**Q51.** create table if not exists World(

name varchar(15),

continent varchar(10),

area int,

population int,

gdp bigint,

constraint pk primary key (name)

);

**IN:**

insert into World values('Afghanistan','Asia',652230,25500100,20343000000), ('Albania','Europe',28748,2831741,12960000000),

('Algeria','Africa',2381741,37100000,188681000000),('Andorra','Europe',468,78115,3712000000),

('Angola','Africa',1246700,20609294,100990000000);

**Query:**

select name,population,area from World

where area>=3000000 or population>=25000000;

**Q52.** create table if not exists Customer(

id int, name varchar(8), referee\_id int,

primary key(id) );

**IN:** insert into Customer values(1,'Will',null),(2,'Jane',null),(3,'Alex',2),(4,'Bill',null),

(5,'Zack',1),(6,'Mark',2);

**Query:** select name from Customer

where referee\_id <> 2 or referee\_id is null;

**Q53. -- Table - Customers**

create table if not exists Customers(

id int, name varchar(5),

primary key(id) );

**IN:**

insert into Customers values(1,'Joe'),(2,'Henry'),(3,'Sam'),(4,'Max');

**-- Table – Orders**

create table if not exists Orders(

id int, customerId int,

primary key(id),

foreign key (customerId) references Customers(id) );

**IN:**

insert into Orders values(1,3),(2,1);

**Query:**

select name from Customers

where id not in (select customerId from Orders);

**Q54**. create table if not exists Employee(

employee\_id int, team\_id int,

primary key(employee\_id) );

**IN:** insert into Employee values(1,8),(2,8),(3,8),(4,7),(5,9),(6,9);

**Query:**

select employee\_id,count(team\_id) over(partition by team\_id) as team\_size from Employee

order by employee\_id;

**Q55. -- Table - Person**

create table if not exists Person(

id int, name varchar(10), phone\_number varchar(11),

primary key(id) );

**IN:**

insert into Person values(3,'Jonathan','051-1234567'),(12,'Elvis','051-7654321'),

(1,'Moncef','212-1234567'), (2,'Maroua','212-6523651'),(7,'Meir','972-1234567'),

(9,'Rachel','972-0011100');

**-- Table - Country**

create table if not exists Country(

name varchar(8), country\_code varchar(3),

primary key(country\_code) );

**IN:**

insert into Country values('Peru','051'),('Israel','972'),('Morocco','212'),

('Germany','049'),('Ethiopia','251');

**-- Table - Calls**

create table if not exists Calls(

caller\_id int, callee\_id int, duration int );

**IN:**

insert into Calls values(1,9,33),(2,9,4),(1,2,59),(3,12,102),(3,12,330),(12,3,5),(7,9,13),(7,1,3),

(9,7,1),(1,7,7);

**Query:**

select c.name as country from person p join country c on

left(phone\_number,3) = country\_code

join calls c on p.id in (c.caller\_id, c.callee\_id)

group by co.name

having avg(duration) > (select avg(duration) from calls);

**Q56.**

create table if not exists Activity(

player\_id int, device\_id int,

event\_date date, games\_played int,

primary key(player\_id,event\_date) );

**IN:**

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);

**Query:**

select player\_id, device\_id from activity

where event\_date in (select min(event\_date) over (partition by player\_id) from Activity);

**Q57.**

create table if not exists Orders(

order\_number int, customer\_number int,

primary key(order\_number) );

**IN:**

insert into Orders values(1,1),(2,2),(3,3),(4,3);

**Query:**

select customer\_number,count(\*) from Orderss

group by customer\_number

having count(\*) >= all ( select count(customer\_number) from orders

group by customer\_number);

**Q58.**

create table if not exists Cinema(

seat\_id int not null auto\_increment, free bool,

primary key(seat\_id) );

**IN:**

insert into Cinema(free) values(1),(0),(1),(1),(1);

**Query:**

select distinct t1.seat\_id from cinema as t1 join cinema as t2

on abs(t1.seat\_id - t2.seat\_id) = 1 and t1.free = 1 and t2.free = 1

order by t1.seat\_id ;

**Q59. -- Table - SalesPerson**

create table if not exists SalesPerson(

sales\_id int, name varchar(5), salary int,

commission\_rate int, hire\_date varchar(10),

primary key(sales\_id) );

**IN:**

insert into SalesPerson

values(1,'Joh',100000,6,'4/1/2006'),(2,'Amy',12000,5,'5/1/2010'),

(3,'Mark',65000,12,'12/25/2008'),(4,'Pam',25000,25,'1/1/2005'),(5,'Alex',5000,10,'2/3/2007');

**-- Table - Company**

create table if not exists Company(

com\_id int, name varchar(8), city varchar(10),

primary key(com\_id) );

**IN:**

insert into company values(1,'RED','Boston'),(2,'ORANGE','New York'),(3,'YELLOW','Boston'),

(4,'GREEN','Austin');

**-- Table - Orders**

create table if not exists Orders(

order\_id int, order\_date varchar(10),

com\_id int, sales\_id int, amount int,

primary key(order\_id),

foreign key(com\_id) references Company(com\_id),

foreign key(sales\_id) references SalesPerson(sales\_id) );

**IN:**

insert into orders values(1,'1/1/2014',3,4,10000),(2,'2/1/2014',4,5,5000),

(3,'3/1/2014',1,1,50000),(4,'4/1/2014',1,4,25000);

**Query:**

select name from salesPerson

where sales\_id not in (select sales\_id from orders o join company c

on o.com\_id=c.com\_id and o.com\_id =1);

**Q60.**

create table if not exists Triangle(

x int, y int, z int,

primary key(x,y,z) );

**IN:**

insert into Triangle values(13,15,30),(10,20,15);

**Query:**

select \* ,

case

when ((x>y and x>z and y+z>x) or

(y>x and y>z and x+z>y) or

(z>x and z>y and x+y>z) ) then 'Yes'

else "No"

end as "Traingle"

from triangle;

**Q61.** create table if not exists Point(

x int,

primary key(x) );

**IN:**

insert into Point values(-1),(0),(2);

**Query:**

select min ( abs (e.x - v.x) ) as shortest from Point e, Point v

where abs (e.x - v.x) <> 0;

**Q62.**

create table if not exists ActorDirector(

actor\_id int, director\_id int,

timestamp int,

primary key(timestamp) );

**IN:**

insert into ActorDirector values(1,1,0),(1,1,1),(1,1,2),(1,2,3),(1,2,4),(2,1,5),(2,1,6);

**Query:**

select actor\_id, director\_id from ActorDirector

where actor\_id = director\_id

group by director\_id

having count(actor\_id)>=3;

**Q63. -- Table – Sales**

create table if not exists Sales(

sale\_id int, product\_id int, year int,

quantity int, price int, primary key(sale\_id ,year) );

**IN:**

insert into Sales values (1,100,2008,10,5000), (2,100,2009,12,5000), (7,200,2011,15,9000);

**-- Table - Product**

create table if not exists Product(

product\_id int, product\_name varchar(8),

primary key (product\_id) );

**IN:**

insert into Product values(100,'Nokia'),(200,'Apple'),(300,'Samsung');

**Query:**

select p.product\_name, s.year, s.price from Sales s join Product p

on s.product\_id = p.product\_id;

**Q64. -- Table - Project**

create table if not exists Project(

project\_id int, employee\_id int,

primary key(project\_id, employee\_id) );

**IN:**

insert into Project values(1,1),(1,2),(1,3),(2,1),(2,4);

show tables;

**-- Table - Employee**

create table if not exists Employee(

employee\_id int, name varchar(8),

experience\_years int,

primary key(employee\_id) );

**IN:**

insert into Employee values(1,'Khaled',3),(2,'Ali',2),(3,'John',1),(4,'Doe',2);

**Query:**

select p.project\_id, avg(e.experience\_years) as average\_years from Project p join Employee e on

p.employee\_id = e.employee\_id

group by p.project\_id;

**Q65. -- Table - Product**

create table if not exists Product(

product\_id int, product\_name varchar(6),

unit\_price int,

primary key(product\_id) );

**IN:**

insert into Product values(1,'S8',1000),(2,'G4',800),(3,'iPhone',1400);

show tables;

**-- Table - Sales**

create table if not exists Sales(

seller\_id int, product\_id int,

buyer\_id int, sale\_date date,

quantity int, price int) ;

**IN:**

insert into Sales values(1,1,1,'2019-01-21',2,2000),(1,2,2,'2019-02-17',1,800),

(2,2,3,'2019-06-02',1,800),(3,3,4,'2019-05-13',2,2800);

**Query:**

select seller\_id from sales

group by seller\_id

having sum(price) >= All(select sum(price) over (partition by seller\_id) from sales);

**Q66. Same above table Q65.**

**Query**

select s.buyer\_id from sales s join Product p

on s.product\_id=p.product\_id

where p.product\_id=1;

**Q67.**

create table if not exists Customers(

customer\_id int, name varchar(8),

visited\_on date, amount int,

primary key(customer\_id,visited\_on) );

**IN:**

insert into Customers values (1,'Jhon','2019-01-01',100),

(2,'Daniel','2019-01- 02',110), (3,'Jade','2019-01-03',120),

(4,'Khaled','2019-01-04',130), (5,'Winston','2019-01-05',110), (6,'Elvis','2019-01-06',140),

(7,'Anna','2019-01-07',150), (8,'Maria','2019-01-08',80), (9,'Jaze','2019-01-09',110),

(1,'Jhon','2019-01-10',130), (3,'Jade','2019-01-10',150);

**Query:**

with tmp as (select distinct visited\_on,sum(amount) over(order by visited\_on range

between Interval 6 Day preceding and current row) as amount from customers)

select visited\_on,amount,round(amount/7,2) as average\_amount from tmp t join

(select min(visited\_on) as min\_visited\_on from customerss) as b

on datediff(visited\_on, min\_visited\_on) >= 6;

**Q68.**

create table if not exists Scores(

player\_name varchar(12), gender varchar(2),

day date, score\_points int,

primary key(gender,day) );

**IN:**

insert into Scores values('Aron','F','2020-01-01',17),('Alice','F','2020-01-07',23),

('Bajrang','M','2020-01-07',7),('Khali','M','2019-12-25',11),('Slaman','M','2019-12-30',13),

('Joe','M','2019-12-31',3),('Jose','M','2019-12-18',2),('Priya','F','2019-12-31',23),

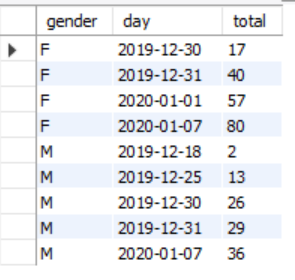
('Priyanka','F','2019-12-30',17);

**Query:**

select gender, day,

sum(score\_points) over(partition by gender order by day) as total

from Scores;



**Q69.**

create table logs(

log\_id int,

primary key(log\_id) );

**Insert:**

insert into logs values(1),(2),(3),(7),(8),(10);

**Query:**

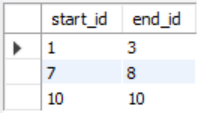
select log\_start.log\_id as start\_id, min(log\_end.log\_id) as End\_id from

(select log\_id from logs where log\_id - 1 not in (select \* from Logs)) log\_start,

(select log\_id from logs where log\_id + 1 not in (select \* from Logs)) log\_end

where log\_start.log\_id <= log\_end.log\_id

group by log\_start.log\_id;



**Q70.** **-- Table - Students**

create table if not exists Students(

student\_id int, student\_name varchar(8),

primary key(student\_id) );

**Insert:**

insert into Students values(1,'Alice'),(2,'Bob'),(13,'John'),(6,'Alex');

**-- Table - Subjects**

create table if not exists Subjects(

subject\_name varchar(12),

primary key(subject\_name) );

**Insert:**

insert into Subjects values('Math'),('Physics'),('Programming');

**-- Table - Examinations**

create table if not exists Examinations(

student\_id int,

subject\_name varchar(12) );

**Insert:**

insert into Examinations values(1,'Math'),(1,'Physics'),(1,'Programming'),(2,'Programming'),

(1,'Physics'),(1,'Math'),(13,'Math'),(13,'Programming'),

(13,'Physics'),(2,'Math'),(1,'Math');

**Query:**

with tmp as (select s.student\_id, s.student\_name, st.subject\_name, count(E.subject\_name) as tot

from Examinations E inner join subjects st on st.subject\_name=e.subject\_name

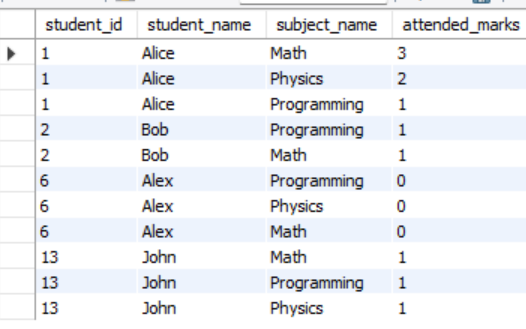
right join students s on e.student\_id=s.student\_id

group by s.student\_id,st.subject\_name,s.student\_name)

select tmp.student\_id, tmp.student\_name, ifnull(tmp.subject\_name , st.subject\_name) as subject\_name, tmp.tot as attended\_marks

from tmp, subjects st

group by tmp.subject\_name, tmp.student\_id, subject\_name, attended\_marks**;**



**Q71.**

create table if not exists Employees(

employee\_id int, employee\_name varchar(10), manager\_id int,

primary key(employee\_id) );

**Insert:**

insert into employees values(1,'Boss',1), (3,'Alice',3), (2,'Bob',1), (4,'Daniel',2),

(7,'Luis',4), (8,'Jhon',3), (9,'Angela',8), (77,'Robert',1);

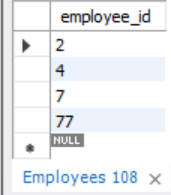
**Query:**

select employee\_id from Employees where manager\_id in

(select employee\_id from Employees WHERE manager\_id in

(select employee\_id from Employees where manager\_id =1))

and employee\_id <>1;



**Q72.**

create table if not exists Transactions(

id int,country varchar(2),

state enum('approved','declined'),

amount int,trans\_date date,

primary key(id) );

**Insert:**

insert into transactions values(121,'US','approved',1000,'2018-12-18'),(122,'US','declined',2000,'2018-12-19'),

(123,'US','approved',2000,'2019-01-01'),(124,'DE','approved',2000,'2019-01-07');

**Query:**

with tmp as (select count(\*) as count, amount, country,

date\_format (trans\_date,'%Y-%m') as `month`

from transactions where state='approved'

group by country, date\_format(trans\_date,'%Y-%m'), amount) ,

tmp\_2 as (select date\_format(trans\_date,'%Y-%m') as `month`, country,

count(country) as trans\_count,

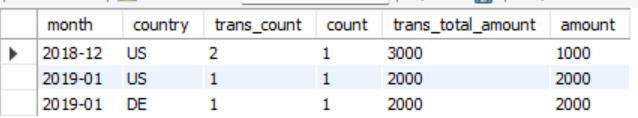
sum(t.amount) as trans\_total\_amount from transactions t

group by `month`,country)

select tmp\_2.`month`, tmp\_2.country, tmp\_2.trans\_count, tmp.count,

tmp\_2.trans\_total\_amount, tmp.amount

from tmp inner join tmp\_2 on tmp.`month`=tmp\_2.`month` and tmp.country=tmp\_2.country;



**Q73.** **-- Table -- Actions**

user\_id int, post\_id int, action\_date date,

create table Actions(

`action` enum('view','like','reaction','comment','report','share'),

extra varchar(6));

**Insert:**

insert into Actions values(1,1,'2019-07-01','view','null'),(1,1,'2019-07-01','like','null'),(1,1,'2019-07-01','share','null'),(2,2,'2019-07-04','view','null'),

(2,2,'2019-07-04','report','spam'),(3,4,'2019-07-04','view','null'),(3,4,'2019-07-04','report','spam'),(4,3,'2019-07-02','view','null'),

(4,3,'2019-07-02','report','spam'),(5,2,'2019-07-03','view','null'),(5,2,'2019-07-03','report','racism'),(5,5,'2019-07-03','view','null'),

(5,5,'2019-07-03','report','racism');

**-- Table -- Removals**

create table Removals(

post\_id int, remove\_date date,

primary key(post\_id) );

**Insert:**

insert into Removals values(2,'2019-07-20'),(3,'2019-07-18');

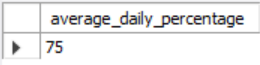
**Query:**

select format(avg ( tmp.deletes\_per\_day), 0) as average\_daily\_percentage from (

select (count(r.post\_id)/count(a.post\_id))\*100 as deletes\_per\_day from actions a left join removals r on a.post\_id = r.post\_id

where a.extra = 'spam'

group by action\_date) tmp ;



**Q74. & Q75.**

create table if not exists Activity(

player\_id int, device\_id int,

event\_date date, games\_played int,

primary key(player\_id,event\_date) );

**IN:**

insert into Activity values(1,2,'2016-03-01',5),(1,2,'2016-03-02',6),(2,3,'2017-06-25',1),

(3,1,'2016-03-02',0),(3,4,'2018-07-03',5);

**Query:**

with cte as (select count(t1.player\_id) as tot from activity t1,activity t2

where datediff(t1.event\_date,t2.event\_date) =1 and

t1.player\_id = t2.player\_id) ,

cte\_2 as (select count(distinct player\_id) as players from activity)

select round(cte.tot/cte\_2.players,2) as fraction from cte,cte\_2;



**Q76.**

create table Salaries(

company\_id int , employee\_id int ,

employee\_name varchar(15) , salary int,

primary key(company\_id, employee\_id) );

**IN:**

insert into Salaries values(1,1,'Tony',2000),(1,2,'Pronub',21300),(1,3,'Tyrrox',10800),(2,1,'Pam',300),

(2,7,'Bassem',450),(2,9,'Hermione',700),(3,7,'Bocaben',100),(3,2,'Ognjen',2200),

(3,13,'Nyan Cat',3300),(3,15,'Morning Cat',7777);

**Query:**

select company\_id , employee\_id , employee\_name,

CASE

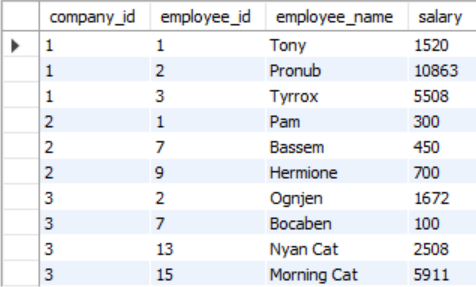
When salary < 1000 then salary

when salary between 1000 and 10000 then round(salary - salary \* 24/100,0)

else round(salary - salary \* 49/100,0)

END as salary

from salaries;



**Q77. -- Table - variables**

create table variables(

name varchar(1) , value int,

primary key(name) );

**IN:**

insert into variables values ('x',66),('y',77);

**-- Table - expressions**

create table expressions(

left\_operand varchar(1),

operator enum('<','>','='),

right\_operand varchar(1),

primary key (left\_operand, operator, right\_operand) );

**IN:**

insert into expressions values('x','>','y'),('x','<','y'),('x','=','y'),('y','>','x'),('y','<','x'),('x','=','x');

**Query:**

SELECT e.left\_operand, e.operator, e.right\_operand,

if(CASE

WHEN e.operator = '>' THEN (v1.value > v2.value)

WHEN e.operator = '<' THEN (v1.value < v2.value)

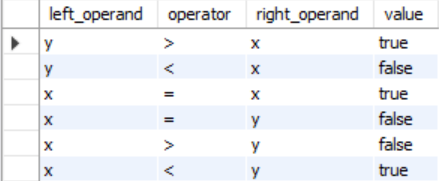
WHEN e.operator = '=' THEN (v1.value = v2.value)

END,'true','false' )AS value

FROM Expressions e

JOIN Variables v1 ON e.left\_operand = v1.name

JOIN Variables v2 ON e.right\_operand = v2.name;



**Q78. Question repeated Q55.**

**Q79.**

create table employee(

employee\_id int, name varchar(15),

months int, salary int );

**IN:**

insert into employee values(12228,'Rose',15,1968),(33645,'Angela',1,3443),(45692,'Frank',17),(56118,'Patrick',7,1345),

(59725,'Lisa',11,2330),(74197,'Kimberly',16,4372),(78454,'Bonnie',8,1771),(83565,'Michael',6,2017),

(98607,'Todd',5,3396),(99989,'Joe',9,3573);

**Query:**

select name from employee order by name;

**Q80.**

create table user\_transactions(

transaction\_id int, product\_id int,

spend double, transaction\_date datetime);

**IN:**

insert into user\_transactions values(1341, 123424, 1500.60, STR\_TO\_DATE('12/31/2019 12:00:00', '%m/%d/%Y %H:%i:%s')),

(1423, 123424, 1000.20, STR\_TO\_DATE('12/31/2020 12:00:00', '%m/%d/%Y %H:%i:%s')),

(1623, 123424, 1246.44, STR\_TO\_DATE('12/31/2021 12:00:00', '%m/%d/%Y %H:%i:%s')),

(1322, 123424, 2145.32, STR\_TO\_DATE('12/31/2022 12:00:00', '%m/%d/%Y %H:%i:%s'));

**Query:**

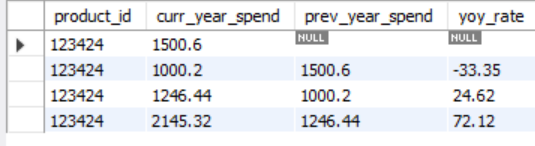
select product\_id,

spend as curr\_year\_spend,

lag(spend) over(order by transaction\_date) as prev\_year\_spend,

round((spend - lag(spend) over(order by transaction\_date))\*100/lag(spend) over(order by transaction\_date),2) as yoy\_rate

from user\_transactions;



**Q81.**

create table amazon\_inventory(

item\_id int, item\_type varchar(25),

item\_category varchar(25), square\_footage double);

**IN:**

insert into amazon\_inventory values(1374,'prime\_eligible','minirefrigerator',68.00),(4245,'not\_prime','standing lamp',26.40),(2452,'prime\_eligible','television',85.00),

(3255,'not\_prime','side table',22.60),(1672,'prime\_eligible','laptop',8.50);

**Query:**

with prime\_tmp as (select floor((500000/sum(square\_footage))) as std,sum(square\_footage) as val

from amazon\_inventory

where item\_type = 'prime\_eligible')

select item\_type,

Case

when item\_type = 'prime\_eligible' then p.std\*count(item\_type)

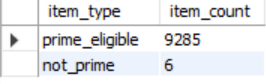
when item\_type = 'not\_prime' then

floor((500000 - (p.std\*p.val))\*count(item\_type)/(sum(square\_footage)))

end as item\_count

from amazon\_inventory,prime\_tmp p

group by item\_type,p.std,p.val;



**Q82.**

create table user\_actions(

user\_id int, event\_id int,

event\_type varchar(10), event\_date datetime);

**IN:**

INSERT INTO user\_actions VALUES

(445,7765,'sign-in',STR\_TO\_DATE('05/31/2022 12:00:00', '%m/%d/%Y %H:%i:%s')),

(742,6458,'sign-in',STR\_TO\_DATE('06/03/2022 12:00:00', '%m/%d/%Y %H:%i:%s')),

(445,3634,'like',STR\_TO\_DATE('06/05/2022 12:00:00', '%m/%d/%Y %H:%i:%s')),

(742,1374,'comment',STR\_TO\_DATE('06/05/2022 12:00:00', '%m/%d/%Y %H:%i:%s')),

(648,3124,'like',STR\_TO\_DATE('06/18/2022 12:00:00', '%m/%d/%Y %H:%i:%s'));

**Query:**

with dir as (select user\_id , extract(month from event\_date) as mon from user\_actions

order by user\_id)

select a.mon , count(a.user\_id) as monthly\_active\_users from dir a join dir b

on a.user\_id = b.user\_id

where a.mon - b.mon = 1

group by a.mon;



**Q84.** **–TABLE - ADVERTISER**

create table advertiser(

user\_id varchar(15),status varchar(15) );

**IN:**

insert into advertiser values('bing','NEW'),('yahoo','NEW'),('alibaba','EXISTING');

**--TABLE -DAILY\_PAY**

create table daily\_pay(

user\_id varchar(10),paid decimal);

**IN:**

insert into daily\_pay values('yahoo',45.00),('alibaba',100.00),('target',13.00);

**QUERY:**

select a.user\_id,

case

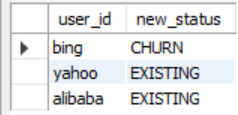
when a.status in ('NEW','EXISTING') and paid is not null then 'EXISTING'

when a.status = 'CHURN' and paid is not null then 'RESURRECT'

else 'CHURN'

END AS new\_status

from advertiser a left join daily\_pay d on a.user\_id = d.user\_id;



**Q85.**

create table server\_utilization(

server\_id int, status\_time timestamp,

session\_status varchar(5));

**IN:**

insert into server\_utilization values(1,STR\_TO\_DATE('08/02/2022 10:00:00', '%m/%d/%Y %H:%i:%s'),'start'),

(1,STR\_TO\_DATE('08/04/2022 10:00:00', '%m/%d/%Y %H:%i:%s'),'stop'),

(2,STR\_TO\_DATE('08/17/2022 10:00:00', '%m/%d/%Y %H:%i:%s'),'start'),

(2,STR\_TO\_DATE('08/24/2022 10:00:00', '%m/%d/%Y %H:%i:%s'),'stop');

**Query:**

SELECT FLOOR(SUM(TIME\_TO\_SEC(TIMEDIFF(stop\_time, start\_time))) / (60\*60\*24)) AS total\_uptime\_days

FROM (

SELECT server\_id, MIN(status\_time) AS start\_time, MAX(status\_time) AS stop\_time

FROM server\_utilization

GROUP BY server\_id

HAVING COUNT(\*) % 2 = 0

) AS server\_uptime;



**Q86.**

create table transactions\_new (

transaction\_id int, merchant\_id int,

credit\_card\_id int, amount int,

transaction\_timestamp datetime);

**IN:**

insert into transactions\_new values (1,101,1,100,str\_to\_date('09/25/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(2,101,1,100,str\_to\_date('09/25/2022 12:08:00','%m/%d/%Y %H:%i:%s')),

(3,101,1,100,str\_to\_date('09/25/2022 12:28:00','%m/%d/%Y %H:%i:%s')),

(4,102,2,300,str\_to\_date('09/25/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(6,102,2,400,str\_to\_date('09/25/2022 14:00:00','%m/%d/%Y %H:%i:%s'));

**Query:**

select count(\*) as payment\_count

FROM ( select \*,

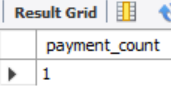
lag(merchant\_id) over(order by credit\_card\_id,transaction\_timestamp) as prev\_merchant,

lag(transaction\_timestamp) OVER (ORDER BY credit\_card\_id,transaction\_timestamp) AS prev\_datetime

FROM transactions\_new ) t

WHERE timstampdiff(minute, prev\_datetime, transaction\_timestamp) between 1 and 10

and prev\_merchant - merchant\_id = 0;



**Q87.** **-- Table - door\_orders**

create table door\_orders(

order\_id int, customer\_id int,

trip\_id int, status varchar(30),

order\_timestamp timestamp );

**IN:**

insert into door\_orders values(727424,8472,100463,'completed successfully',str\_to\_date('06/05/2022 09:12:00','%m/%d/%Y %H:%i:%s')),

(242513,2341,100482,'completed incorrectly',str\_to\_date('06/05/2022 14:40:00','%m/%d/%Y %H:%i:%s')),

(141367,1314,100362,'completed incorrectly',str\_to\_date('06/07/2022 15:03:00','%m/%d/%Y %H:%i:%s')),

(582193,5421,100657,'never\_received',str\_to\_date('07/07/2022 15:22:00','%m/%d/%Y %H:%i:%s')),

(253613,1314,100213,'completed successfully',str\_to\_date('06/12/2022 13:43:00','%m/%d/%Y %H:%i:%s'));

**-- Table - trips**

create table trips(

dasher\_id int, trip\_id int,

estimated\_delivery\_timestamp timestamp, actual\_delivery\_timestamp timestamp);

**IN:**

insert into trips values(101,100463,str\_to\_date('06/05/2022 09:42:00','%m/%d/%Y %H:%i:%s'),str\_to\_date('06/05/2022 09:38:00','%m/%d/%Y %H:%i:%s')),

(102,100482,str\_to\_date('06/05/2022 15:10:00','%m/%d/%Y %H:%i:%s'),str\_to\_date('06/05/2022 15:46:00','%m/%d/%Y %H:%i:%s')),

(101,100362,str\_to\_date('06/07/2022 15:33:00','%m/%d/%Y %H:%i:%s'),str\_to\_date('06/07/2022 16:45:00','%m/%d/%Y %H:%i:%s')),

(102,100657,str\_to\_date('07/07/2022 15:52:00','%m/%d/%Y %H:%i:%s'),null),

(103,100213,str\_to\_date('06/12/2022 14:13:00','%m/%d/%Y %H:%i:%s'),str\_to\_date('06/12/2022 14:10:00','%m/%d/%Y %H:%i:%s'));

**-- Table - door\_customers**

create table door\_customers(

customer\_id int, signup\_timestamp timestamp);

**IN:**

insert into door\_customers values(8472,str\_to\_date('05/30/2022 00:00:00','%m/%d/%Y %H:%i:%s')),

(2341,str\_to\_date('06/01/2022 00:00:00','%m/%d/%Y %H:%i:%s')),

(1314,str\_to\_date('06/03/2022 00:00:00','%m/%d/%Y %H:%i:%s')),

(1435,str\_to\_date('06/05/2022 00:00:00','%m/%d/%Y %H:%i:%s')),

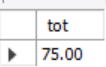
(5421,str\_to\_date('06/07/2022 00:00:00','%m/%d/%Y %H:%i:%s'));

**Query:**

SELECT round((COUNT(CASE WHEN status <> 'completed successfully' THEN 1 ELSE NULL END)/count(\*))\*100,2) as tot

FROM door\_orders o join door\_customers c on o.customer\_id = c.customer\_id

where month(c.signup\_timestamp) = 6;



**Q88. Question repeated Q68.**

**Q89. Question repeated Q78.**

**Q90.**

create table numbers(

num int, frequency int,

primary key (num) );

**IN:**

insert into numbers values(0,7),(1,1),(2,3),(3,1);

**Query:**

with recursive tmp as(

select num,frequency,1 as tag from numbers

union

select num,frequency,tag+1 from tmp where tag < frequency

)

select round(sum(num)/2,0) as median from (select num,row\_number() over(order by num) as r\_num from tmp) as test

where r\_num = (select count(\*) from tmp)/2

or r\_num = (select count(\*) from tmp)/2 +1;



**Q91. –Table- Employee**

create table employee(

employee\_id int, department\_id int,

primary key(employee\_id) );

**IN:**

insert into employee values(1,1),(2,2),(3,2);

**--Table – Salary**

create table salary(

id int, employee\_id int,

amount int, pay\_date date,

primary key(id),

foreign key (employee\_id) references employee(employee\_id) );

**IN:**

insert into salary values(8,1,9000,'2017/03/31'),

(2,2,6000,'2017/03/31'),

(3,3,1000,'2017/03/31'),

(4,1,7000,'2017/02/28'),

(5,2,6000,'2017/02/28'),

(6,3,8000,'2017/02/28');

**Query:**

with tmp as (select date\_format(pay\_date,"%Y-%m") as pay\_month, department\_id,

avg(amount) over(partition by month(pay\_date) ) as average\_1,

avg(amount) over(partition by department\_id order by month(pay\_date) ) as average\_2

from salary s join employee e on s.employee\_id = e.employee\_id)

select pay\_month,department\_id,

case when average\_2 > average\_1 then 'higher'

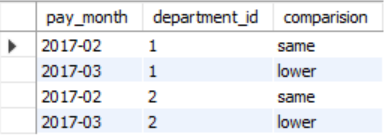
when average\_2 < average\_1 then 'lower'

else 'same'

end as comparision

from tmp

group by pay\_month,department\_id,comparision;



**Q92.**

Table is already created

**Query:**

select install\_date as install\_dt,

count(install\_date) as installs,

round(

count(case when activity\_date = DATE\_ADD(install\_date, interval 1 day) then player\_id end) / count(install\_date),

2

) as Day1\_retention

from

(select player\_id, MIN(event\_date) as install\_date,

case

when max(event\_date) = DATE\_ADD(MIN(event\_date), interval 1 day) then max(event\_date)

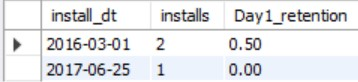
else 0

end AS activity\_date

from Activity

group by player\_id) as subquery

group by install\_date;



**Q93.**

**-- Table - Players**

create table players(

player\_id int primary key,

group\_id int) ;

**IN:**

insert into players(player\_id, group\_id) values(15,1),(25,1),(30,1),(45,1),(10,2),(35,2),(50,2),(20,3),(40,3);

**-- Table - Matches**

create table Matches(

match\_id int primary key,

first\_player int, second\_player int,

first\_score int, second\_score int);

**IN:**

insert into Matches values(1,15,45,3,0),(2,30,25,1,2),(3,30,15,2,0),(4,40,20,5,2),(5,35,50,1,1);

**Query:**

select group\_id, player\_id

from

(SELECT group\_id, player as player\_id, points,

ROW\_NUMBER() OVER (PARTITION BY group\_id ORDER BY points DESC, player ASC) AS row\_num

from

(select group\_id, first\_player as player, sum(first\_score) as points from Matches

join players on player\_id = first\_player

group by group\_id, first\_player

union

select group\_id, second\_player as player, sum(second\_score) as points from Matches

join players on player\_id = second\_player

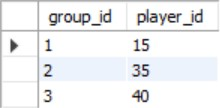
group by group\_id, second\_player

) as tmp

) as result

where row\_num = 1

order by group\_id asc;



**Q94.**

**-- Table - Student**

create table student(

student\_id int primary key,

student\_name varchar(10) );

**IN:**

insert into student values(1,'Daniel'),(2,'Jade'),(3,'Stella'),(4,'Jonathan'),(5,'Will');

**-- Table - Exam**

create table exam(

exam\_id int, student\_id int,

score int,

primary key(exam\_id, student\_id) );

**IN:**

insert into exam values(10,1,70),(10,2,80),(10,3,90),(20,1,80),

(30,1,70),(30,3,80),(30,4,90),(40,1,60),(40,2,70),(40,4,80);

**Query:**

SELECT s.student\_id, s.student\_name

FROM Student s

WHERE s.student\_id IN (

SELECT DISTINCT e.student\_id

FROM Exam e

WHERE e.student\_id NOT IN (

SELECT DISTINCT student\_id

FROM Exam

WHERE (exam\_id, score) IN (

SELECT exam\_id, MIN(score)

FROM Exam

GROUP BY exam\_id

UNION

SELECT exam\_id, MAX(score)

FROM Exam

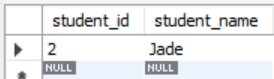
GROUP BY exam\_id

)

)

)

ORDER BY s.student\_id;



**Q95.**

Question 94 is repeated

**Q96.**

**-- Table - songs\_history**

create table songs\_history(

history\_id int, user\_id int,

song\_id int, song\_plays int);

**IN:**

insert into songs\_history values(10011, 777, 1238, 11),(12452, 695, 4520, 1);

**-- Table - songs\_weekly**

create table songs\_weekly(

user\_id int, song\_id int,

listen\_time datetime);

**IN:**

insert into songs\_weekly values(777,1238,str\_to\_date('08/01/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(695,4520,str\_to\_date('08/04/2022 08:00:00','%m/%d/%Y %H:%i:%s')),

(125,9630,str\_to\_date('08/04/2022 16:00:00','%m/%d/%Y %H:%i:%s')),

(695,9852,str\_to\_date('08/07/2022 12:00:00','%m/%d/%Y %H:%i:%s'));

**Query:**

WITH all\_song\_list AS(

SELECT sw.user\_id, sw.song\_id, count(\*) as song\_plays

FROM songs\_weekly sw

WHERE

STR\_TO\_DATE(listen\_time,'%Y-%m-%d') <= STR\_TO\_DATE('08/04/2022','%m/%d/%Y')

GROUP BY sw.user\_id, sw.song\_id

UNION ALL

SELECT sh.user\_id, sh.song\_id, sh.song\_plays

FROM songs\_history sh

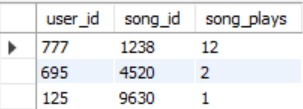
)

SELECT user\_id, song\_id, sum(song\_plays) as song\_plays

FROM all\_song\_list

GROUP BY user\_id,song\_id

ORDER BY song\_plays DESC;



**Q97.**

**-- Table Emails**

create table emails(

email\_id int, user\_id int,

signup\_date datetime);

**In:**

insert into emails values(125,7771,str\_to\_date('06/14/2022 00:00:00','%m/%d/%Y %H:%i:%s')),(236,6950,str\_to\_date('07/01/2022 00:00:00','%m/%d/%Y %H:%i:%s')),

(433,1052,str\_to\_date('07/09/2022 00:00:00','%m/%d/%Y %H:%i:%s'));

**-- Table Texts**

create table texts(

text\_id int, email\_id int,

signup\_action varchar(15) );

**IN:**

insert into texts values(6878,125,'Confirmed'),(6920,236,'Not Confirmed'),(6994,236,'Confirmed');

**Query:**

select round(sum(t.signup\_action = 'Confirmed') / count(\*), 2) as confirm\_rate

from texts t inner join emails e

on t.email\_id = e.email\_id;



**Q98.**

create table tweets(

tweet\_id int, user\_id int,

tweet\_date timestamp);

**IN:**

insert into tweets values(214252,111,str\_to\_date('06/01/2022 12:00:00','%m/%d/%Y %H:%i:%s')),(739252,111,str\_to\_date('06/01/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(846402,111,str\_to\_date('06/02/2022 12:00:00','%m/%d/%Y %H:%i:%s')),(241425,254,str\_to\_date('06/02/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(137374,111,str\_to\_date('06/04/2022 12:00:00','%m/%d/%Y %H:%i:%s'));

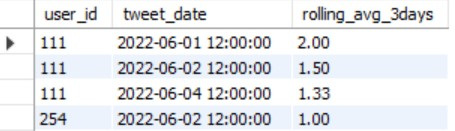
**Query:**

SELECT user\_id,tweet\_date,

ROUND(AVG(COUNT(\*)) OVER (PARTITION BY user\_id ORDER BY tweet\_date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2) AS rolling\_avg\_3days

FROM tweets

GROUP BY user\_id,tweet\_date;



**Q99.**

**-Table - activities**

create table activities(

activity\_id int, user\_id int,

activity\_type enum('send','open','chat'),

time\_spent float,

activity\_date datetime);

**IN:**

insert into activities values(7274,123,'open',4.50,str\_to\_date('06/22/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(2425,123,'send',3.50,str\_to\_date('06/22/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(1413,456,'send',5.67,str\_to\_date('06/23/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(1414,789,'chat',11.00,str\_to\_date('06/25/2022 12:00:00','%m/%d/%Y %H:%i:%s')),

(2536,456,'open',3.00,str\_to\_date('06/25/2022 12:00:00','%m/%d/%Y %H:%i:%s'));

**--Table – age\_breakdown**

create table age\_breakdown(

user\_id int, age\_bucket enum('21-25','26-30','31-35') );

**IN:**

insert into age\_breakdown values(123,'31-35'),(456,'26-30'),(789,'21-25');

**Query:**

select age\_bucket,

ROUND((send\_time / (send\_time + open\_time)) \* 100.0, 2) AS send\_perc,

ROUND((open\_time / (send\_time + open\_time)) \* 100.0, 2) AS open\_perc

FROM ( select ab.age\_bucket,

SUM(CASE WHEN a.activity\_type = 'send' THEN a.time\_spent ELSE 0 END) AS send\_time,

SUM(CASE WHEN a.activity\_type = 'open' THEN a.time\_spent ELSE 0 END) AS open\_time

FROM activities a JOIN

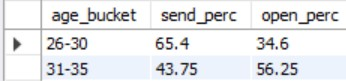
age\_breakdown ab ON a.user\_id = ab.user\_id

where activity\_type in ('open', 'send')

GROUP BY ab.age\_bucket

order by ab.age\_bucket

) AS breakdown;



**Q100.**

**-- Talbe - personal\_profiles**

create table personal\_profiles(

profile\_id int, name varchar(20),

followers int );

**IN:**

insert into personal\_profiles values(1,'Nick Singh', 92000),(2,'Zach Wilson', 199000),(3,'Daliana Liu', 171000),

(4,'Ravit Jain', 107000),(5,'Vin Vashishta', 139000),(6,'Susan Wojcicki', 39000);

**-- Talbe - employee\_company**

create table employee\_company(

personal\_profile\_id int,

company\_id int);

**IN:**

insert into employee\_company values(1,4),(1,9),(2,2),(3,1),(4,3),(5,6),(6,5);

**-- Talbe - company\_pages**

create table company\_pages(

company\_id int, name varchar(35),

followers int);

**IN:**

insert into company\_pages values(1,'The Data Science Podcast',8000),(2,'Airbnb',700000),(3,'The Ravit Show',6000),

(4,'DataLemur',200),(5,'YouTube',16000000),(6,'DataScience.Vin',4500),(9,'Ace The Data Science Interview',4479);

**Query:**

SELECT p.profile\_id

FROM personal\_profiles p

JOIN (

SELECT personal\_profile\_id, MAX(followers) AS max\_followers

FROM employee\_company ec

JOIN company\_pages cp ON ec.company\_id = cp.company\_id

GROUP BY personal\_profile\_id

) AS tmp ON p.profile\_id = tmp.personal\_profile\_id

WHERE p.followers > tmp.max\_followers

ORDER BY p.profile\_id ASC;

