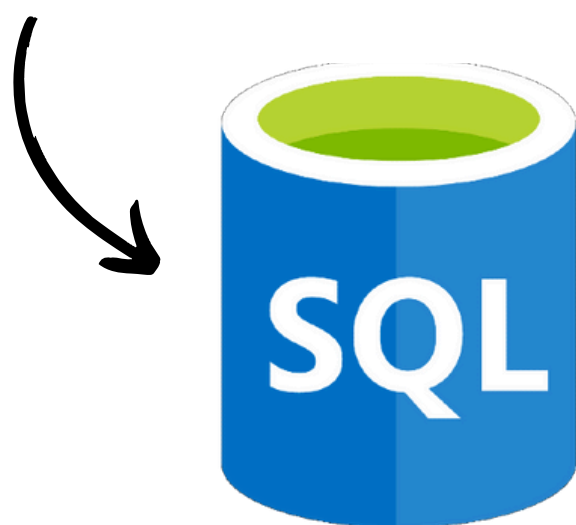


دليلك الشامل لتعلم الـ SQL



محتوى الدليل الشامل:

• **ليه تتعلم الـ SQL كمحلل بيانات - ص 3**

- 5 أسباب تخليك تتعلم SQL كمحلل بيانات؟
- 4 مهارات هتطور بسبب استخدامك للـ SQL!؟

• **أوامر الـ SQL بأمثلة عملية - ص 6**

- 6 مبادئ للـ SQL لمحلل البيانات
- 5 أمثلة عملية لأوامر الـ SQL
- 6 أمثلة عملية للـ Data Cleaning باستخدام الـ SQL
- كيف تتعامل مع التواريخ فى الـ SQL
- ازاي تستخدم الـ SQL WINDOW FUNCTIONS كمحلل بيانات؟

• **نصائح لتطوير مستواك فى الـ SQL - ص 37**

- 5 نصائح عملية لكتابة SQL Query بشكل أفضل
- 4 أسباب تخليك تراجع SQL QUERY بتاعت زمايلك؟



ليه تتعلم الـ SQL كمحلل بيانات



5 أسباب تخليك تتعلم SQL كمحلل بيانات؟

1- التطبيق العالمى

من اللغات المستخدمة عالميا فى التعامل مع قواعد البيانات فى كل الصناعات والمجالات

2- كفاءة فى التعامل مع البيانات

فى وقت قصير وبكفاءة عالية تقدر تتعامل مع مجموعة بيانات ضخمة ببساطة

3- بساطة العمليات على البيانات

ببساطة تقدر تجمع البيانات وتشكل جداول، وتعمل عمليات حسابية

4- مطلوبة فى سوق العمل

من أكثر المهارات المطلوبة لمحلل البيانات هى قدرة على التعامل مع الـ SQL

5- بتساعد محلل البيانات بشكل كبير

الـ SQL هيساعد تتعامل مع حجم داتا كبير وتعمل عمليات حسابية معقدة وده هيساعدك إنك تحلل البيانات وتقدم مقترحات للبيزنس



6 مبادئ للـ SQL لمحلل البيانات

1- حل المشاكل

هتطور مهارة حل المشاكل عندك لانك كل يوم هتواجه مشاكل فى البيانات سواء (جودة - حجم - التعقيدات) وهتستخدم الـ SQL عشان تحلها. ومهارة حل المشاكل من أهم أهم المهارات اللى لازم تبقى قوية عند محلل البيانات.

2- التفكير المنطقي

ترتيب أفكارك و الـ Actions اللى هتعملها بشكل منطقي مهارة مهمة هتطور باستخدامك للـ SQL عن طريق استخدام الـ caes statement أو aggregation أو Join.

3- معرفة الـ Function

معرفة الـ Function أو caluses واستخدامها هيساعدك تتعلم أدوات تانية لتحليل البيانات بشكل أسرع بسبب التشابه الكبير ما بين الـ Functions فى كل الـ Tools.

4- التعاون المشترك

سواء هتتعاون مع زميل معاك فى التيم أو تيم تانى عشان تكتب Query يساعدكوا فى شغلوكوا أو يسهل على التيم التانى ان البيانات تكون updated ودقيقة كل يوم.



أوامر الـ SQL بأمثلة عملية



ال 6 مبادئ للـ SQL :

- 1- تعرف أنواع الـ Joins
- 2- الفرق ما بين Where و Having
- 3- الفرق ما بين Union و Union all
- 4- استخدام Case when داخل Sum
- 5- تعرف تعمل CTE و Subquery
- 6- تعرف ترتيب تنفيذ الـ Query

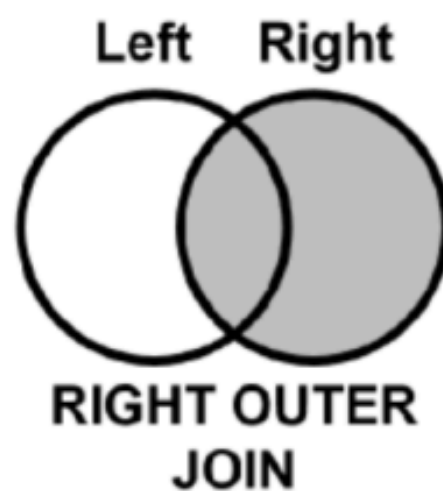
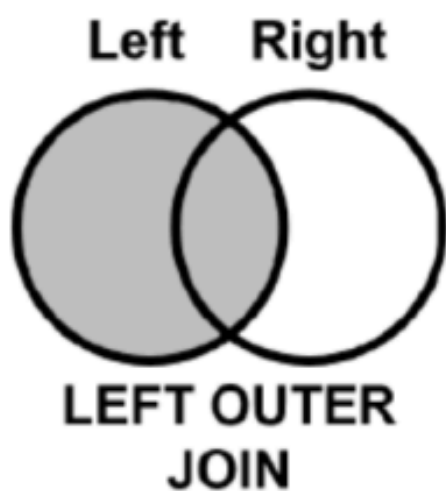
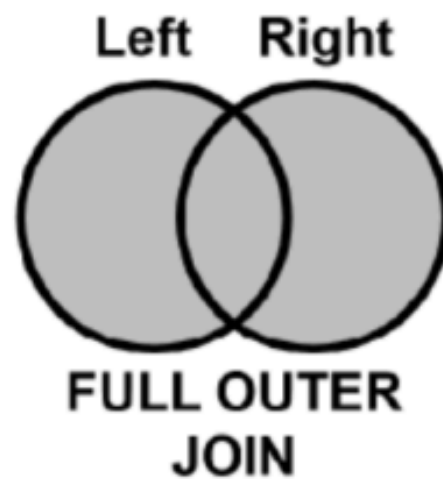
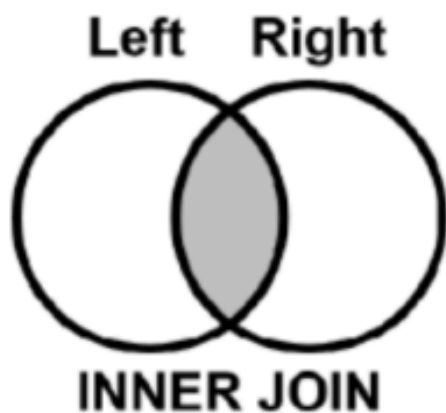


1- تعرف أنواع ال Joins

ال Joins بتستخدمها عشان تربط rows من أكثر من Table بناء على colume مشترك.

ودي صورة بتوضح ال 4 أنواع الرئيسية لل Joins:

- 1- ال Inner Join ، بيرجع ال records اللي فيها matching من ال Tables
- 1- ال Left Join ، بيرجع ال records من ال left table اللي ب match ال right table
- 1- ال Right Join ، بيرجع ال records من ال right table اللي ب match ال left table
- 1- ال Full Join ، بيرجع كل ال records اللي من ال Tables



2- الفرق ما بين Where و Having

بتستخدم ال Where أو ال Having عشان تعمل Filter للبيانات.

ولكن ال where بتعمل filter للبيانات قبل ما تم عملية ال aggregation سواء sum أو AVG مثلا. اما ال having بتعمل filter للبيانات بعد ال aggregation.

ال Having بتعمل filter لكل Group اما ال Where بتعمل filter لكل row فى ال Table

3- تعرف تعمل CTE و Subquery

الهدف النهائى من ال CTE أو ال Subquery إنك تعمل نتيجة مؤقتة تقدر تستخدمها بعد كده فى حساب table تانى.

الفرق اللى ما بينهم هو ان ال subquery بتبقى صعبة شويه فى القراءة، اما ال CTE مميز انك سهل إنك تقرأه واسهل ف ال trouble shooting



4- الفرق ما بين Union و Union All

بتستخدمها عشان تعمل Combine لـ Two Table أو أكثر.

الفرق ما بينهم إن الـ Union بتشيل الـ Duplicate rows من الـ Tables اللي بتعملها combin. أما الـ Union All بتسيب الـ Duplicate rows عادى

ID	NAME	EMAIL
1	RAJ	R@R.COM
2	MUNESH	M@M.COM

TABLE 1

ID	NAME	EMAIL
1	BAN	B@B.COM
2	MUNESH	M@M.COM

TABLE 2

UNION

ID	NAME	EMAIL
1	BAN	B@B.COM
1	MUNESH	M@M.COM
2	RAJ	R@R.COM

UNION ALL

ID	NAME	EMAIL
1	RAJ	R@R.COM
2	MUNESH	M@M.COM
1	BAN	B@B.COM
2	MUNESH	M@M.COM



5- استخدام Case when داخل Sum

هدف استخدام ال Case When داخل ال aggregation هو تعديل القيمة النهائية اللى هتظهرلك بناء على معايير أنت محتاجها.

المثال بيوضح استخدام case when داخل sum لحساب عدد المكالمات ال inbound و ال outbound

```
37
38 SELECT DATEPART(hour, Date) AS HourOfDay,
39 sum(case when Typ = 'outbound' then 1 else 0 end) as outbound,
40 sum(case when Typ = 'inbound' then 1 else 0 end) as inbound
41 from #pstntable
42 where Date >= @_StartTime and Date <= @_EndTime
43 group by datepart(hour,Date)
44
45
```

100 % <

Results Messages

	HourOfDay	outbound	inbound
1	7	0	2
2	8	5	3
3	9	0	3
4	11	0	2



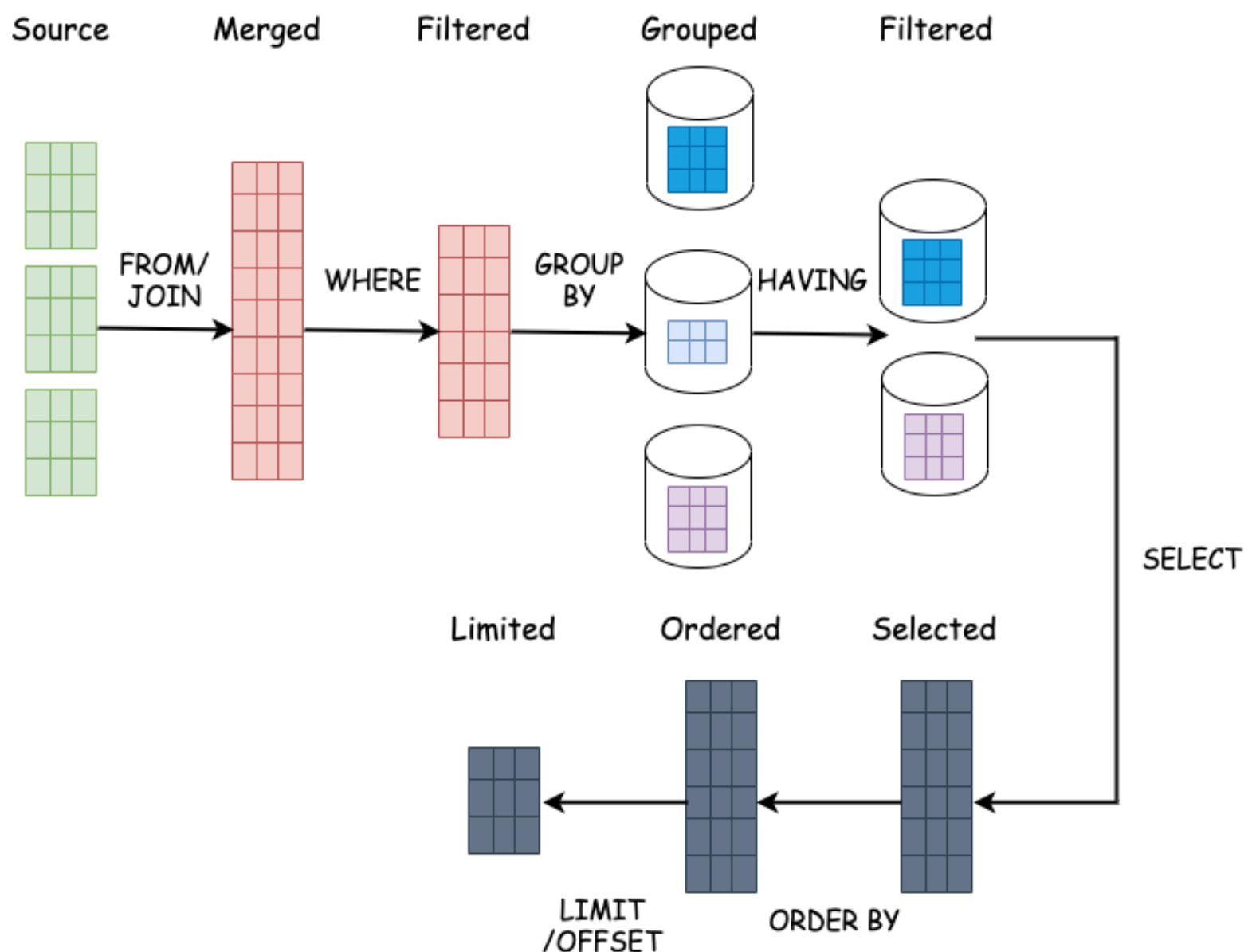
6- تعرف ترتيب تنفيذ ال Query

كلنا عارفين ال Syntax order in SQL ولكن ده الظاهر لينا !?

والترتيب اللى كلنا عارفه هو

Select, from, where, order by

الصورة بتوضح الترتيب الحقيقى لتنفيذ ال query



ال 5 أوامر للـ SQL بستخدمهم كل يوم كمحلل بيانات:

SELECT / FROM ال -1

Where ال -2

Left Join ال -3

Case When ال -4

Group by ال -5



SELECT / FROM ال 1

Select ال

المهمة الأساسية لها هي جلب البيانات عن طريق اختيار كل الأعمدة - Columns أو اعمدة معينة

From ال

الجدول الى يتم جلب منه البيانات

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT customer_id , first_name , age  
FROM Customers
```

ال Query لاختيار الأعمدة (/ customer id
first name / age) من ال table

customer_id	first_name	age
1	John	31
2	Robert	22
3	David	22
4	John	25
5	Betty	28



Where ال 2

عندنا شروط محتاجنها في البيانات، محتاجين فلتر بالشروط دي

WHERE = ''

WHERE..IN ('', '', '')

WHERE LIKE '%..%'

```
SELECT customer_id , first_name , age
FROM Customers
where age > 25
```

ال Query لاختيار الشرط - العمر أكبر
من 25 من عمود ال age

Output

customer_id	first_name	age
1	John	31
5	Betty	28



3 ال LEFT JOIN

المهمة الأساسية هو جلب البيانات من أكثر من جدول وربطهم ببعض.

جلب البيانات من الجدول اللى على اليسار سواء الجدول اللى على اليمين موجود فيه البيانات دي ولا لا

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

Orders

order_id	item	amount	customer_id
1	Keyboard	400	4
2	Mouse	300	4
3	Monitor	12000	3
4	Keyboard	400	1
5	Mousepad	250	2

```
SELECT C.customer_id , first_name , age , amount
FROM Customers as C
left join orders as O on O.customer_id = C.customer_id
```

ال Query هدفه انه يدمج ال Tables -
عن طريق ال Customer_id ونطلع منهم
ال Amount لكل Customer



3 ال LEFT JOIN

```
SELECT C.customer_id , first_name , age , amount
FROM Customers as C
left join orders as O on O.customer_id = C.customer_id
```



Output

customer_id	first_name	age	amount
1	John	31	400
2	Robert	22	250
3	David	22	12000
4	John	25	300
4	John	25	400
5	Betty	28	

النتيجة - الكمية لكل Customer والعميل اللى معندهوش قيمة
يبقى مكانها Null



4 جی CASE WHEN ... THEN ... ELSE ...END

زیہا زی IF فی الاکسیل، یعنی لما تلاقى الشرط کذا
اعمل کذا

عاوز نعمل colume ونكتب فيه Above 25 لكل ال customer عمرهم
اکبر من 25 واللى غير کده سواء اقل او يساوى 25 نكتب other

```
SELECT
customer_id ,
first_name ,
age ,
CASE WHEN age > 25 THEN 'Above 25' ELSE 'Other' END as 'Age Category'
FROM Customers
```



Output			
customer_id	first_name	age	Age Category
1	John	31	Above 25
2	Robert	22	Other
3	David	22	Other
4	John	25	Other
5	Betty	28	Above 25



5 ال GROUP BY

المهمة الأساسية هو تجميع البيانات في مجموعات أو أنواع

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT count (customer_id ),country
FROM Customers
group by country
```

ال Query هدفه تجميع عدد ال customer بالنسبة ل country

count (customer_id)	country
1	UAE
2	UK
2	USA



6 أمثلة عملية للـ Data Cleaning باستخدام الـ SQL

COALESCE الـ -1

Round الـ -2

Trim الـ -3

replace الـ -4

Concat الـ -5

Cast الـ -6



1 ال COALESCE

وظيفة ال COALESCE هو تبديل ال Null Values بقيمة أنت عاوزها

SQLite	SQLite	SQLite	
2	SELECT *		
3	FROM DataCleaning		
4			
Name	Salary		
Atef	5000		
Ali	4000		
Ahmed			

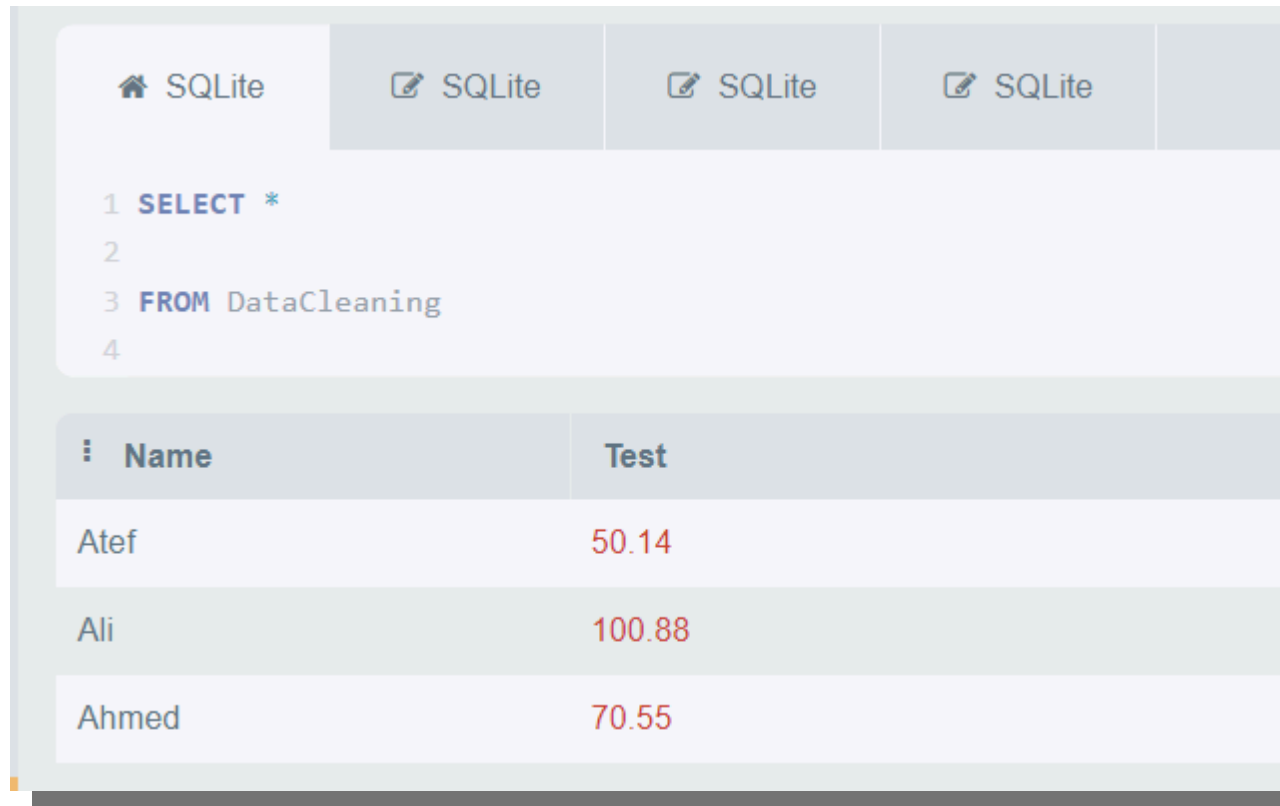
ال Query هدفه تبديل ال Null Values واللى هيا ال Salary اللى قدام Ahmed ونحط مكانه كلمة "Not Available"

SQLite	SQLite	SQLite	
1	SELECT		
2	name,		
3	COALESCE(salary, 'Not Available') AS salary		
4	FROM DataCleaning;		
Name	salary		
Atef	5000		
Ali	4000		
Ahmed	Not Available		



Round ال 2

وظيفة ال Round انها تقرب الرقم لأقرب رقم عشري أنت عاوزه



The screenshot shows the SQLite application interface. At the top, there are four tabs, each labeled 'SQLite' with a home icon. The first tab is active. Below the tabs, the SQL query is displayed: `1 SELECT *`, `2`, `3 FROM DataCleaning`, and `4`. Below the query, the results are shown in a table with two columns: 'Name' and 'Test'. The data rows are: Atef (50.14), Ali (100.88), and Ahmed (70.55).

Name	Test
Atef	50.14
Ali	100.88
Ahmed	70.55

ال Query هدفه تقرب الرقم لأقرب واحد رقم عشري



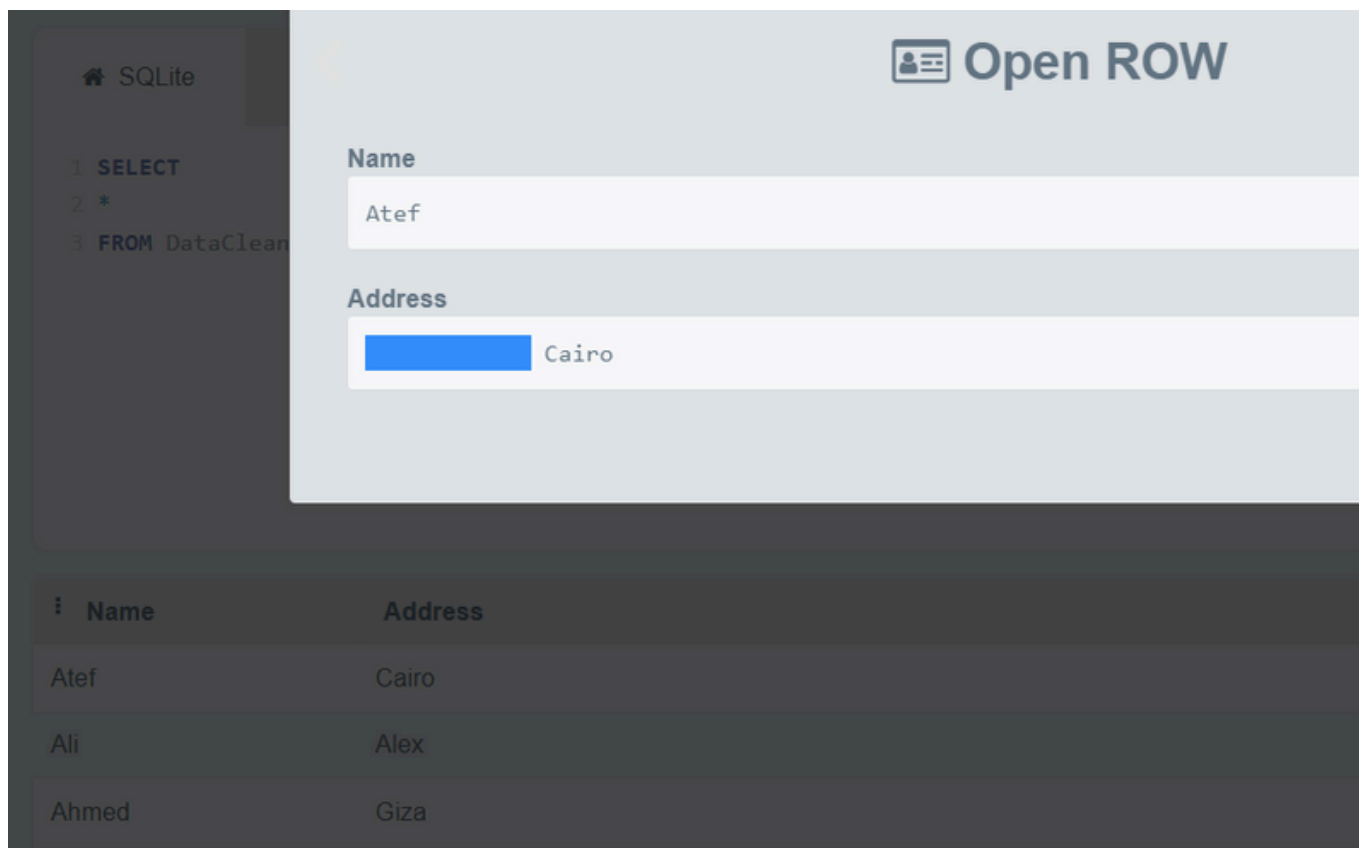
The screenshot shows the SQLite application interface. At the top, there are four tabs, each labeled 'SQLite' with a home icon. The first tab is active. Below the tabs, the SQL query is displayed: `1 SELECT`, `2 name,`, `3 round (test , 1) AS 'Round Test'`, and `4 FROM DataCleaning`. Below the query, the results are shown in a table with two columns: 'Name' and 'Round Test'. The data rows are: Atef (50.1), Ali (100.9), and Ahmed (70.6). A blue arrow points from the first screenshot to this one.

Name	Round Test
Atef	50.1
Ali	100.9
Ahmed	70.6

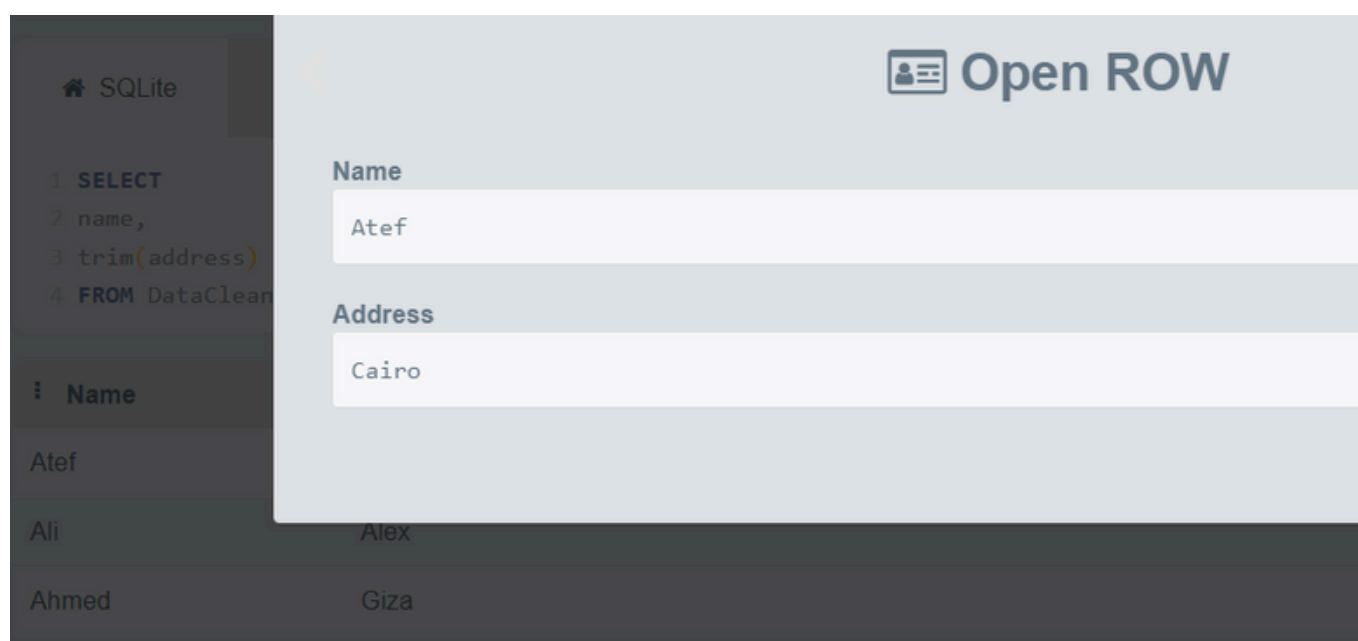


3 ال Trim

وظيفة ال Trim انها تشيل ال Whitespace الزيادة فى الكلمة سواء فى البداية أو النهاية، وعندنا برضه LTRIM & RTRIM عشان تشيل الزيادة من اليمين أو الشمال



ال Query هدفه حذف المساحة البيضاء - ال Whitespace الى موجودة فى أول كلمة Cairo



4 ال Replace

وظيفة ال Replace تبدل الكلمات أو الحروف الى مش عاوزها
بكلمات أو حروف جديدة

```
1 SELECT first_name
2 FROM DataCleaning
```

First_Name

Atef@gmail.comm

Ali@gmail.comm

Ahmed@gmail.comm

ال Query هدفه نشيل كلمة gmail.comm@ عشان مكتوبة غلط بحرف M زيادة ونبدلها
بالكلمة الجديدة الصح

```
4 SELECT
5 first_name,
6 REPLACE (first_name , '@gmail.comm' , '@gmail.com') AS 'New Name'
7
8 FROM DataCleaning
```

First_Name

New Name

Atef@gmail.comm

Atef@gmail.com

Ali@gmail.comm

Ali@gmail.com

Ahmed@gmail.comm

Ahmed@gmail.com



Concat ال 5

وظيفة ال Concat هو دمج كلمتين أو أكثر من كذا عمود

```
1 SELECT *  
2 FROM DataCleaning
```

First_Name	Last_Name
Atef	Mohamed
Ali	Khaled
Ahmed	Reda

ال Query هدفه دمج ال First name مع ال last name مع موجود مساحة ما بينهم

```
1 SELECT  
2 first_name,  
3 last_name ,  
4 concat ( first_name , ' ' ,last_name ) AS 'Full Name'  
5 FROM DataCleaning
```

first_name	last_name	Full Name
Atef	Mohamed	Atef Mohamed
Ali	Khaled	Ali Khaled
Ahmed	Reda	Ahmed Reda



6 ال Cast

وظيفة ال cast هو تحويل نوع الداتا من data type ل data type تانى

```
1 SELECT *  
2 FROM DataCleaning
```

Name	Hiring_Date
Atef	2019-03-14
Ali	2017-08-20
Ahmed	2020-01-05

ال Query هدفه دمج تحويل ال hiring date من date ل datetime

```
1 SELECT  
2  
3 name,  
4 cast ( hiring_date AS datetime ) AS 'Date Time'  
5  
6 FROM DataCleaning
```

name	Date Time
Atef	2019-03-14 00:00:00
Ali	2017-08-20 00:00:00
Ahmed	2020-01-05 00:00:00

كيف تتعامل مع التواريخ في الـ SQL

الـ 5 Functions لتحويل التاريخ:

DATEFORMAT الـ 1-

DATEDIFF الـ 2-

DATEADD الـ 3-

YYY_CAST الـ 4-

EXTRACT الـ 5-



DATEFORMAT JI -1

تحويل التاريخ من سنة، شهر، ويوم لـ يوم ، شهر ، سنة. والشهر ميكونش رقم ويبقى مكتوب بأول 3 حروف من الشهر.

Input:

- Original Date: '2023-11-03'
- Desired Format: '03-Nov-2023'

SQL Query:

sqlCopy code

```
SELECT DATEFORMAT('2023-11-03', 'dd-MMM-yyyy') AS FormattedDate;
```

Output:

- Formatted Date: '03-Nov-2023'

DATEDIFF JI -2

هدفها انك تعرف الفرق ما بين تاريخين، سواء الفرق ده أيام، شهور، أو سنين. وفي المثال اللى معانا الفرق بالأيام.

Input:

- Start Date: '2023-10-15'
- End Date: '2023-11-03'

SQL Query:

sqlCopy code

```
SELECT DATEDIFF(day, '2023-10-15', '2023-11-03') AS DayDifference;
```

Output:

- Day Difference: 19



DATEADD جI -3

هدفها انك تضيف أو تقلل عدد الأيام أو الشهور أو السنين حسب ما تحب، فى المثالى اللى معانا بنضيف 7 أيام.

Input:

- Initial Date: '2023-10-15'
- Days to Add: 7

SQL Query:

sql

Copy code

```
SELECT DATEADD(day, 7, '2023-10-15') AS NewDate;
```

Output:

- New Date: '2023-10-22'

TRY_CAST جI -4

هدفها انها تحول التاريخ اللى مكتوب ولكنه فى الأساس نوع الداتا بتاعته Text ، تحوله لتاريخ.

Input:

- Textual Date: '2023-11-03'

SQL Query:

sql

Copy code

```
SELECT TRY_CAST('2023-11-03' AS DATE) AS ConvertedDate;
```

Output:

- Converted Date: '2023-11-03'



EXTRACT ال-5


هدفها انك تشوف اليوم، الشهر، أو السنة من التاريخ. فى المثال بنشوف الشهر من التاريخ.

Input:

- Date: '2023-11-03'

SQL Query:

sql

 Copy code

```
SELECT EXTRACT(MONTH FROM '2023-11-03') AS Month;
```

Output:

- Month: 11

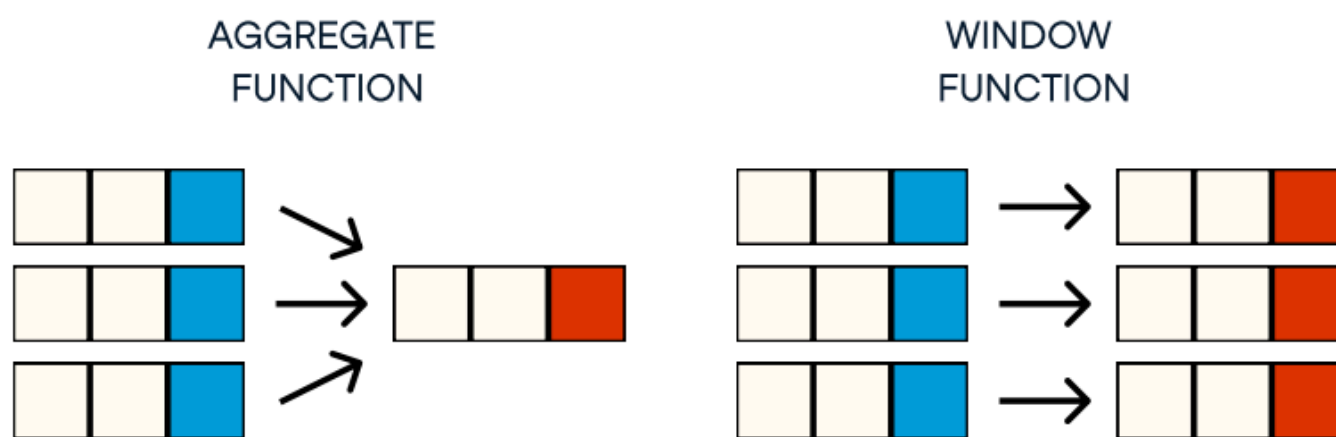


ازای تستخدم ال SQL WINDOW FUNCTIONS كمحلل بيانات؟

ما هي ال Window Functions:

من ال Function اللى مش هيعدي عليك أسبوع فى شغلك غير وأنت بتستخدمها.

هتساعدك انك تعمل عملية حسابية زي ال Sum / AVG أو RANK لاي لعدد من الصفوف ويكون متربط بالصف الحالى.



* ازای تکتب ال Window Function

ال Window function نقدر نقسمها ل 4 أقسام رئيسية:

1- ال Window function نفسها عملية ال SUM / AVG وغيرها أو ال RANK

2- ال OVER

3- ال PARTITION BY

4- ال ORDER BY



```
SELECT
  window_function() OVER(
    PARTITION BY partition_expression
    ORDER BY order_expression
    window_frame_extent
  ) AS window_column_alias
FROM table_name
```



Window Function *

مثال ال Window Function ، هدف ال Query هو إخراج المتوسط AVG للـ List Price بالنسبة لـ Model_year. والـ Table النهائي يكون فيه الـ Model_Price و الـ Product_Name والـ List_Price.

طبعا مكان الـ AVG نقدر نتستخدم أى Aggregation احنا عاوزينه زي الـ SUM / AVG / MAX / MIN



```
SELECT
model_year,
product_name,
list_price,
AVG(list_price) OVER
(PARTITION BY model_year)
avg_price
FROM products
```

model_year	product_name	list_price	avg_price
2018	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	899.99	1658.470441
2017	Electra Amsterdam Fashion 7i Ladies' - 2017	1099.99	1279.931176
2017	Electra Amsterdam Original 3i - 2015/2017	659.99	1279.931176



* الفرق ما بين استخدام الـ Group by و الـ Window Function

GROUP BY الـ

```
SELECT
model_year,
AVG(list_price) avg_price
FROM products
GROUP BY model_year
```

model_year	avg_price
2016	980.29923
2017	1279.931176
2018	1658.470441
2019	2583.323333

PARTITION BY الـ

```
SELECT
model_year,
product_name,
list_price,
AVG(list_price) OVER
(PARTITION BY model_year)
avg_price
FROM products
```

model_year	product_name	list_price	avg_price
2018	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	899.99	1658.470441
2017	Electra Amsterdam Fashion 7i Ladies' - 2017	1099.99	1279.931176
2017	Electra Amsterdam Original 3i - 2015/2017	659.99	1279.931176

المثالين بييجوا AVG الأسعار ولكن في مثال الـ Group by بيجب الـ AVG بالنسبة للـ Model_Year.

ولكن في مثال الـ Partition by بيجب AVG الأسعار بالنسبة للـ Model_Year برضه مع موجود Product Name ، نفس نتيجة الأرقام ولكن انك تحط colume (الـ Product name) زيادة من غير ما يآثر على النتيجة النهائية



ORDER BY *

هدف ال ORDER BY هو ترتيب نتيجة ال PARTITION BY سواء ASC أو DEC

فى المثال بنعمل ترتيب عن طريق استخدام Function اسمها RANK وبنستخدم معاها OVER وبنعمل ORDER BY بال LIST PRICE وبالتالى النتيجة النهائية هو TABLE فيه ال Product Name و ال List Price و ال Rank



```
/* Rank price from LOW->HIGH */  
SELECT  
    product_name,  
    list_price,  
    RANK() OVER  
        (ORDER BY list_price DESC) rank  
FROM products
```

product_name	list_price	rank
Trek Domane SLR 9 Disc - 2018	11999.99	1
Trek Domane SLR 8 Disc - 2018	7499.99	2
Trek Domane SL Frameset - 2018	6499.99	3

```
/* Rank price from HIGH->LOW */  
SELECT  
    product_name,  
    list_price,  
    RANK() OVER  
        (ORDER BY list_price ASC) rank  
FROM products
```

product_name	list_price	rank
Strider Classic 12 Balance Bike - 2018	89.99	1
Sun Bicycles Lil Kitt'n - 2017	109.99	2
Trek Boy's Kickster - 2015/2017	149.99	3



Ranking Window Function *

في المثال اللي فات شوفنا RANK بدل ال Aggregation وهدفها انك تعمل rank لكل صف row بقيمة محددة بناء على ال Order By اللي أنت بتحددها

هنا بنستخدم طرق مختلفة لل Rank زي:

ال ROW_NUMBER: هدفها انك تعمل index لكل row في ال Table النهائي

ال DENSE_RANK: هدفها تعمل rank لكل List Price بيتكرر معنا

ال RANK: هدفها تعمل rank لكل List Price بيتكرر معنا ولو حصل تكرار لل LIST PRICE القيمة اللي بعدها بتعمل skip لل row زي المثال مفهوش rank رقم 4

```
/* Rank all products by price */
SELECT
  product_name,
  list_price,
  ROW_NUMBER() OVER (ORDER BY list_price) AS row_num,
  DENSE_RANK() OVER (ORDER BY list_price) AS dense_rank,
  RANK() OVER (ORDER BY list_price) AS rank,
FROM products
```



product_name	list_price	row_num	dense_rank	rank
Strider Classic 12 Balance Bike - 2018	89.99	1	1	1
Sun Bicycles Lil Kitt'n - 2017	109.99	2	2	2
Trek Boy's Kickster - 2015/2017	149.99	3	3	3
Trek Girl's Kickster - 2017	149.99	4	3	3
Trek Kickster - 2018	159.99	5	4	5
Trek Precaliber 12 Boys - 2017	189.99	6	5	6
Trek Precaliber 12 Girls - 2017	189.99	7	5	6



نصائح لتطوير مستواك ف الـ SQL



5 نصائح عملية لكتابة SQL Query بشكل أفضل

- 1- رجع البيانات الى محتاجها فقط
- 2- اعمل Limit للنتائج
- 3- متعقدش ال Query
- 4- استخدم ال Joins
- 5- استخدم aggregation و group by



1- رجع البيانات الى محتاجها فقط

مع البيانات الكبير، هتلاقى Tables فيها عدد ضخمة من الـ Columns وانت مش محتاج كل دول.

عشان كده ديما رجع فقط الـ Columns اللي محتاجها



SELECT *

FROM [Product_Table]



SELECT Product_name , Price

FROM [Product_Table]



2- اعمل Limit للنتائج

لو مش مفيش Filter معين هتعمله، استخدم Limit
أو Top عشان تظهر جزء بسيط من البيانات تبص عليه



```
SELECT id, name, email  
FROM customers
```



```
SELECT id, name, email  
FROM customers  
LIMIT 5;
```



3- متعقدش ال Query

مش محتاج انك تكتب Query معقد عشان تبان إنك فاهم SQL ، البساطة و الفاعلية لل Query بتاعك هما أهم عاميلين تاخذ بالك منهم وأنت شغال.

عشان لما تراجع ال Query بتاعك تعرف تقرأه ببساطه وزمايلك فى الشغل يفهموه برضه



4- استخدم ال Joins

ركز على فهم واستخدام ال Joins الصح، لانه هياثر بشكل كبير على طريقة كتابتك و أداء ال Query



```
SELECT * FROM customers, orders WHERE customers.customer_id = orders.customer_id;
```



```
SELECT c.customer_id, c.customer_name, o.order_id, o.order_date  
FROM customers c  
INNER JOIN orders o ON c.customer_id = o.customer_id;
```



5- استخدم aggregation و group by

لما تعمل عمليات جمع أو بتعد أو متوسط أى
Aggregation function ديما استخدم Group by
لأنها هتساعدك تلخص البيانات بشكل أفضل.



```
SELECT customer_id, COUNT(order_id) AS order_count, SUM(order_total) AS total_amount  
FROM orders  
GROUP BY customer_id;
```



4 أسباب تخليك تتعلم من الـ SQL Query

من أفضل الطرق عشان تطور مهارتك فى استخدام الـ SQL فمهمك وتحليلك لـ SQL Query بتاعت زمايلك هيساعدك فى فهم المبادئ بشكل أعمق.

1. أنماط مختلفة
2. أساليب جديدة
3. أفضل الممارسات
4. الأخطاء الشائعة



1 أنماط مختلفة

من خلال مراجعة الـ SQL Query لزمائك أو حتى أونلاين
هتتعرف على أنماط وأساليب مختلفة.

هتشوف query الأولوية بتاعته الـ readability واستخدام
الـ aliases بشكل مفيد، وهتشوف query تاني مفهوش
حاجه واضحة.

هتشوف query بيستخدم فيه الـ Subqueries عشان
يعمل عملية حسابية معينة وواحد تاني بيستخدم Window
function وكل أسلوب بيشارك في الـ readability
والـ performance بتاع الـ Query



2 أسلوب جديدة

هتتعرف على أساليب جديدة أول مرة تعرف عنها

وده هيساعد تحل المشاكل اللى هتواجهك بطرق مختلفة،
مجرد معرفتك بالـ CTE (Common Table Expression)
هيساعدك بشكل كبير فى الـ Query انك تجهز البيانات اللى
عاوز تشتغل عليها.

وتجربتك لـ LAG Function هيساعدك تعرف الفرق ما بين
الوقت ما بين rows وده هيساعدك فى تحليل الـ Pattern
والـ Trends.



3 أفضل الممارسات

معرفتک بالـ Best Practices - أفضل الممارسات للـ query هيساعدك تحسن مهارتك في استخدام الـ SQL.

وده هيساهم بشكل كبير في سهولة التعديل و أداء الـ query

هتفهم بشكل أكبر الـ Schema design و طرق الـ optimization. زي استخدام index في الـ tables هياثر بشكل كبير في سرعة أداء الـ query وعلى أداء الـ database كلها.



4 الأخطاء الشائعة

تعلم من الأخطاء الشائعة اللى بتشوفها بتكرر قدامك فى ال SQL Query هيساعدك تبعد عنها.

زي استخدام ال Joins بدون On واللى هيسبب مشاكل فى البيانات اللى هتطلع بسبب عدم تأكيد On (استخدام ال Columes المناسبة من ال 2 Tables أو أكثر بشكل صحيح.



متنساش تعمل شير و تابعنى عشان يوصلك كل جديد عن تحليل البيانات



Mohamed Atef