

I. PREREQUISITE

- First install sqlite3 using the command lines as follows
 1. `sudo apt update`
 2. `sudo apt install sqlite3`
 3. `sudo apt install sqlitebrowser`

II. SQL

Let's create a table 'UserTable' containing two fields Name and Email. This can be done manually in sqlitebrowser or by running a sql command. In the following, commands are printed in with bold letter.

- **CREATE TABLE 'UserTable' ('Name' TEXT,'Email' TEXT)** . Here I say create a table called UserTable that contains two fields "Name" and "Email" and whose values are text.
- Let fill fields with command as **INSERT INTO 'UserTable' ('Name','Email') VALUES ('Kamal', 'kamal@random.com')** . In this command I say fill the 'Name' and 'Email' columns of 'UserTable' with 'Kamal' and 'kamal@random.com'.
- To delete something from the 'UserTable', the command is: **DELETE FROM UserTable WHERE Email='kamal@random.com'**. Here the command says that delete an item from UserTable if Email is 'kamal@random.com'

III. SQL CONTINUE

- To update some information in 'UserTable', The following command is used.
UPDATE 'UserTable' SET Name='Kamal Random' WHERE Email='kamal@random.com' . This means update the Name field of 'UserTable' by a new name 'Kamal Random' whose Email field is 'kamal@random.com'
- IF table needs to be deleted altogether than
DROP TABLE IF EXISTS UserTable . Basically the command says that if there is a table name UserTable then the whole table will be removed or dropped (means drop).

In the following SQL query, I'm using data from five different tables. They are provided in the format of csv files in the Data_Restaurant directory.

IV. FIRST SQL QUESTIONS

Find customers who have never ordered?

The underlying meaning of the question is that find all users who are registered in the platform but have never made any food order yet. This query is related to two tables. The reason is that the customers information are in **user** table and the information related to orders are in order table. We have to query such that it checks all the userID from the user table that are not in the order table.

- **SELECT user_id**
FROM users
WHERE user_id NOT IN (SELECT user_id FROM orders)

This gives user id associated to user but I want user name. For this the following query is used.

- **SELECT name**
FROM users
WHERE user_id NOT IN (SELECT user_id FROM orders)

V. SECOND SQL QUESTION

Average price of all items sold on the platform?

There may be different restaurants on the platform that sell the same product at a different price. The question asks us to find the average price for all items present in different restaurants but sold by the same platform. This means we need to find the prices of the items, sum them up, and divide by the number of restaurants that sell them. The name of foods are in food table, the prices of food are in menu table, and the connection between them is the food id. Just because we want the food name we have to join the two tables together.

- **SELECT f.id, AVG(price) FROM menu**
- **SELECT f.id, AVG(price) FROM menu GROUP BY f.id**

This gives the average price for each item in the menu.

I want the average price with name of food. For this I have to join the food table (for food) and menu table (for price) on the basis of food id. The query looks as the following

- **SELECT f.f_name, ROUND(AVG(price),3)
FROM menu m
JOIN food f ON m.f_id=f.f_id
GROUP BY m.f_id**

The following is the same as above. Above we use aliasing for food and menu as f and m respectively.

- **SELECT food.f_name, ROUND(AVG(price),3)
FROM menu
JOIN food ON menu.f_id=food.f_id
GROUP BY menu.f_id**

VI. THIRD QUESTION

Find top restaurant in terms of number of orders in a given month?

- First extract the month from date column of order table

```
SELECT *, strftime('%m',date) AS 'Month'
FROM orders
```

Here * is used to denote all columns of the table.

- Select a particular month

```
SELECT *, strftime('%m',date) AS 'Month'
FROM orders
WHERE strftime('%m',date) LIKE '07'
```

- Count all restaurant that got order in July

```
SELECT r_id, COUNT(*) AS 'Count'
FROM orders
WHERE strftime('%m',date) LIKE '07'
```

- Count how many orders each restaurant got in July

```
SELECT r_id, COUNT(*) AS 'Count' FROM orders
WHERE strftime('%m',date) LIKE '07'
GROUP BY r_id
ORDER BY COUNT(*) DESC LIMIT 3
```

- Give the name of restaurant who got the most order

```
SELECT restaurant.r_name, COUNT(*) AS 'Count' FROM orders
JOIN restaurant
ON orders.r_id=restaurant.r_id
WHERE strftime('%m',date) LIKE '07'
GROUP BY orders.r_id
ORDER BY COUNT(*) DESC LIMIT 3
```

VII. RESTAURANT WITH MONTHLY SALES IS GREATER THEN X

First let the question be understood. The question is: give the name of the restaurant whose turnover is greater than a certain amount x. The x could be 100 or 500 or 1000 etc.

In tables, the information about the sales for each month is in the "Orders" table as an amount. My strategies are that I first take out the month from date, pick a specific month, total the amount and apply "group by" to the restaurant ID. I want a restaurant name, not a restaurant ID, but the restaurant name is in the restaurant table. So I join the orders table and the restaurant table by restaurant id.

- First select all from orders table as

```
SELECT * FROM orders
```

- Select a specific month, say June, from the date column as

```
SELECT *, strftime('%m',date) AS 'Month'
FROM orders
WHERE strftime('%m',date) LIKE '06' or just
```

```
SELECT *
FROM orders
WHERE strftime('%m',date) LIKE '06'
```

- Total the amount and apply 'Group By' via Restaurant ID as

```
SELECT *, SUM(amount) AS 'TOTAL'
FROM orders
WHERE strftime('%m',date) LIKE '06'
GROUP BY r_id
```

- Keep only the restaurant ID and the total amount

```
SELECT r_id,SUM(amount) AS 'TOTAL'
FROM orders
WHERE strftime('%m',date) LIKE '06'
GROUP BY r_id
```

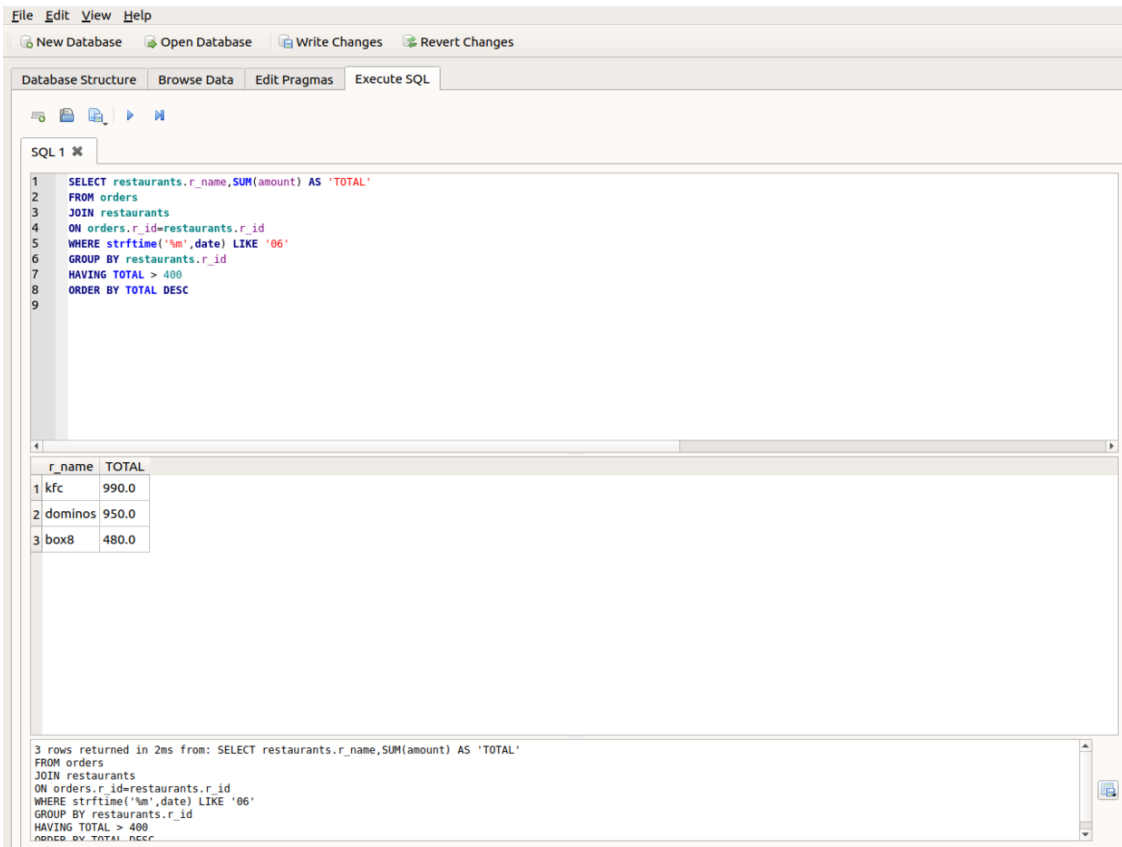
- Instead of the restaurant ID, I'm interested in the name of the restaurant. To do this, I connect the order table to the restaurant table, as indicated below. There will be two r_id and to reduce ambiguity I will specify order.r_id and restaurant.r_id just to say which r_id comes from where. In addition, since I join two table as orders.r_id=restaurant.r_id, the contents of order table will be on the left.

```
SELECT restaurant.r_name,SUM(amount) AS 'Total'
FROM orders
JOIN restaurant
ON orders.r_id=restaurant.r_id
```

```

WHERE strftime('%m',date) LIKE '06'
GROUP BY orders.r_id
HAVING Total > 300
ORDER BY Total DESC

```



VIII. SHOW ALL ORDERS WITH ORDER DETAILS FOR A PARTICULAR CUSTOMER IN A PARTICULAR DATE RANGE

First let the question be understood. It states that find all food items (in this scenario) ordered by a customer in a certain period of time. The necessary information I need is information of the user and what food he/she ordered. All users who have ever ordered food are in the orders table. The details of users are in the users table and details of orders are in the order_details table. And the name of the food is in the food table. This is a nested query and requires joining multiple tables, which I try to explain detail below as much as possible.

I start to work with an orders table that contains the date "OrderID", "UserID", "RID". I do everything that can be done in the orders table for this question such as retrieving date ranges and orders. Then to get a specific customer name I point the orders table user id to the users table user id for details.

- need order id, restaurant name, cuisine, food name and date range

SQL 1 ✖

```
1 SELECT date,users.name,restaurants.cuisine,restaurants.r_name,food.f_name
2 FROM orders
3 JOIN users
4 ON users.user_id=orders.user_id
5
6 JOIN restaurants
7 ON orders.r_id=restaurants.r_id
8
9 JOIN order_details od
10 ON od.order_id=orders.order_id
11
12 JOIN food
13 ON food.f_id=od.f_id
14
15 WHERE date BETWEEN '2022-06-10' AND '2022-07-10'
16 AND orders.user_id like 4
```

	date	name	cuisine	r_name	f_name	
1	2022-06-15	Ankit	South Indian	Dosa Plaza	Schezwan Noodles	
2	2022-06-15	Ankit	South Indian	Dosa Plaza	Veg Manchurian	
3	2022-06-30	Ankit	Chinese	China Town	Schezwan Noodles	
4	2022-06-30	Ankit	Chinese	China Town	Veg Manchurian	

IX. FIND NAME OF STUDENT WHOSE MARK IS GREATER THAN THE AVERAGE AND PRESENT IN THE DESCENDING ORDER.

ID	Name	Mark
1	Rahul	56
2	Jack	35
3	Amar	62
4	Anthony	55
5	Ram	42
6	Shiva	51

To create this table in 'DB Browser for SQLite', following commands are used.

create table studentMark (ID integer, Name TEXT, Mark integer);

insert into studentMark (ID,Name,Mark) values (1,'Rahul',56);

insert into studentMark (ID,Name,Mark) values (2,'Jack',35);

insert into studentMark (ID,Name,Mark) values (3,'Amar',62);

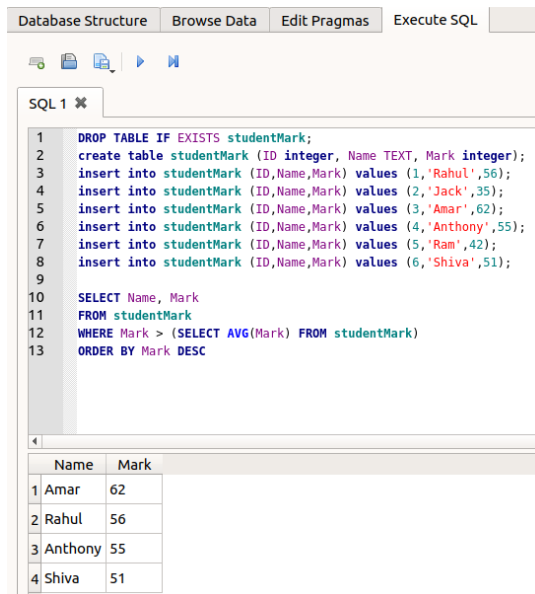
insert into studentMark (ID,Name,Mark) values (4,'Anthony',55);

```
insert into studentMark (ID,Name,Mark) values (5,'Ram',42);
```

```
insert into studentMark (ID,Name,Mark) values (6,'Shiva',51);
```

To complete the asked question both the inner query and the outer query are used. First the inner query is executed to find out the average marks of student. Then query becomes like something WHERE Mark > some_number.

- SELECT Name, Mark
FROM studentMark
WHERE Mark > (SELECT AVG(Mark) FROM studentMark)
ORDER BY Mark DESC



```

1 DROP TABLE IF EXISTS studentMark;
2 create table studentMark (ID integer, Name TEXT, Mark integer);
3 insert into studentMark (ID,Name,Mark) values (1,'Rahul',56);
4 insert into studentMark (ID,Name,Mark) values (2,'Jack',35);
5 insert into studentMark (ID,Name,Mark) values (3,'Amar',62);
6 insert into studentMark (ID,Name,Mark) values (4,'Anthony',55);
7 insert into studentMark (ID,Name,Mark) values (5,'Ram',42);
8 insert into studentMark (ID,Name,Mark) values (6,'Shiva',51);
9
10 SELECT Name, Mark
11 FROM studentMark
12 WHERE Mark > (SELECT AVG(Mark) FROM studentMark)
13 ORDER BY Mark DESC
  
```

	Name	Mark
1	Amar	62
2	Rahul	56
3	Anthony	55
4	Shiva	51

- * Aggregate functions in SQL are COUNT(), AVG(), MIN() and MAX(). Aggregate functions are those function who takes many values and returns one. COUNT(id) scans the data in id column but more data efficient and faster is COUNT(1).
- * It doesn't make sense using GROUP BY method without aggregate function.

X. SQL FROM KAGGLE

This commands are from the introductory course on SQL provided in the Kaggle platform using the BigQuery. All the process there are done using the python scrips which are compiled as below

- Import bigquery module from google.cloud
from google.cloud import bigquery
- Establish the connection
client = bigquery()

- Create the reference to the database

```
database_reference = client.dataset('NameOfDatabase',project='NameOfProjectWhereTheDataBaseIs')  
database_reference = client.dataset('openaq',project='bigquery-public-data')
```

Here it says that the name of database is openaq and it is accessed from 'bigquery-public-data'. Later it will be shown that in database there is a table call global-air-quality. This table can be accessed as *bigquery-public-data.openaq.global-air-quality*

- Get the dataset

```
dataset = client.get_dataset(database_reference)
```

-