Advanced SQL

In SQL, a window function or analytic function is a function which uses values from one or multiple rows to return a value for each row. (This contrasts with an aggregate function, which returns a single value for multiple rows.)

Window functions have an OVER clause; any function without an OVER clause is not a window function, but rather an aggregate or single-row (scalar) function.

Using sqllite database

In [515…
```
# !pip install prettytable==3.0.0
```

In [516…
```
%reload_ext sql
%config SqlMagic.autopandas = True
%config SqlMagic.feedback = False
%config SqlMagic.style = 'default'  # This is the key fix
```

In [518…
```
# # First, close all SQL connections
# %sql --close sqlite:///practice.db
```

In [519…
```
# Delete the database file
!rm -f practice.db
```

In [520…
```
# Connect to an SQLite database (creates if doesn't exist)
%sql sqlite:///practice.db
```

In [521…
```
%%sql
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    department TEXT NOT NULL,
    salary REAL
);
```

```
 * sqlite:///practice.db
```

Out[521]: —

In [522…
```
%%sql
INSERT INTO employees (name, department, salary)
VALUES
    ('John Doe', 'Engineering', 75000),
    ('Jane Smith', 'Marketing', 65000),
    ('Bob Johnson', 'Engineering', 80000);
```

```
 * sqlite:///practice.db
```

Out[522]: —

In [523…
```
# Simple query
%sql SELECT * FROM employees;
```

```
 * sqlite:///practice.db
```

Out[523]:

	id	name	department	salary
0	1	John Doe	Engineering	75000.0
1	2	Jane Smith	Marketing	65000.0
2	3	Bob Johnson	Engineering	80000.0

Queries

In [524…
```
%%sql
CREATE TABLE IF NOT EXISTS Student(
    id INTEGER PRIMARY KEY,
    Marks int NOT NULL,
    Rank int NOT NULL,
    Dense_Rank int NOT NULL,
    Row_Number int NOT NULL
);
```

```
 * sqlite:///practice.db
```

Out[524]: —

In [525…
```
%%sql
INSERT INTO Student (Marks, Rank, Dense_Rank, Row_Number)
VALUES
    (78, 1,1, 1),
    (68, 2,2, 2),
    (68, 2,2, 3),
    (65, 4,3, 4);
```

```
 * sqlite:///practice.db
```

Out[525]: —

In [526…
```
%sql select * from student
```

```
 * sqlite:///practice.db
```

Out[526]:

	id	Marks	Rank	Dense_Rank	Row_Number
0	1	78	1	1	1
1	2	68	2	2	2
2	3	68	2	2	3
3	4	65	4	3	4

In [527…
```
%%sql
select *,rank() over (order by Marks asc) as rnk, dense_rank() over (order by Marks asc) as dense_rnk,
row_number()  over (order by Marks asc) as row_no from Student
```

```
 * sqlite:///practice.db
```

Out[527]:

	id	Marks	Rank	Dense_Rank	Row_Number	rnk	dense_rnk	row_no
0	4	65	4	3	4	1	1	1
1	2	68	2	2	2	2	2	2
2	3	68	2	2	3	2	2	3
3	1	78	1	1	1	4	3	4

In [528…
```
%sql drop table if exists department
```

```
 * sqlite:///practice.db
```

Out[528]: —

In [529…
```
%%sql
create table if not exists Department(
    id INTEGER PRIMARY KEY autoincrement,
    Department varchar(200),
    Salary int
);
```

```
 * sqlite:///practice.db
```

Out[529]: —

In [530…
```
%%sql
insert into Department(Department, Salary) values
('Sales',1000),
('IT',1500),
('Sales',2000),
('Sales',1700),
('IT',1800),
('Accounts',1200),
('Accounts',1100);
```

```
 * sqlite:///practice.db
```

Out[530]: —

In [531…
```
%sql select * from Department
```

```
 * sqlite:///practice.db
```

Out[531]:

	id	Department	Salary
0	1	Sales	1000
1	2	IT	1500
2	3	Sales	2000
3	4	Sales	1700
4	5	IT	1800
5	6	Accounts	1200
6	7	Accounts	1100

In [532…
```
%%sql
select *, rank() over(partition by department order by salary desc) as emp_rnk from department
```

```
 * sqlite:///practice.db
```

Out[532]:

	id	Department	Salary	emp_rnk
0	6	Accounts	1200	1
1	7	Accounts	1100	2
2	5	IT	1800	1
3	2	IT	1500	2
4	3	Sales	2000	1
5	4	Sales	1700	2
6	1	Sales	1000	3

Store the output in a dataframe

In [533…
```
import pandas as pd

# Store query result directly in a DataFrame
df = %sql select *, rank() over(partition by department order by salary desc) as emp_rnk from department

# Display the DataFrame
print(df)
```

```
 * sqlite:///practice.db
   id Department  Salary  emp_rnk
0   6   Accounts    1200        1
1   7   Accounts    1100        2
2   5         IT    1800        1
3   2         IT    1500        2
4   3      Sales    2000        1
5   4      Sales    1700        2
6   1      Sales    1000        3
```

using multiline select statement and store in a dataframe

In [534…
```
%%sql
df << select *, rank() over(partition by department order by salary desc) as emp_rnk
from department
```

```
 * sqlite:///practice.db
Returning data to local variable df
```

In [535…
```
print(df)
```

```
   id Department  Salary  emp_rnk
0   6   Accounts    1200        1
1   7   Accounts    1100        2
2   5         IT    1800        1
3   2         IT    1500        2
4   3      Sales    2000        1
5   4      Sales    1700        2
6   1      Sales    1000        3
```

ROWS BETWEEN Clause

In [536…
```
%sql drop table sales
```

```
 * sqlite:///practice.db
(sqlite3.OperationalError) no such table: sales
[SQL: drop table sales]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

In [537…
```
%%sql
create table if not exists sales(
    ID integer primary key autoincrement,
    Date date,
    Sales int
);
```

```
 * sqlite:///practice.db
```

Out[537]: —

In [538…
```
%%sql
insert into sales(Date,Sales) values
('22-06-2022',603),
('21-06-2022',478),
('20-06-2022',679),
('19-06-2022',443),
('18-06-2022',540),
('17-06-2022',740),
('16-06-2022',850),
('15-06-2022',604),
('14-06-2022',339),
('13-06-2022',905);
```

```
 * sqlite:///practice.db
```

Out[538]: —

In [539…
```
%sql select * from sales
```

```
 * sqlite:///practice.db
```

Out[539]:

	ID	Date	Sales
0	1	22-06-2022	603
1	2	21-06-2022	478
2	3	20-06-2022	679
3	4	19-06-2022	443
4	5	18-06-2022	540
5	6	17-06-2022	740
6	7	16-06-2022	850
7	8	15-06-2022	604
8	9	14-06-2022	339
9	10	13-06-2022	905

Data we want: current sales + prev. day sales+ next day sales for each row except first and last row

So, in general, we want current row value+ m rows values preceding + n rows values following.

In [540…
```
%%sql
select *,sum(Sales) over (order by Date desc rows between 1 preceding and 1 following) as New_Sale
from sales
```

```
 * sqlite:///practice.db
```

Out[540]:

	ID	Date	Sales	New_Sale
0	1	22-06-2022	603	1081
1	2	21-06-2022	478	1760
2	3	20-06-2022	679	1600
3	4	19-06-2022	443	1662
4	5	18-06-2022	540	1723
5	6	17-06-2022	740	2130
6	7	16-06-2022	850	2194
7	8	15-06-2022	604	1793
8	9	14-06-2022	339	1848
9	10	13-06-2022	905	1244

SQL Query for sum of all rows before a particular row and the all rows after a particular row:

In [541…
```
%%sql
select *,sum(Sales) over (order by Date desc rows between unbounded preceding and unbounded following) as New_S
from sales
```

```
 * sqlite:///practice.db
```

Out[541]:

	ID	Date	Sales	New_Sale
0	1	22-06-2022	603	6181
1	2	21-06-2022	478	6181
2	3	20-06-2022	679	6181
3	4	19-06-2022	443	6181
4	5	18-06-2022	540	6181
5	6	17-06-2022	740	6181
6	7	16-06-2022	850	6181
7	8	15-06-2022	604	6181
8	9	14-06-2022	339	6181
9	10	13-06-2022	905	6181

Cummulative Sum (Running Sum) in SQL

In [542…
```
%%sql
select *,sum(Sales) over (order by Date desc rows between unbounded preceding and current row) as New_Sale
```

```
from sales
```

```
 * sqlite:///practice.db
```

Out[542]:

	ID	Date	Sales	New_Sale
0	1	22-06-2022	603	603
1	2	21-06-2022	478	1081
2	3	20-06-2022	679	1760
3	4	19-06-2022	443	2203
4	5	18-06-2022	540	2743
5	6	17-06-2022	740	3483
6	7	16-06-2022	850	4333
7	8	15-06-2022	604	4937
8	9	14-06-2022	339	5276
9	10	13-06-2022	905	6181

In [543…

```
%%sql
alter table sales
add State varchar(500)
```

```
 * sqlite:///practice.db
```

Out[543]: —

In [544…

```
%sql select * from sales
```

```
 * sqlite:///practice.db
```

Out[544]:

	ID	Date	Sales	State
0	1	22-06-2022	603	None
1	2	21-06-2022	478	None
2	3	20-06-2022	679	None
3	4	19-06-2022	443	None
4	5	18-06-2022	540	None
5	6	17-06-2022	740	None
6	7	16-06-2022	850	None
7	8	15-06-2022	604	None
8	9	14-06-2022	339	None
9	10	13-06-2022	905	None

In [545…

```
%%sql
UPDATE sales SET State =
    CASE
        WHEN Sales in (603,478,679) THEN 'Jharkhand'
        WHEN Sales in (443,540,740) THEN 'Bihar'
        WHEN Sales in (850,604,339) THEN 'Uttar Pradesh'
        ELSE 'Maharastra'
    END;
```

```
 * sqlite:///practice.db
```

Out[545]: —

In [546…

```
%sql select * from sales
```

```
 * sqlite:///practice.db
```

Out[546]:

	ID	Date	Sales	State
0	1	22-06-2022	603	Jharkhand
1	2	21-06-2022	478	Jharkhand
2	3	20-06-2022	679	Jharkhand
3	4	19-06-2022	443	Bihar
4	5	18-06-2022	540	Bihar
5	6	17-06-2022	740	Bihar
6	7	16-06-2022	850	Uttar Pradesh
7	8	15-06-2022	604	Uttar Pradesh
8	9	14-06-2022	339	Uttar Pradesh
9	10	13-06-2022	905	Maharastra

In [547…
```
%%sql
select *, sum(sales) over (partition by State order by Date rows between unbounded preceding and current row)
running_total from sales
```

```
 * sqlite:///practice.db
```

Out[547]:

	ID	Date	Sales	State	running_total
0	6	17-06-2022	740	Bihar	740
1	5	18-06-2022	540	Bihar	1280
2	4	19-06-2022	443	Bihar	1723
3	3	20-06-2022	679	Jharkhand	679
4	2	21-06-2022	478	Jharkhand	1157
5	1	22-06-2022	603	Jharkhand	1760
6	10	13-06-2022	905	Maharastra	905
7	9	14-06-2022	339	Uttar Pradesh	339
8	8	15-06-2022	604	Uttar Pradesh	943
9	7	16-06-2022	850	Uttar Pradesh	1793

First Value, Last Value and Nth Value in SQL:

In [548…
```
%%sql
select *, first_value(Sales) over (partition by State order by Date) as first_day_sales,
last_value(Sales) over (partition by State order by Date rows between unbounded preceding and unbounded followi
as last_day_sales from sales
```

```
 * sqlite:///practice.db
```

Out[548]:

	ID	Date	Sales	State	first_day_sales	last_day_sales
0	6	17-06-2022	740	Bihar	740	443
1	5	18-06-2022	540	Bihar	740	443
2	4	19-06-2022	443	Bihar	740	443
3	3	20-06-2022	679	Jharkhand	679	603
4	2	21-06-2022	478	Jharkhand	679	603
5	1	22-06-2022	603	Jharkhand	679	603
6	10	13-06-2022	905	Maharastra	905	905
7	9	14-06-2022	339	Uttar Pradesh	339	850
8	8	15-06-2022	604	Uttar Pradesh	339	850
9	7	16-06-2022	850	Uttar Pradesh	339	850

In [549…
```
%%sql
select *,
nth_value(Sales, 1) over (partition by State order by Date) as second_day_sales
from sales
```

```
 * sqlite:///practice.db
```

Out[549]:

	ID	Date	Sales	State	second_day_sales
0	6	17-06-2022	740	Bihar	740
1	5	18-06-2022	540	Bihar	740
2	4	19-06-2022	443	Bihar	740
3	3	20-06-2022	679	Jharkhand	679
4	2	21-06-2022	478	Jharkhand	679
5	1	22-06-2022	603	Jharkhand	679
6	10	13-06-2022	905	Maharastra	905
7	9	14-06-2022	339	Uttar Pradesh	339
8	8	15-06-2022	604	Uttar Pradesh	339
9	7	16-06-2022	850	Uttar Pradesh	339

In [550…
```
%%sql
select *,
nth_value(Sales, 2) over (partition by State order by Date) as second_day_sales
from sales
```

```
 * sqlite:///practice.db
```

Out[550]:

	ID	Date	Sales	State	second_day_sales
0	6	17-06-2022	740	Bihar	NaN
1	5	18-06-2022	540	Bihar	540.0
2	4	19-06-2022	443	Bihar	540.0
3	3	20-06-2022	679	Jharkhand	NaN
4	2	21-06-2022	478	Jharkhand	478.0
5	1	22-06-2022	603	Jharkhand	478.0
6	10	13-06-2022	905	Maharastra	NaN
7	9	14-06-2022	339	Uttar Pradesh	NaN
8	8	15-06-2022	604	Uttar Pradesh	604.0
9	7	16-06-2022	850	Uttar Pradesh	604.0

In [551…
```
%%sql
select *,
nth_value(Sales, 3) over (partition by State order by Date) as second_day_sales
from sales
```

```
 * sqlite:///practice.db
```

Out[551]:

	ID	Date	Sales	State	second_day_sales
0	6	17-06-2022	740	Bihar	NaN
1	5	18-06-2022	540	Bihar	NaN
2	4	19-06-2022	443	Bihar	443.0
3	3	20-06-2022	679	Jharkhand	NaN
4	2	21-06-2022	478	Jharkhand	NaN
5	1	22-06-2022	603	Jharkhand	603.0
6	10	13-06-2022	905	Maharastra	NaN
7	9	14-06-2022	339	Uttar Pradesh	NaN
8	8	15-06-2022	604	Uttar Pradesh	NaN
9	7	16-06-2022	850	Uttar Pradesh	850.0

Partition By in SQL:

Similar to Group By.

In [552…
```
%%sql
create table if not exists cricket(
    PlayerName varchar(200) primary key,
    StadiumName varchar(200),
    Year year,
    Runs integer,
    Country varchar(200)
);
```

```
 * sqlite:///practice.db
```

Out[552]: —

In [553…
```
%%sql
insert into cricket values ('P1','Eden Garden', '2018',421, 'India'),('P2','Wankhede', '2018',450, 'England');
```

```
 * sqlite:///practice.db
```

Out[553]: —

In [554…
```
%sql select * from cricket
```

```
 * sqlite:///practice.db
```

Out[554]:

	PlayerName	StadiumName	Year	Runs	Country
0	P1	Eden Garden	2018	421	India
1	P2	Wankhede	2018	450	England

A. Filter Indian Players:

In [555…
```
%sql select * from cricket where country='India'
```

```
 * sqlite:///practice.db
```

Out[555]:

	PlayerName	StadiumName	Year	Runs	Country
0	P1	Eden Garden	2018	421	India

apply rank partition by StadiumName,Year:

In [556…
```
%sql select *, rank() over (partition by stadiumname order by runs desc) rnk from cricket
```

```
 * sqlite:///practice.db
```

Out[556]:

	PlayerName	StadiumName	Year	Runs	Country	rnk
0	P1	Eden Garden	2018	421	India	1
1	P2	Wankhede	2018	450	England	1

In [557…
```
%%sql
select * from (select *, rank() over (partition by stadiumname order by runs desc) rnk from cricket) a
where a.rnk=1
```

```
 * sqlite:///practice.db
```

Out[557]:

	PlayerName	StadiumName	Year	Runs	Country	rnk
0	P1	Eden Garden	2018	421	India	1
1	P2	Wankhede	2018	450	England	1

In [558…
```
%sql  select *, rank() over (partition by stadiumname, year order by runs desc) rnk from cricket
```

```
 * sqlite:///practice.db
```

Out[558]:

	PlayerName	StadiumName	Year	Runs	Country	rnk
0	P1	Eden Garden	2018	421	India	1
1	P2	Wankhede	2018	450	England	1

In [559…
```
%%sql
select * from (select *, rank() over (partition by stadiumname, year order by runs desc) rnk from cricket) a
where a.rnk=1
```

```
 * sqlite:///practice.db
```

Out[559]:

	PlayerName	StadiumName	Year	Runs	Country	rnk
0	P1	Eden Garden	2018	421	India	1
1	P2	Wankhede	2018	450	England	1

using first value top run scorer in ground/year

In [560…
```
%%sql
select *, first_value(playername) over(partition by stadiumname order by runs desc) highest_run_scored
from cricket
```

```
 * sqlite:///practice.db
```

Out[560]:

	PlayerName	StadiumName	Year	Runs	Country	highest_run_scored
0	P1	Eden Garden	2018	421	India	P1
1	P2	Wankhede	2018	450	England	P2

In [561…
```
%%sql
select *, last_value(playername) over(partition by stadiumname order by runs desc rows between
unbounded preceding and unbounded following) as highest_run_scored from cricket
```

```
 * sqlite:///practice.db
```

Out[561]:

	PlayerName	StadiumName	Year	Runs	Country	highest_run_scored
0	P1	Eden Garden	2018	421	India	P1
1	P2	Wankhede	2018	450	England	P2

Get difference between top scorer and player

In [562…
```
%%sql
select *, first_value(runs) over(partition by stadiumname order by runs desc) - runs diff_runs  from cricket
```

```
 * sqlite:///practice.db
```

Out[562]:

	PlayerName	StadiumName	Year	Runs	Country	diff_runs
0	P1	Eden Garden	2018	421	India	0
1	P2	Wankhede	2018	450	England	0

In [563…
```
%%sql
```

In [563…
```
%%sql
select *, first_value(runs) over(partition by stadiumname,year order by runs desc) - runs diff_runs
from cricket
```

```
 * sqlite:///practice.db
```

Out[563]:

	PlayerName	StadiumName	Year	Runs	Country	diff_runs
0	P1	Eden Garden	2018	421	India	0
1	P2	Wankhede	2018	450	England	0

Moving Average/Rolling Average/Rolling Mean in SQL:

it is a smoothing technique for time series data. it removed the spikes/noise

In [564…
```
%%sql
create table moving_avg(
    Date date,
    Close integer
);
```

```
 * sqlite:///practice.db
```

Out[564]: —

In [565…
```
%%sql
insert into moving_avg values
('28-5-2021',1103.5),
('31-5-2021',1125.65),
('01-6-2021',1100.9),
('02-6-2021',1124.05),
('03-6-2021',1120.7),
('04-6-2021',1128.7),
('07-6-2021',1111.1),
('08-6-2021',1114.45),
('09-6-2021',1158.35);
```

```
 * sqlite:///practice.db
```

Out[565]: —

In [566…
```
%sql select * from moving_avg
```

```
 * sqlite:///practice.db
```

Out[566]:

	Date	Close
0	28-5-2021	1103.50
1	31-5-2021	1125.65
2	01-6-2021	1100.90
3	02-6-2021	1124.05
4	03-6-2021	1120.70
5	04-6-2021	1128.70
6	07-6-2021	1111.10
7	08-6-2021	1114.45
8	09-6-2021	1158.35

In [567…
```
%%sql
select *, avg(Close) over (order by Date asc rows between 2 preceding and current row) as
three_days_moving_avg from moving_avg
```

```
 * sqlite:///practice.db
```

Out[567]:

	Date	Close	three_days_moving_avg
0	01-6-2021	1100.90	1100.900000
1	02-6-2021	1124.05	1112.475000
2	03-6-2021	1120.70	1115.216667
3	04-6-2021	1128.70	1124.483333
4	07-6-2021	1111.10	1120.166667
5	08-6-2021	1114.45	1118.083333
6	09-6-2021	1158.35	1127.966667
7	28-5-2021	1103.50	1125.433333
8	31-5-2021	1125.65	1129.166667

Lead and Lag in SQL:

In [568…
```
%%sql
```

```
create table journey(
    Train_Number integer primary key,
    Station varchar(200),
    Time time
);
```

```
 * sqlite:///practice.db
```

Out[568]: —

In [569…
```
%%sql
insert into journey values
(22863,'Howrah','10:50:00'),
(22864,'Kharagpur','12:30:00'),
(22865,'Balasore','13:52:00'),
(22866,'Cuttack','15:47:00'),
(22867,'Bhubaneswar','16:25:00'),
(12262,'Howrah','05:45:00'),
(12263,'Tatanagar','09:00:00'),
(12264,'Bilaspur','15:05:00'),
(12265,'Raipur','16:37:00'),
(12266,'Nagpur','20:55:00');
```

```
 * sqlite:///practice.db
```

Out[569]: —

In [570…
```
%sql select * from journey
```

```
 * sqlite:///practice.db
```

Out[570]:

	Train_Number	Station	Time
0	12262	Howrah	05:45:00
1	12263	Tatanagar	09:00:00
2	12264	Bilaspur	15:05:00
3	12265	Raipur	16:37:00
4	12266	Nagpur	20:55:00
5	22863	Howrah	10:50:00
6	22864	Kharagpur	12:30:00
7	22865	Balasore	13:52:00
8	22866	Cuttack	15:47:00
9	22867	Bhubaneswar	16:25:00

In [571…
```
%%sql
select *, lead(Time) over(order by time) as temp  from journey
```

```
 * sqlite:///practice.db
```

Out[571]:

	Train_Number	Station	Time	temp
0	12262	Howrah	05:45:00	09:00:00
1	12263	Tatanagar	09:00:00	10:50:00
2	22863	Howrah	10:50:00	12:30:00
3	22864	Kharagpur	12:30:00	13:52:00
4	22865	Balasore	13:52:00	15:05:00
5	12264	Bilaspur	15:05:00	15:47:00
6	22866	Cuttack	15:47:00	16:25:00
7	22867	Bhubaneswar	16:25:00	16:37:00
8	12265	Raipur	16:37:00	20:55:00
9	12266	Nagpur	20:55:00	None

In [572…
```
%%sql
select *, lead(Time) over(order by time)-time time_to_next_station  from journey
```

```
 * sqlite:///practice.db
```

Out[572]:

	Train_Number	Station	Time	time_to_next_station
0	12262	Howrah	05:45:00	4.0
1	12263	Tatanagar	09:00:00	1.0
2	22863	Howrah	10:50:00	2.0
3	22864	Kharagpur	12:30:00	1.0
4	22865	Balasore	13:52:00	2.0
5	12264	Bilaspur	15:05:00	0.0
6	22866	Cuttack	15:47:00	1.0
7	22867	Bhubaneswar	16:25:00	0.0
8	12265	Raipur	16:37:00	4.0
9	12266	Nagpur	20:55:00	NaN

In [573…
```
%%sql
create table players(
    Player varchar(200),
    Year year,
    Runs integer
);
```

```
 * sqlite:///practice.db
```

Out[573]: —

In [574…
```
%%sql
insert into players values
('Virat','2008',159),
('Virat','2009',325),
('Rohit','2010',225);
```

```
 * sqlite:///practice.db
```

Out[574]: —

In [575…
```
%sql select * from players
```

```
 * sqlite:///practice.db
```

Out[575]:

	Player	Year	Runs
0	Virat	2008	159
1	Virat	2009	325
2	Rohit	2010	225

1. total runs scored by virat and rohit:

In [576…
```
%%sql
select Player,sum(Runs) as total_runs from players group by Player
```

```
 * sqlite:///practice.db
```

Out[576]:

	Player	total_runs
0	Rohit	225
1	Virat	484

1. Which year scored what percent of runs:

In [577…
```
%%sql
select player,year,runs,(runs*1.0/sum(runs) over (partition by player order by year rows between
unbounded preceding and unbounded following))*100 total_runs_percentage
from players
```

```
 * sqlite:///practice.db
```

Out[577]:

	Player	Year	Runs	total_runs_percentage
0	Rohit	2010	225	100.00000
1	Virat	2008	159	32.85124
2	Virat	2009	325	67.14876

or

In [578…
```
%%sql
select player,year,runs,(cast(runs as float)/sum(runs) over (partition by player order by year rows
between unbounded preceding and unbounded following))*100 total_runs_percentage
```

```
from players
```

```
 * sqlite:///practice.db
```

Out[578]:

	Player	Year	Runs	total_runs_percentage
0	Rohit	2010	225	100.00000
1	Virat	2008	159	32.85124
2	Virat	2009	325	67.14876

1. In how many years they scored runs less than previous year

In [579…

```
%%sql
select player, year, runs,lag(runs) over (partition by player order by year) as lag_value from players
```

```
 * sqlite:///practice.db
```

Out[579]:

	Player	Year	Runs	lag_value
0	Rohit	2010	225	NaN
1	Virat	2008	159	NaN
2	Virat	2009	325	159.0

In [580…

```
%%sql
select player, year, runs,lag(runs) over (partition by player order by year) - runs prev_year_runs from players
```

```
 * sqlite:///practice.db
```

Out[580]:

	Player	Year	Runs	prev_year_runs
0	Rohit	2010	225	NaN
1	Virat	2008	159	NaN
2	Virat	2009	325	-166.0

In [581…

```
%%sql
select *, case when prev_year_runs> 0 then 1 else 0 end as more_runs_less_runs from (select player, year, runs,
lag(runs) over (partition by player order by year) - runs prev_year_runs from players) a
```

```
 * sqlite:///practice.db
```

Out[581]:

	player	year	runs	prev_year_runs	more_runs_less_runs
0	Rohit	2010	225	NaN	0
1	Virat	2008	159	NaN	0
2	Virat	2009	325	-166.0	0

In [582…

```
%%sql
select player, sum(more_runs_less_runs) from
(select *, case when prev_year_runs> 0 then 1 else 0 end as more_runs_less_runs from (select player, year, runs
lag(runs) over (partition by player order by year) - runs prev_year_runs from players) a) b group by player
```

```
 * sqlite:///practice.db
```

Out[582]:

	player	sum(more_runs_less_runs)
0	Rohit	0
1	Virat	0

1. Count number of years in which rohit has scored more than virat

In [583…

```
%%sql
select player,year,runs,lead(runs) over(partition by year order by player) runs_scored_by_virat from players
```

```
 * sqlite:///practice.db
```

Out[583]:

	Player	Year	Runs	runs_scored_by_virat
0	Virat	2008	159	None
1	Virat	2009	325	None
2	Rohit	2010	225	None

In [584…

```
%%sql
select player, sum(diff_runs) from (select *, case when diff<0 then 1 else 0 end as diff_runs
from (select *,runs_scored_by_virat - runs diff from (select player,year,runs,lead(runs)
over(partition by year order by player) runs_scored_by_virat from players) a
where runs_scored_by_virat is not null) b) c group by player
```

```
 * sqlite:///practice.db
```

Out[584]: —

1. layers scored runs in prev year and next year, count number of times in which score is increasing for continuously 3 years.

In [585…
```
%%sql
select player,sum(incr_runs) from
(select *, case when prev_year_runs < runs and runs< next_year_runs then 1 else 0 end as incr_runs
from (select player, year,runs, lag(runs) over (partition by player order by year) prev_year_runs,
lead(runs) over(partition by player order by year) next_year_runs
from players) a) b group by player
```

```
 * sqlite:///practice.db
```

Out[585]:

	player	sum(incr_runs)
0	Rohit	0
1	Virat	0

Nth value and ntile in sql:

In [586…
```
%sql select * from sales
```

```
 * sqlite:///practice.db
```

Out[586]:

	ID	Date	Sales	State
0	1	22-06-2022	603	Jharkhand
1	2	21-06-2022	478	Jharkhand
2	3	20-06-2022	679	Jharkhand
3	4	19-06-2022	443	Bihar
4	5	18-06-2022	540	Bihar
5	6	17-06-2022	740	Bihar
6	7	16-06-2022	850	Uttar Pradesh
7	8	15-06-2022	604	Uttar Pradesh
8	9	14-06-2022	339	Uttar Pradesh
9	10	13-06-2022	905	Maharastra

In [587…
```
%%sql
select *,nth_value(sales,2) over
(partition by state order by Date rows between unbounded preceding and unbounded following) nth_value from sale
```

```
 * sqlite:///practice.db
```

Out[587]:

	ID	Date	Sales	State	nth_value
0	6	17-06-2022	740	Bihar	540.0
1	5	18-06-2022	540	Bihar	540.0
2	4	19-06-2022	443	Bihar	540.0
3	3	20-06-2022	679	Jharkhand	478.0
4	2	21-06-2022	478	Jharkhand	478.0
5	1	22-06-2022	603	Jharkhand	478.0
6	10	13-06-2022	905	Maharastra	NaN
7	9	14-06-2022	339	Uttar Pradesh	604.0
8	8	15-06-2022	604	Uttar Pradesh	604.0
9	7	16-06-2022	850	Uttar Pradesh	604.0

ntile is used for grouping based on number of rows in a particular group.

for example, if we have 12 rows in a dataset and we want a group of 3 rows, so we will have 4 groups.

-- example: 3 groups:

In [588…
```
%%sql
select id,date,state,sales, case when a.n=1 then 'High Sales' when a.n=2 then 'Medium Sales' else 'Low Sales'
end as Sales_Value from (select *, ntile(3) over(partition by State order by Sales desc) n from sales) a
```

```
 * sqlite:///practice.db
```

Out[588]:

	ID	Date	State	Sales	Sales_Value
0	6	17-06-2022	Bihar	740	High Sales
1	5	18-06-2022	Bihar	540	Medium Sales
2	4	19-06-2022	Bihar	443	Low Sales
3	3	20-06-2022	Jharkhand	679	High Sales
4	1	22-06-2022	Jharkhand	603	Medium Sales
5	2	21-06-2022	Jharkhand	478	Low Sales
6	10	13-06-2022	Maharastra	905	High Sales
7	7	16-06-2022	Uttar Pradesh	850	High Sales
8	8	15-06-2022	Uttar Pradesh	604	Medium Sales
9	9	14-06-2022	Uttar Pradesh	339	Low Sales

order of execution: where --> group by --> having --> order by --> limit

cross join works even there is no common column by using cartestian product.

`select a.*,b.* from table1 a, table2 b`

Non-Equi Join is used when we have to apply join condition other than "=" like <,>,!=

ISNULL and NULLIF in SQL:

In [589…
```
%%sql
create table subjects(
    id integer primary key autoincrement,
    maths integer,
    english integer,
    physics integer,
    chemistry integer,
    computer_science integer
);
```

```
 * sqlite:///practice.db
```

Out[589]: —

In [590…
```
%%sql
insert into subjects(maths,english,physics,chemistry,computer_science) values
(34,31,NULL,12,36),
(NULL,NULL,45,NULL,35);
```

```
 * sqlite:///practice.db
```

Out[590]: —

In [591…
```
%sql select * from subjects
```

```
 * sqlite:///practice.db
```

Out[591]:

	id	maths	english	physics	chemistry	computer_science
0	1	34.0	31.0	NaN	12.0	36
1	2	NaN	NaN	45.0	NaN	35

we need to find sum of all subjects for each id.

replace null with zero.

SQLite doesn't have an `ISNULL()` function. Instead, you should use the `IFNULL()` function or the more standard COALESCE() function.

In [592…
```
%%sql
select *, ifnull(maths,0)+ifnull(english,0)+ifnull(physics,0)+ifnull(chemistry,0)+ifnull(computer_science,0)
as total_marks from subjects
```

```
 * sqlite:///practice.db
```

Out[592]:

	id	maths	english	physics	chemistry	computer_science	total_marks
0	1	34.0	31.0	NaN	12.0	36	113
1	2	NaN	NaN	45.0	NaN	35	80

In [593…
```
%%sql
select *, coalesce(maths,0)+coalesce(english,0)+coalesce(physics,0)+coalesce(chemistry,0)+
coalesce(computer_science,0) as total_marks
from subjects
```

```
 * sqlite:///practice.db
```

Out[593]:

	id	maths	english	physics	chemistry	computer_science	total_marks
0	1	34.0	31.0	NaN	12.0	36	113
1	2	NaN	NaN	45.0	NaN	35	80

opposite of `isnull()` is `nullif()` --> replace non null to null

Joins

In [594…

```
# Inner Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4
# 3

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2

# (total 6 rows)

# Left Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4
# 3

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# 4    NULL
# (total 7 rows)

# Right Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4
# 3

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# (total 6 rows) -- all values from right table even if no matching value does not found in left table

# Outer Join:

# id             id

# 1              1
```

```
# 1             2
# 2             3
# 3             2
# 4             6
# 3

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# 4    NULL
# NULL 6

# (total 8 rows)
```

Full Outer Join = Union of right and left join

In [595…

```
# Inner Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4              NULL
# 3
# NULL
# NULL

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2

# (total 6 rows)

# Left Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4              NULL
# 3
# NULL
# NULL

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# 4    NULL
# NULL NULL
# NULL NULL

# (total 9 rows)

# Right Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4              NULL
# 3
```

```
# NULL
# NULL

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# NULL NULL

# (total 7 rows)

# Outer Join:

# id             id

# 1              1
# 1              2
# 2              3
# 3              2
# 4              NULL
# 3
# NULL
# NULL

# Result:

# id   id

# 1    1
# 1    1
# 2    2
# 3    3
# 3    3
# 2    2
# 4    NULL
# NULL NULL
# NULL NULL
# NULL NULL

# (total 10 rows)
```

Difference between 2 timestamps in SQL:

In [596…
```
# Timing1   |   Timing2    |  diff
# 10:50:00  |   12:30:00   |  1 hour 40 min 0 sec
# 12:30:00  |   13:52:00   |  1 hour 22 min 0 sec
# 05:45:00  |   09:00:00   |  3 hours 15 min 0 sec
```

In [597…
```
%%sql
create table time_table(
    Timing1 time,
    Timing2 time
);
```

```
 * sqlite:///practice.db
```

Out[597]: —

In [598…
```
%%sql
insert into time_table values
('10:50:00','12:30:00'),
('12:30:00','13:52:00'),
('05:45:00','09:00:00');
```

```
 * sqlite:///practice.db
```

Out[598]: —

In [599…
```
%sql select * from time_table;
```

```
 * sqlite:///practice.db
```

Out[599]:

	Timing1	Timing2
0	10:50:00	12:30:00
1	12:30:00	13:52:00
2	05:45:00	09:00:00

-- difference in terms of seconds

SQLite doesn't have a DATEDIFF() function. Instead, you need to use SQLite's date/time functions.

In [600…
```
%%sql
select *, (strftime('%s', timing2) - strftime('%s', timing1)) as time_diff_sec
from time_table
```

```
 * sqlite:///practice.db
```

Out[600]:

	Timing1	Timing2	time_diff_sec
0	10:50:00	12:30:00	6000
1	12:30:00	13:52:00	4920
2	05:45:00	09:00:00	11700

difference in terms of hours part

In [601…
```
%%sql
select *, (strftime('%s', timing2) - strftime('%s', timing1))/3600.0 as time_diff_sec
from time_table
```

```
 * sqlite:///practice.db
```

Out[601]:

	Timing1	Timing2	time_diff_sec
0	10:50:00	12:30:00	1.666667
1	12:30:00	13:52:00	1.366667
2	05:45:00	09:00:00	3.250000

List Aggregate and String Aggregate in SQL:

In [602…
```
# custid   |  orderid    |  item      |  quantity

# c1       |  1          | 'mouse'    |  2
# c1       |  1          | 'keyboard' |  3
# c1       |  1          | 'headphone'|  5
# c1       |  1          | 'laptop'   |  1
# c1       |  1          | 'pendrive' |  3
```

In [603…
```
%%sql
create table items(
    custid varchar(20),
    orderid integer,
    item varchar(200),
    quantity integer
);
```

```
 * sqlite:///practice.db
```

Out[603]: —

In [604…
```
%%sql
insert into items values
('c1',1,'mouse',2),
('c1',1,'keyboard',3),
('c1',1,'headphone',5),
('c1',1,'laptop',1),
('c1',1,'pendrive',3)
```

```
 * sqlite:///practice.db
```

Out[604]: —

In [605…
```
%sql select * from items
```

```
 * sqlite:///practice.db
```

Out[605]:

	custid	orderid	item	quantity
0	c1	1	mouse	2
1	c1	1	keyboard	3
2	c1	1	headphone	5
3	c1	1	laptop	1
4	c1	1	pendrive	3

In [606…
```
# Result we want:

# custid   |  summary

# c1       |  mouse-2,keyboard-3,headphones-5,laptop-1,pendrive-6
```

In [607…
```
%sql select string_agg(item,',') summary from items
```

```
 * sqlite:///practice.db
```

Out[607]:

	summary
0	mouse,keyboard,headphone,laptop,pendrive

In [608…
```
%%sql
select string_agg(detail,',') summary from (select concat(item,'-',quantity) detail from items) a
```

```
 * sqlite:///practice.db
```

Out[608]:

	summary
0	mouse-2,keyboard-3,headphone-5,laptop-1,pendri...

In [609…
```
%%sql
select a.custid, string_agg(detail,',') as summary from (select custid, concat(item,'-',quantity) detail
from items) a group by a.custid
```

```
 * sqlite:///practice.db
```

Out[609]:

	custid	summary
0	c1	mouse-2,keyboard-3,headphone-5,laptop-1,pendri...

or

In [610…
```
%%sql
select a.custid, group_concat(detail, ',') as summary
from (
    select custid, item || '-' || quantity as detail
    from items
) a
group by a.custid
```

```
 * sqlite:///practice.db
```

Out[610]:

	custid	summary
0	c1	mouse-2,keyboard-3,headphone-5,laptop-1,pendri...

SQLite doesn't support the STRING_AGG() function with the WITHIN GROUP syntax. SQLite uses GROUP_CONCAT() instead, which has different syntax.

In [611…
```
%%sql
select group_concat(item, ',') as summary from items
```

```
 * sqlite:///practice.db
```

Out[611]:

	summary
0	mouse,keyboard,headphone,laptop,pendrive

Interview Questions:

In [612…
```
%%sql
create table if not exists customers(
    customerid integer primary key autoincrement,
    customername varchar(200),
    age integer,
    state varchar(200)
);
```

```
 * sqlite:///practice.db
```

Out[612]: —

In [613…
```
%%sql
create table if not exists orders(
    orderid integer primary key autoincrement,
    customerid integer,
    orderdate date,
    amount integer
);
```

```
 * sqlite:///practice.db
```

Out[613]: —

In [614…
```
%%sql
insert into customers(customername,age,state) values
('Ram',21,'Jharkhand'),
('Shyam',26,'Bihar'),
('Raj',38,'Jharkhand'),
('Rahul',29,'Jharkhand'),
('Suresh',40,'Jharkhand'),
('Ramesh',33,'West Bengal')
```

```
 * sqlite:///practice.db
```

Out[614]: —

In [615…
```
%%sql
insert into orders(customerid,orderdate,amount) values
(1,'19-04-2021',560),
(1,'24-04-2021',3824),
(2,'01-05-2021',613),
(3,'03-05-2021',1399),
(3,'28-05-2021',4391),
(3,'04-06-2021',2877),
(5,'08-04-2021',4748),
(6,'16-03-2021',3352),
(6,'04-05-2021',2072)
```

```
 * sqlite:///practice.db
```

Out[615]: —

In [616…
```
%sql select * from customers
```

```
 * sqlite:///practice.db
```

Out[616]:

	customerid	customername	age	state
0	1	Ram	21	Jharkhand
1	2	Shyam	26	Bihar
2	3	Raj	38	Jharkhand
3	4	Rahul	29	Jharkhand
4	5	Suresh	40	Jharkhand
5	6	Ramesh	33	West Bengal

In [617…
```
%sql select * from orders
```

```
 * sqlite:///practice.db
```

Out[617]:

	orderid	customerid	orderdate	amount
0	1	1	19-04-2021	560
1	2	1	24-04-2021	3824
2	3	2	01-05-2021	613
3	4	3	03-05-2021	1399
4	5	3	28-05-2021	4391
5	6	3	04-06-2021	2877
6	7	5	08-04-2021	4748
7	8	6	16-03-2021	3352
8	9	6	04-05-2021	2072

In [618…
```
%%sql
SELECT orderid,orderdate, SUBSTR(orderdate, 4, 2) AS month, SUBSTR(orderdate, 7, 4) AS year
from orders
```

```
 * sqlite:///practice.db
```

Out[618]:

	orderid	orderdate	month	year
0	1	19-04-2021	04	2021
1	2	24-04-2021	04	2021
2	3	01-05-2021	05	2021
3	4	03-05-2021	05	2021
4	5	28-05-2021	05	2021
5	6	04-06-2021	06	2021
6	7	08-04-2021	04	2021
7	8	16-03-2021	03	2021
8	9	04-05-2021	05	2021

Q1. Write a query to get customer name, count of orders purchased in april 2021 and march 2021

In [619…
```
%%sql
select c.customername, count(c.orderid) as total_orders
from (select * from (select a.customername,a.orderid,a.orderdate,substr(a.orderdate,4,2)
as Month,substr(a.orderdate,7,4) as Year
from (select c.customername,o.orderid,o.orderdate
from customers c inner join orders o on c.customerid=o.customerid) a ) b where
```

```
(b.Month='04' and b.Year='2021') or (b.Month='03' and b.Year='2021') ) c group by c.customername
```

```
 * sqlite:///practice.db
```

Out[619]:

	customername	total_orders
0	Ram	2
1	Ramesh	1
2	Suresh	1

Q2. write a query to get customer names who bought in May 2021 and are from Jharkhand

In [620…

```
%%sql
select b.customername from (select *, substr(a.orderdate,4,2) as Month,substr(a.orderdate,7,4) as Year
from(select c.customername,o.orderid,o.orderdate,c.state
from customers c inner join orders o on c.customerid=o.customerid) a ) b
where b.Month='05' and b.Year='2021' and b.state='Jharkhand'
```

```
 * sqlite:///practice.db
```

Out[620]:

	customername
0	Raj
1	Raj

Q3. write a query to get customer name and their latest order information

In [621…

```
%%sql
select * from (select *,rank() over (partition by a.customername order by a.orderdate desc) as rank_order
from (select c.customername,o.orderid,o.orderdate,c.state
from customers c inner join orders o on c.customerid=o.customerid) a) b where b.rank_order=1
```

```
 * sqlite:///practice.db
```

Out[621]:

	customername	orderid	orderdate	state	rank_order
0	Raj	5	28-05-2021	Jharkhand	1
1	Ram	2	24-04-2021	Jharkhand	1
2	Ramesh	8	16-03-2021	West Bengal	1
3	Shyam	3	01-05-2021	Bihar	1
4	Suresh	7	08-04-2021	Jharkhand	1

Q4. write a query to get top 2 customer id and name based on total transaction value for each month

In [622…

```
%%sql
select * from (select b.customername,b.customerid,b.Month, rank() over (partition by b.Month order by b.amount
as top_customers from (select *, substr(a.orderdate,4,2) as Month,substr(a.orderdate,7,4) as Year
from(select c.customername,c.customerid,o.amount,o.orderid,o.orderdate,c.state
from customers c inner join orders o on c.customerid=o.customerid) a) b ) c where c.top_customers in (1,2)
```

```
 * sqlite:///practice.db
```

Out[622]:

	customername	customerid	Month	top_customers
0	Ramesh	6	03	1
1	Suresh	5	04	1
2	Ram	1	04	2
3	Raj	3	05	1
4	Ramesh	6	05	2
5	Raj	3	06	1

CTEs (Common Table Expression) in SQL:

CTEs (temporary table) is a small subset of the dataset for usability.

In [623…

```
%%sql
create table new_orders(
    order_id integer,
    date date,
    cid
);
```

```
 * sqlite:///practice.db
```

Out[623]: —

In [624…

```
%%sql
insert into new_orders values
(1,'05-08-2020',1),
```

```
(2,'04-08-2020',2),
(3,'03-08-2020',3),
(4,'04-08-2020',1),
(5,'05-08-2020',2),
(6,'05-08-2021',3),
(7,'04-08-2021',1);
```

```
 * sqlite:///practice.db
```

Out[624]: —

In [625…
```
%sql select * from new_orders
```

```
 * sqlite:///practice.db
```

Out[625]:

	order_id	date	cid
0	1	05-08-2020	1
1	2	04-08-2020	2
2	3	03-08-2020	3
3	4	04-08-2020	1
4	5	05-08-2020	2
5	6	05-08-2021	3
6	7	04-08-2021	1

In [626…
```
%%sql
create table order_summary(
    order_id integer,
    amount integer,
    quamtity integer
)
```

```
 * sqlite:///practice.db
```

Out[626]: —

In [627…
```
%%sql
insert into order_summary values
(1,4922,8),
(2,7516,8),
(3,1206,4),
(4,2841,7),
(5,2522,2),
(6,5084,3),
(7,6640,4);
```

```
 * sqlite:///practice.db
```

Out[627]: —

In [628…
```
%sql select * from order_summary
```

```
 * sqlite:///practice.db
```

Out[628]:

	order_id	amount	quamtity
0	1	4922	8
1	2	7516	8
2	3	1206	4
3	4	2841	7
4	5	2522	2
5	6	5084	3
6	7	6640	4

In [629…
```
%%sql
create table new_customers(
    cust_id integer,
    cust_first_name varchar(200),
    cust_last_name varchar(200)
);
```

```
 * sqlite:///practice.db
```

Out[629]: —

In [630…
```
%%sql
insert into new_customers values
(1,'Henry','Brown'),
(2,'James','Williams'),
(3,'Jack','Taylor');
```

```
 * sqlite:///practice.db
```

Out[630]:

In [631…
```
%sql select * from new_customers
```

```
 * sqlite:///practice.db
```

Out[631]:

	cust_id	cust_first_name	cust_last_name
0	1	Henry	Brown
1	2	James	Williams
2	3	Jack	Taylor

In [632…
```
%sql select * from new_orders
```

```
 * sqlite:///practice.db
```

Out[632]:

	order_id	date	cid
0	1	05-08-2020	1
1	2	04-08-2020	2
2	3	03-08-2020	3
3	4	04-08-2020	1
4	5	05-08-2020	2
5	6	05-08-2021	3
6	7	04-08-2021	1

CTE Query

In [633…
```
%%sql
select a.order_id,substr(a.date,7,4) yr,a.cid, concat(b.cust_first_name,' ', b.cust_last_name) full_name
from new_orders a inner join new_customers b on a.cid = b.cust_id
```

```
 * sqlite:///practice.db
```

Out[633]:

	order_id	yr	cid	full_name
0	1	2020	1	Henry Brown
1	2	2020	2	James Williams
2	3	2020	3	Jack Taylor
3	4	2020	1	Henry Brown
4	5	2020	2	James Williams
5	6	2021	3	Jack Taylor
6	7	2021	1	Henry Brown

In [634…
```
%%sql
select c.cid,c.yr,c.full_name,sum(d.amount*d.quamtity) total_sales from
(select a.order_id,substr(a.date,7,4) yr,a.cid, concat(b.cust_first_name,' ', b.cust_last_name) full_name
from new_orders a inner join new_customers b on a.cid = b.cust_id) c inner join order_summary d on
c.order_id=d.order_id group by c.cid,c.yr,c.full_name
```

```
 * sqlite:///practice.db
```

Out[634]:

	cid	yr	full_name	total_sales
0	1	2020	Henry Brown	59263
1	1	2021	Henry Brown	26560
2	2	2020	James Williams	65172
3	3	2020	Jack Taylor	4824
4	3	2021	Jack Taylor	15252

In [635…
```
%%sql

with cte_2021_sales (cid, yr, full_name, total_sales) as (
    select
        c.cid,
        c.yr,
        c.full_name,
        sum(d.amount * d.quamtity) as total_sales
    from (
        select
            a.order_id,
            substr(a.date, 7, 4) as yr,
            a.cid,
            b.cust_first_name || ' ' || b.cust_last_name as full_name
        from new_orders a
```

```
        inner join new_customers b on a.cid = b.cust_id
    ) c
    inner join order_summary d on c.order_id = d.order_id
    group by c.cid, c.yr, c.full_name
)
select * from cte_2021_sales;
```

```
 * sqlite:///practice.db
```

Out[635]:

	cid	yr	full_name	total_sales
0	1	2020	Henry Brown	59263
1	1	2021	Henry Brown	26560
2	2	2020	James Williams	65172
3	3	2020	Jack Taylor	4824
4	3	2021	Jack Taylor	15252

SQL Interview Question

In [636…
```
%%sql

create table users(
    voter_id integer,
    signup_date date
);
```

```
 * sqlite:///practice.db
```

Out[636]: —

In [637…
```
%%sql
insert into users values
(1,'22-09-2009'),
(2,'10-09-2011'),
(3,'21-09-2015');
```

```
 * sqlite:///practice.db
```

Out[637]: —

In [638…
```
%%sql
create table transactions(
    transaction_id integer,
    voter_id integer,
    created_at date,
    updated_at date,
    status varchar(200),
    amount integer
);
```

```
 * sqlite:///practice.db
```

Out[638]: —

In [639…
```
%%sql
insert into transactions values
(1,1,'19-04-2017','21-04-2017','fail',105),
(2,3,'18-12-2019','19-12-2019','success',215),
(3,2,'20-02-2020','23-07-2020','fail',436);
```

```
 * sqlite:///practice.db
```

Out[639]: —

In [640…
```
%sql select * from users
```

```
 * sqlite:///practice.db
```

Out[640]:

	voter_id	signup_date
0	1	22-09-2009
1	2	10-09-2011
2	3	21-09-2015

In [641…
```
%sql select * from transactions
```

```
 * sqlite:///practice.db
```

Out[641]:

	transaction_id	voter_id	created_at	updated_at	status	amount
0	1	1	19-04-2017	21-04-2017	fail	105
1	2	3	18-12-2019	19-12-2019	success	215
2	3	2	20-02-2020	23-07-2020	fail	436

Q1. write a query to find all the transactions done by the most recently signed user

In [642…
```
%%sql
SELECT MAX(
    substr(signup_date, 7, 4) || '-' ||
    substr(signup_date, 4, 2) || '-' ||
    substr(signup_date, 1, 2)
) AS most_recent_signup_date
FROM users;
```

```
 * sqlite:///practice.db
```

Out[642]:

	most_recent_signup_date
0	2015-09-21

In [643…
```
%%sql
UPDATE users
SET signup_date =
    substr(signup_date, 7, 4) || '-' ||
    substr(signup_date, 4, 2) || '-' ||
    substr(signup_date, 1, 2);
```

```
 * sqlite:///practice.db
```

Out[643]: —

In [644…
```
%sql select * from users
```

```
 * sqlite:///practice.db
```

Out[644]:

	voter_id	signup_date
0	1	2009-09-22
1	2	2011-09-10
2	3	2015-09-21

In [645…
```
%%sql
select * from transactions where voter_id in (
select voter_id from users where signup_date in (select max(signup_date) from users))
```

```
 * sqlite:///practice.db
```

Out[645]:

	transaction_id	voter_id	created_at	updated_at	status	amount
0	2	3	18-12-2019	19-12-2019	success	215

Q2. write a query to find transaction_ids of second highest amount transaction done by all users

In [646…
```
%sql select * from transactions
```

```
 * sqlite:///practice.db
```

Out[646]:

	transaction_id	voter_id	created_at	updated_at	status	amount
0	1	1	19-04-2017	21-04-2017	fail	105
1	2	3	18-12-2019	19-12-2019	success	215
2	3	2	20-02-2020	23-07-2020	fail	436

In [647…
```
%%sql
select a.transaction_id,a.amount, rank() over (order by a.amount) rnk from
(select t.voter_id,t.transaction_id,t.amount from users u inner join transactions t on u.voter_id=t.voter_id) a
```

```
 * sqlite:///practice.db
```

Out[647]:

	transaction_id	amount	rnk
0	1	105	1
1	2	215	2
2	3	436	3

Pivot in SQL

comes from pivottable in excel

SQLite doesn't have a built-in PIVOT function like some other SQL databases (SQL Server, Oracle, etc.).

Common Date Part Extractions

-- Year
`SELECT strftime('%Y', date_column) AS year_part FROM your_table;`

-- Month (numeric)

```
SELECT strftime('%m', date_column) AS month_part FROM your_table;
```

-- Day of month

```
SELECT strftime('%d', date_column) AS day_part FROM your_table;
```

-- Hour

```
SELECT strftime('%H', datetime_column) AS hour_part FROM your_table;
```

-- Minute

```
SELECT strftime('%M', datetime_column) AS minute_part FROM your_table;
```

-- Second

```
SELECT strftime('%S', datetime_column) AS second_part FROM your_table;
```

Interview Question

In [648…]
```
# orders table:

# order_id    |    product_id     |    quantity

# ORD1        |    PRD1           |     5
# ORD2        |    PRD2           |     1
# ORD3        |    PRD3           |     3

# -- Write a SQL query which will explode data into single unit level records

# Output we want:

# ORD1        |    PRD1           |     1
# ORD1        |    PRD1           |     1
# ORD1        |    PRD1           |     1
# ORD1        |    PRD1           |     1
# ORD1        |    PRD1           |     1
# ORD2        |    PRD2           |     1
# ORD3        |    PRD3           |     1
# ORD3        |    PRD3           |     1
# ORD3        |    PRD3           |     1

# (opposite of group by)
```

In [649…]
```
%%sql
create table orders1(
    order_id varchar(200),
    product_id varchar(200),
    quantity integer
);
```

```
 * sqlite:///practice.db
```

Out[649]: —

In [650…]
```
%%sql
insert into orders1 values
('ORD1','PRD1',5),
('ORD2','PRD2',1),
('ORD3','PRD3',3);
```

```
 * sqlite:///practice.db
```

Out[650]: —

In [651…]
```
%sql select * from orders1
```

```
 * sqlite:///practice.db
```

Out[651]:

	order_id	product_id	quantity
0	ORD1	PRD1	5
1	ORD2	PRD2	1
2	ORD3	PRD3	3

In [652…]
```
%sql select order_id, product_id,quantity from orders1
```

```
 * sqlite:///practice.db
```

Out[652]:

	order_id	product_id	quantity
0	ORD1	PRD1	5
1	ORD2	PRD2	1
2	ORD3	PRD3	3

In [653…]
```
%sql select order_id,product_id,quantity-1 from orders1 where quantity >=2
```

```
 * sqlite:///practice.db
```

Out[653]:

	order_id	product_id	quantity-1
0	ORD1	PRD1	4
1	ORD3	PRD3	2

In [654…

```
%%sql
with cte as (

select order_id, product_id,quantity from orders1
union all
select order_id,product_id,quantity-1 from cte where quantity >=2

)

select order_id,product_id,quantity, case when quantity>=2 then 1 else quantity end as Exploded_Quantity
from cte order by order_id
```

```
 * sqlite:///practice.db
```

Out[654]:

	order_id	product_id	quantity	Exploded_Quantity
0	ORD1	PRD1	5	1
1	ORD1	PRD1	4	1
2	ORD1	PRD1	3	1
3	ORD1	PRD1	2	1
4	ORD1	PRD1	1	1
5	ORD2	PRD2	1	1
6	ORD3	PRD3	3	1
7	ORD3	PRD3	2	1
8	ORD3	PRD3	1	1

SQL Interview Question

In [655…

```
# activity table:

# player_id   |   device_id    |   event_date    |  games_played

# 1           |    2           |   2016-03-01    |   5
# 1           |    2           |   2016-05-02    |   6
# 2           |    3           |   2017-06-25    |   1
# 3           |    1           |   2016-03-02    |   0
# 3           |    4           |   2018-07-03    |   5
```

In [656…

```
%%sql
create table activity(
    player_id integer,
    device_id integer,
    event_date date,
    games_played integer
)
```

```
 * sqlite:///practice.db
```

Out[656]: —

In [657…

```
%%sql
insert into activity values
(1,2,'2016-03-01',5),
(1,2,'2016-05-02',6),
(2,3,'2017-06-25',1),
(3,1,'2016-03-02',0),
(3,4,'2018-07-03',5);
```

```
 * sqlite:///practice.db
```

Out[657]: —

In [658…

```
%sql select * from activity
```

```
 * sqlite:///practice.db
```

Out[658]:

	player_id	device_id	event_date	games_played
0	1	2	2016-03-01	5
1	1	2	2016-05-02	6
2	2	3	2017-06-25	1
3	3	1	2016-03-02	0
4	3	4	2018-07-03	5

Q. Write a SQL query that reports the device that is first logged in for each player

In [659…
```
%%sql
select device_id,event_date from activity where event_date in
(select distinct(a.first_log) from (select device_id, max(event_date)
over (partition by player_id) as first_log from activity) a)
```

```
 * sqlite:///practice.db
```

Out[659]:

	device_id	event_date
0	2	2016-05-02
1	3	2017-06-25
2	4	2018-07-03

In [660…
```
%sql select * from activity
```

```
 * sqlite:///practice.db
```

Out[660]:

	player_id	device_id	event_date	games_played
0	1	2	2016-03-01	5
1	1	2	2016-05-02	6
2	2	3	2017-06-25	1
3	3	1	2016-03-02	0
4	3	4	2018-07-03	5

-- Write an SQL query that reports for each player and date, how many games played so far by the player. That is, the total number of games played by the player until that date.

In [661…
```
%%sql
select *, sum(games_played) over (partition by player_id order by event_date asc rows between unbounded precedi
and current row) games_layed_so_far from activity
```

```
 * sqlite:///practice.db
```

Out[661]:

	player_id	device_id	event_date	games_played	games_layed_so_far
0	1	2	2016-03-01	5	5
1	1	2	2016-05-02	6	11
2	2	3	2017-06-25	1	1
3	3	1	2016-03-02	0	0
4	3	4	2018-07-03	5	5

SQL Interview Question

In [662…
```
%%sql
create table employees1(
    employee_name char,
    employee_salary integer
);
```

```
 * sqlite:///practice.db
```

Out[662]: —

In [663…
```
%%sql
insert into employees1 values
('A',24000),
('C',30000),
('D',12000),
('E',21000),
('F',13000);
```

```
 * sqlite:///practice.db
```

Out[663]: —

In [664…
```
%sql select * from employees1
```

```
 * sqlite:///practice.db
```

Out[664]:

	employee_name	employee_salary
0	A	24000
1	C	30000
2	D	12000
3	E	21000
4	F	13000

-- Write the SQL query to get the third maximum salary of an employee

In [665…
```
%%sql
select a.employee_name,min(a.employee_salary) from
((select * from employees1 order by employee_salary desc limit 3) a)
```

```
 * sqlite:///practice.db
```

Out[665]:

	employee_name	min(a.employee_salary)
0	E	21000

In [666…
```
%%sql
select * from (select *, rank() over(order by employee_salary desc) rnk from employees1) e where e.rnk=3
```

```
 * sqlite:///practice.db
```

Out[666]:

	employee_name	employee_salary	rnk
0	E	21000	3

SQL Interview Question

In [667…
```
%%sql
create table mails(
    ename char,
    email varchar(200)
);
```

```
 * sqlite:///practice.db
```

Out[667]: —

In [668…
```
%%sql
insert into mails values
('A','fdc@email.com'),
('C','fdoos@email.com');
```

```
 * sqlite:///practice.db
```

Out[668]: —

In [669…
```
%sql select * from mails
```

```
 * sqlite:///practice.db
```

Out[669]:

	ename	email
0	A	fdc@email.com
1	C	fdoos@email.com

-- Get the name before the @ sign in email id

In [670…
```
%%sql
--select left(email,charindex('@')-1) from mails
SELECT SUBSTR(email, 1, INSTR(email, '@') - 1) AS username
FROM mails;
```

```
 * sqlite:///practice.db
```

Out[670]:

	username
0	fdc
1	fdoos

-- Get the domain name after the @ sign in email id:

In [671…
```
%%sql
-- select right(email,len(email)-charindex('@',email)) from tablename
select substr(email,instr(email,'@')+1,length(email)) domain_name from mails
```

```
 * sqlite:///practice.db
```

Out[671]:

	domain_name
0	email.com
1	email.com

SQL Interview Question

In [672…
```
# id     |   name     |   salary   | managerid

# 1      |   Joe      |   70000    |  3
# 2      |   Henry    |   80000    |  4
# 3      |   Sam      |   60000    |  null
```

In [673…
```
%%sql
create table if not exists manager(
    id integer,
    name varchar(200),
    salary integer,
    managerid integer
);
```

```
 * sqlite:///practice.db
```

Out[673]: —

In [674…
```
%%sql
insert into manager values
(1,'Joe',70000,3),
(2,'Henry',80000,4),
(3,'Sam',60000,NULL);
```

```
 * sqlite:///practice.db
```

Out[674]: —

In [675…
```
%sql select * from manager
```

```
 * sqlite:///practice.db
```

Out[675]:

	id	name	salary	managerid
0	1	Joe	70000	3.0
1	2	Henry	80000	4.0
2	3	Sam	60000	NaN

-- Find employees who earn more than managers

In [676…
```
%%sql
select e.id,e.name,e.salary,e.managerid,m.id,m.name,m.salary,m.managerid from manager e
inner join manager m on e.managerid=m.id and e.salary > m.salary
```

```
 * sqlite:///practice.db
```

Out[676]:

	id	name	salary	managerid	id	name	salary	managerid
0	1	Joe	70000	3	3	Sam	60000	None

SQL Interview Question

In [677…
```
%%sql
create table if not exists customers2(
    id integer,
    customer varchar(200)
);
```

```
 * sqlite:///practice.db
```

Out[677]: —

In [678…
```
%%sql
insert into customers2 values
(1,'n1'),
(2,'n2'),
(3,'n3'),
(4,'n4');
```

```
 * sqlite:///practice.db
```

Out[678]: —

In [679…
```
%%sql
create table if not exists orders2(
    id integer,
    orderid integer
);
```

```
 * sqlite:///practice.db
```
Out[679]: —

In [680…
```
%%sql
insert into orders2 values
(1,2),
(2,1);
```
```
 * sqlite:///practice.db
```
Out[680]: —

In [681…
```
%sql select * from customers2
```
```
 * sqlite:///practice.db
```
Out[681]:

	id	customer
0	1	n1
1	2	n2
2	3	n3
3	4	n4

In [682…
```
%sql select * from orders2
```
```
 * sqlite:///practice.db
```
Out[682]:

	id	orderid
0	1	2
1	2	1

-- Write an SQL query to report all customers who never order anything

In [683…
```
%%sql
select id,customer from customers2 where id not in (select id from orders2)
```
```
 * sqlite:///practice.db
```
Out[683]:

	id	customer
0	3	n3
1	4	n4

In [684…
```
%%sql
select a.id,a.customer,b.orderid from customers2 a left join orders2 b on a.id=b.id
```
```
 * sqlite:///practice.db
```
Out[684]:

	id	customer	orderid
0	1	n1	2.0
1	2	n2	1.0
2	3	n3	NaN
3	4	n4	NaN

In [685…
```
%%sql
select id,customer from
(select a.id,a.customer,b.orderid from customers2 a left join orders2 b on a.id=b.id) a where
orderid is null
```
```
 * sqlite:///practice.db
```
Out[685]:

	id	customer
0	3	n3
1	4	n4

SQL Interview Question

In [686…
```
%%sql
create table if not exists dept_salary(
    id integer,
    name varchar(200),
    salary integer,
    departmentid integer
)
```
```
 * sqlite:///practice.db
```
Out[686]: —

In [687…
```
%%sql
```

In [687…
```
%%sql
insert into dept_salary values
(1,'n1',85000,1),
(2,'n2',80000,1),
(3,'n3',60000,1),
(4,'n4',81000,1),
(5,'n5',90000,1),
(6,'n6',69000,1),
(7,'n7',70000,1);
```

```
 * sqlite:///practice.db
```

Out[687]: —

In [688…
```
%sql select * from dept_salary
```

```
 * sqlite:///practice.db
```

Out[688]:

	id	name	salary	departmentid
0	1	n1	85000	1
1	2	n2	80000	1
2	3	n3	60000	1
3	4	n4	81000	1
4	5	n5	90000	1
5	6	n6	69000	1
6	7	n7	70000	1

-- write a query for who earns the most money in each department. A high earner in a department is someone who earns one of the department's top three highest salary.

In [689…
```
%%sql
select * from (select *,rank() over (partition by departmentid order by salary desc) rnk from dept_salary) a
where a.rnk=1
```

```
 * sqlite:///practice.db
```

Out[689]:

	id	name	salary	departmentid	rnk
0	5	n5	90000	1	1

SQL Interview Question

In [690…
```
# name   |   salary

# n1     |   2831
# n2     |   1988
# n3     |   914
```

In [691…
```
%%sql
create table if not exists deviation(
    name varchar(200),
    salary integer
);
```

```
 * sqlite:///practice.db
```

Out[691]: —

In [692…
```
%%sql
insert into deviation values
('n1',2831),
('n2',1988),
('n3',914);
```

```
 * sqlite:///practice.db
```

Out[692]: —

In [693…
```
%sql select * from deviation
```

```
 * sqlite:///practice.db
```

Out[693]:

	name	salary
0	n1	2831
1	n2	1988
2	n3	914

-- Write a query to find out the deviation from average salary for the employesss who are getting more than average salary

why below query is wrong

In [694…
```sql
%%sql
select *,salary-avg(salary) as dev from deviation where salary > (select avg(salary) from deviation)
```

```
 * sqlite:///practice.db
```

Out[694]:

	name	salary	dev
0	n1	2831	421.5

In [695…
```sql
%%sql
select * from (select *, salary - avg(salary) over (rows between unbounded preceding and unbounded following)
as avg_salary_deviation  from deviation) a where a.salary > (select avg(salary) from deviation)
```

```
 * sqlite:///practice.db
```

Out[695]:

	name	salary	avg_salary_deviation
0	n1	2831	920.0
1	n2	1988	77.0

In [696…
```sql
%%sql
select name,salary, avg(salary) over () avg_salary from deviation
```

```
 * sqlite:///practice.db
```

Out[696]:

	name	salary	avg_salary
0	n1	2831	1911.0
1	n2	1988	1911.0
2	n3	914	1911.0

In [697…
```sql
%%sql
select *, salary-c.avg_salary deviation from
(select name,salary, avg(salary) over () avg_salary from deviation) c where c.salary > c.avg_salary
```

```
 * sqlite:///practice.db
```

Out[697]:

	name	salary	avg_salary	deviation
0	n1	2831	1911.0	920.0
1	n2	1988	1911.0	77.0

In [698…
```sql
## update query using case:

# update tablename set name=

# case when name='AS' then 'ASSAM'
# when name='BR' then 'BIHAR'
# when name='GA' then 'GOA'
# when name='GJ' then 'GUJARAT'
# when name='HR' then 'HARYANA'
# end;
```

SQL Interview Question

In [699…
```
# id    |   name    |   gender    |   salary

# 1     |   A       |   m         |   2500
# 2     |   B       |   f         |   1500
# 3     |   C       |   m         |   5500
# 4     |   D       |   f         |   500
```

In [700…
```sql
%%sql
create table if not exists gender(
    id integer,
    name varchar(200),
    gender char,
    salary integer
);
```

```
 * sqlite:///practice.db
```

Out[700]: —

In [701…
```sql
%%sql
insert into gender values
(1,'A','m',2500),
(2,'B','f',1500),
(3,'C','m',5500),
(4,'D','f',500);
```

```
 * sqlite:///practice.db
```

Out[701]: —

In [702…
```
%sql select * from gender
```

```
 * sqlite:///practice.db
```

Out[702]:

	id	name	gender	salary
0	1	A	m	2500
1	2	B	f	1500
2	3	C	m	5500
3	4	D	f	500

-- Write an SQL query to swap all 'f' and 'm' values with a single update statement.

In [703…
```
%%sql
update gender set gender=case when gender='m' then 'f' else 'm' end ;
```

```
 * sqlite:///practice.db
```

Out[703]: —

In [704…
```
%sql select * from gender
```

```
 * sqlite:///practice.db
```

Out[704]:

	id	name	gender	salary
0	1	A	f	2500
1	2	B	m	1500
2	3	C	f	5500
3	4	D	m	500

In [705…
```
%%sql
SELECT julianday('2023-12-31') - julianday('2023-01-01') AS days_between;
```

```
 * sqlite:///practice.db
```

Out[705]:

	days_between
0	364.0

SQL Interview Question

In [706…
```
# order id   |  user id    |  order date    |  sale

# 1          |  1          |  1/3/2023      |  8363
# 2          |  2          |  1/15/2023     |  9196
# 3          |  3          |  1/10/2023     |  9663
# 4          |  1          |  2/3/2023      |  2639
```

In [707…
```
%sql drop table if exists order3
```

```
 * sqlite:///practice.db
```

Out[707]: —

In [708…
```
%%sql
create table if not exists order3(
    orderid integer,
    userid integer,
    order_date date,
    sale integer
)
```

```
 * sqlite:///practice.db
```

Out[708]: —

In [709…
```
%%sql
insert into order3 values
(1,1,'2023-01-03',8363),
(2,3,'2023-01-15',9196),
(3,3,'2023-01-10',9663),
(4,1,'2023-02-03',2339);
```

```
 * sqlite:///practice.db
```

Out[709]: —

In [710…
```
%sql select * from order3
```

```
 * sqlite:///practice.db
```

Out[710]:

	orderid	userid	order_date	sale
0	1	1	2023-01-03	8363
1	2	3	2023-01-15	9196
2	3	3	2023-01-10	9663
3	4	1	2023-02-03	2339

-- Write a query to identify returning active users. A returning active user is a user who has made a second purchase within 7 days of any other of their purchases

In [711…
```
%%sql
SELECT a.*, b.*
FROM order3 a
INNER JOIN order3 b ON a.userid = b.userid
WHERE a.order_date < b.order_date  -- Ensure b is later than a
AND julianday(b.order_date) - julianday(a.order_date) < 7
```

```
 * sqlite:///practice.db
```

Out[711]:

	orderid	userid	order_date	sale	orderid	userid	order_date	sale
0	3	3	2023-01-10	9663	2	3	2023-01-15	9196

In [712…
```
# id    |    revenue    |   mnth

# 1     |    8000       |   Jan
# 2     |    9000       |   Jan
# 3     |    10000      |   Feb
# 1     |    7000       |   Feb
# 1     |    6000       |   Mar
```

In [713…
```
%%sql
create table if not exists revenue(
    id integer,
    revenue integer,
    mnth month
);
```

```
 * sqlite:///practice.db
```

Out[713]:

In [714…
```
%%sql
insert into revenue values
(1,8000,'Jan'),
(2,9000,'Jan'),
(3,10000,'Feb'),
(1,7000,'Feb'),
(1,6000,'Mar');
```

```
 * sqlite:///practice.db
```

Out[714]:

In [715…
```
%sql select * from revenue
```

```
 * sqlite:///practice.db
```

Out[715]:

	id	revenue	mnth
0	1	8000	Jan
1	2	9000	Jan
2	3	10000	Feb
3	1	7000	Feb
4	1	6000	Mar

In [716…
```
%%sql
select id,
max(case when mnth='Jan' then revenue end) as jan_r,
max(case when mnth='Feb' then revenue end) as feb_r,
max(case when mnth='Mar' then revenue end) as mar_r
from revenue group by id
```

```
 * sqlite:///practice.db
```

Out[716]:

	id	jan_r	feb_r	mar_r
0	1	8000.0	7000.0	6000.0
1	2	9000.0	NaN	NaN
2	3	NaN	10000.0	NaN

SQL Interview Question

In [717…
```
# date_id    |    make_name    |   lead_id     |   partner_id

# 2020-12-8  |    toyota       |   0           |   1
# 2020-12-8  |    toyota       |   1           |   0
# 2020-12-8  |    toyota       |   1           |   2
# 2020-12-7  |    toyota       |   0           |   2
# 2020-12-7  |    honda        |   0           |   1
# 2020-12-8  |    honda        |   0           |   1
```

In [718…
```
%%sql
create table DailySales(
    date_id date,
    make_name varchar(200),
    lead_id integer,
    partner_id
);
```

```
 * sqlite:///practice.db
```

Out[718]: —

In [719…
```
%%sql
insert into DailySales values
('2020-12-08','toyota',0,1),
('2020-12-08','toyota',1,0),
('2020-12-08','toyota',1,2),
('2020-12-07','toyota',0,2),
('2020-12-08','honda',0,1),
('2020-12-08','honda',0,1);
```

```
 * sqlite:///practice.db
```

Out[719]: —

In [720…
```
%sql select * from DailySales
```

```
 * sqlite:///practice.db
```

Out[720]:

	date_id	make_name	lead_id	partner_id
0	2020-12-08	toyota	0	1
1	2020-12-08	toyota	1	0
2	2020-12-08	toyota	1	2
3	2020-12-07	toyota	0	2
4	2020-12-08	honda	0	1
5	2020-12-08	honda	0	1

Write an SQL query that will, for each date_id and make_name,return the number of distinct lead_id's and distinct partner_id's

In [721…
```
%%sql
select date_id,make_name,count(distinct lead_id) as number_of_distinct_lead_id,
count(distinct partner_id) as number_of_distinct_partner_id
from DailySales group by date_id,make_name
```

```
 * sqlite:///practice.db
```

Out[721]:

	date_id	make_name	number_of_distinct_lead_id	number_of_distinct_partner_id
0	2020-12-07	toyota	1	1
1	2020-12-08	honda	1	1
2	2020-12-08	toyota	2	3

CASE WHEN END statements with Aggregates group by in SQL

In [722…
```
# sid    |    marks

# 1      |    72
# 2      |    16
# 3      |    69
# 4      |    43
# 5      |    23

# output:

# 63-100 -- Excellent
# 33-62  -- pass
# 0-32   -- Fail
```

In [723…
```
%%sql
create table marks(
```

```
    sid integer,
    marks integer
)
```

```
 * sqlite:///practice.db
```

Out[723]: —

In [724…
```
%%sql
insert into marks values
(1,72),
(2,16),
(3,69),
(4,43),
(5,23);
```

```
 * sqlite:///practice.db
```

Out[724]: —

In [725…
```
%%sql
select *, case when marks>=63 and marks<=100 then 'Excellent' when marks>=33 and marks<=62 then 'Pass'
else 'Fail' end as Status from marks
```

```
 * sqlite:///practice.db
```

Out[725]:

	sid	marks	Status
0	1	72	Excellent
1	2	16	Fail
2	3	69	Excellent
3	4	43	Pass
4	5	23	Fail

In [726…
```
# Dataset:

# orderid    |   stateid    |   status   | amount

# 1          |    s1        |   shipped  | 67050
# 2          |    s2        |   delivered| 67050
# 3          |    s3        |   packed   | 60050
# 4          |    s2        |   shipped  | 67050
# 5          |    s1        |   shipped  | 67650
```

In [727…
```
%%sql
create table dataset(
    orderid integer,
    stateid varchar(200),
    status varchar(200),
    amount integer
);
```

```
 * sqlite:///practice.db
```

Out[727]: —

In [728…
```
%%sql
insert into dataset values
(1,'s1','shipped',67050),
(2,'s2','delivered',67050),
(3,'s3','packed',60050),
(4,'s2','shipped',67050),
(5,'s1','shipped',67650);
```

```
 * sqlite:///practice.db
```

Out[728]: —

In [729…
```
%sql select * from dataset
```

```
 * sqlite:///practice.db
```

Out[729]:

	orderid	stateid	status	amount
0	1	s1	shipped	67050
1	2	s2	delivered	67050
2	3	s3	packed	60050
3	4	s2	shipped	67050
4	5	s1	shipped	67650

we need to find no of orders shipped,delivered and packed

In [730…
```
%%sql
select stateid,
```

```
count(case when status='shipped' then orderid end) as shipped_orders,
count(case when status='delivered' then orderid end) as delivered_orders,
count(case when status='packed' then orderid end) as packed_orders
 from dataset group by stateid
```

```
 * sqlite:///practice.db
```

Out[730]:

	stateid	shipped_orders	delivered_orders	packed_orders
0	s1	2	0	0
1	s2	1	1	0
2	s3	0	0	1