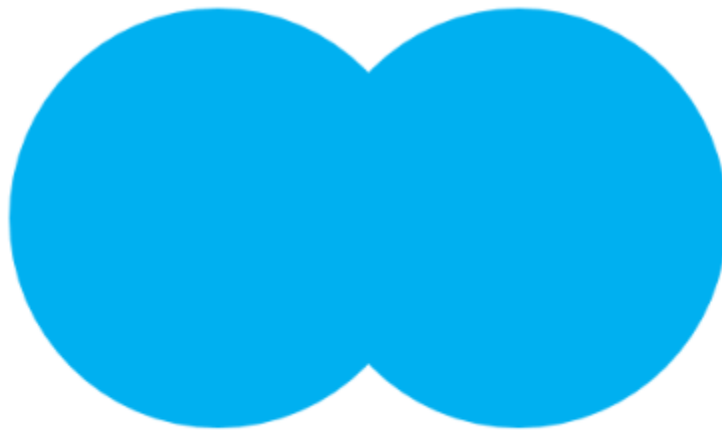
○ **الـ Full join:**

إرجاع كافة الصفوف من الجدولين، مع القيم "null" اذا لم تفي بالشرط.

```

SELECT a, fruit_a, b, fruit_b
FROM basket_a
FULL OUTER JOIN basket_b
ON fruit_a = fruit_b;

```



FULL OUTER JOIN

PostgreSQL Join - Full Outer Join

Result:

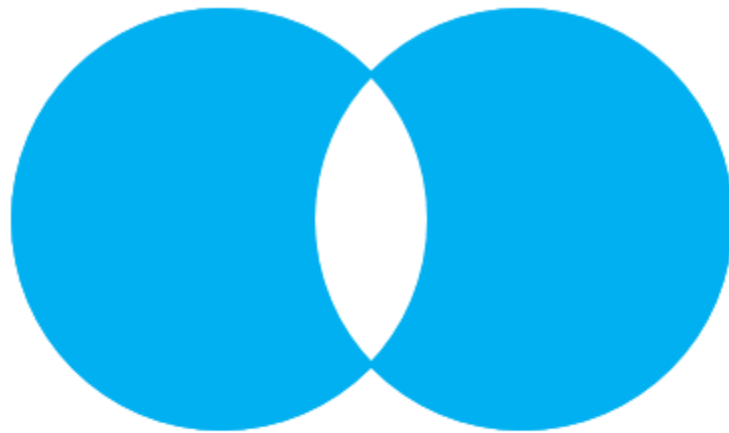
	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana	[null]	[null]
4	4	Cucumber	[null]	[null]
5	[null]	[null]	3	Watermelon
6	[null]	[null]	4	Pear

لإرجاع الصفوف في الجدول التي لا تحتوي على صفوف متطابقة في الجدول الآخر ، يمكنك استخدام full join مع عبارة WHERE مثل هذه:

```

SELECT a, fruit_a, b, fruit_b
FROM basket_a
FULL OUTER JOIN basket_b
    ON fruit_a = fruit_b
WHERE a IS NULL OR b IS NULL;

```



**FULL OUTER JOIN – only
rows unique to both tables**

PostgreSQL Join - Full Outer Join with Where

Result:

	a integer	fruit_a character varying (100)	b integer	fruit_b character varying (100)
1	3	Banana	[null]	[null]
2	4	Cucumber	[null]	[null]
3	[null]	[null]	3	Watermelon
4	[null]	[null]	4	Pear

Customers table

ID	Name	Age	Address	Salary	supervisor_ID
1	Ahmad	32	Abha	2000	null
2	Ali	25	Jeddah	1500	null
3	Fawwaz	23	Makkah	2000	1
4	Fahad	25	Abha	6500	1
5	Sultan	32	Najran	8500	1
6	Ahmad	22	Jeddah	4500	2
7	Salem	24	Riyadh	10000	2

Orders table

OID	O_DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08	3	3000
100	2009-10-08	3	1500
101	2009-11-20	2	1560
103	2008-05-20	NULL	2060

- **الـUnion:**

يتم استخدام عامل التشغيل UNION لدمج مجموعة النتائج المكونة من عبارتين SELECT أو أكثر, يزيل الصفوف المكررة بين عبارات SELECT المختلفة.
ملاحظة: يجب أن يكون لكل عبارة SELECT داخل مشغل UNION نفس عدد الحقول في مجموعات النتائج مع أنواع بيانات مماثلة.

○ لنقوم بعرض الـID (قيم مميزة فقط) من كل من جدول "Customers" و "Orders":

```
SELECT ID as CUSTOMERID
FROM CUSTOMERS
UNION
SELECT CUSTOMER_ID FROM ORDERS;
```

Result:

CUSTOMERID
1
2
3
4
5
6
7
(null)

ملاحظة: لا يمكن استخدام UNION لسرد كافة ID الـ customers من الجدولين. استخدم UNION ALL لتحديد القيم المكررة.

○ الـ Union all:

هو مشابه جداً لي Union, لكن تُرجع UNION فقط سجلاً فريداً ، بينما تُرجع UNION ALL جميع السجلات (بما في ذلك التكرارات).

```
SELECT ID as CUSTOMERID FROM CUSTOMERS
UNION ALL
SELECT CUSTOMER_ID FROM ORDERS;
```

Result:

CUSTOMERID
1
2
3
4
5
6
7
3
3
2
(null)

● الـ Intersection:

يتم استخدام عامل التشغيل INTERSECT لإرجاع نتائج 2 أو أكثر من عبارات SELECT. ومع ذلك ، فإنه يقوم فقط بإرجاع الصفوف المحددة بواسطة كافة الاستعلامات أو مجموعات البيانات ويزيل INTERSECT التكرارات.

○ نريد عرض ID الـ Customers الذين لديهم طلبات, باستخدام الـ Intersection؟

```
SELECT ID AS CUSTOMERID FROM CUSTOMERS
INTERSECT
SELECT CUSTOMER_ID FROM ORDERS;
```

Result:

CUSTOMERID
2
3

الـ Intersect Vs. Inner join:

قد نرا تشابة كبير بينهم و قد تختلط لدينا الامور, لكن الإثنان مختلفان جدا. INNER JOIN هو عامل يتطابق بشكل عام مع مجموعة محدودة من الأعمدة ويمكن أن يُرجع صفراً من الصفوف أو المزيد من الصفوف من أي جدول. INTERSECT هو عامل تشغيل قائم على مجموعة يقارن الصفوف الكاملة بين مجموعتين ولا يمكنه أبداً إرجاع صفوف أكثر من الجدول الأصغر.

• الـ Minus:

هو استعلام يستخدم عامل الطرح في SQL لطرح مجموعة نتائج واحدة من مجموعة نتائج أخرى لتقييم فرق مجموعة النتائج.

○ نريد عرض ID الـ Customers الذين لم يكن لديهم اي طلبات, باستخدام الـ Minus ؟

```
SELECT ID AS CUSTOMERID FROM CUSTOMERS
MINUS
SELECT CUSTOMER_ID FROM ORDERS;
```

Result:

CUSTOMERID
1
4
5
6
7

دورة منصة سطر:

- المستوى المبتدئ

<https://satr.codes/courses/FtkmhtUpQW/view>

 منصة سطر التعليمية • satr.codes


- مستوى متوسط

<https://satr.codes/courses/APjgdQqVWR/view>

 منصة سطر التعليمية • satr.codes

- مستوى متقدم

<https://satr.codes/courses/bOXiOFzkMv/view>

 منصة سطر التعليمية • satr.codes