Q51)

Ans-

create table if not exists World

(

name VARCHAR(50),

continent varchar(50),

area int,

population bigint,

gdp bigint,

constraint pk PRIMARY KEY (name)

);

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);

select name,population,area from World where area>=3000000 or population>=25000000;

|  |
| --- |
| --Q52. |
|  | --Table: Customer |
|  | create table if not exists Customer |
|  | ( |
|  | id int, |
|  | name varchar(50), |
|  | referee\_id int, |
|  | constraint pk PRIMARY KEY (id) |
|  | ); |
|  |  |
|  | insert into Customer VALUES (1,'Will',null),(2,'Jane',null),(3,'Alex',2),(4,'Bill',null),(5,'Zack',1),(6,'Mark',2); |
|  | select \* from Customer; |
|  |  |
|  | --Write an SQL query to report the names of the customer that are not referred by the customer with id= 2. |
|  | --Return the result table in any order. |
|  | select name from Customer where referee\_id != 2 or referee\_id is null; |

Q53)

Ans-

|  |
| --- |
|  |
|  | --Table: Customers |
|  | create table if not exists Customers |
|  | ( |
|  | id int, |
|  | name varchar(50), |
|  | constraint pk PRIMARY KEY (id) |
|  | ); |
|  |  |
|  | insert into Customers VALUES (1,'Joe'),(2,'Henry'),(3,'Sam'),(4,'Max'); |
|  |  |
|  | select \* from Customers; |
|  |  |
|  | --Table: Orders |
|  | create table if not exists Orders |
|  | ( |
|  | id int, |
|  | customerId int, |
|  | constraint pk PRIMARY KEY (id), |
|  | constraint fk FOREIGN KEY (customerId) REFERENCES Customers(id) |
|  | ); |
|  |  |
|  | insert into Orders VALUES (1,3),(2,1); |
|  |  |
|  | select \* from Orders; |
|  |  |
|  | --Write an SQL query to report all customers who never order anything. |
|  | --Return the result table in any order. |
|  | SELECT C.Name FROM Customers C LEFT JOIN Orders O ON C.Id = O.CustomerId WHERE O.CustomerId is NULL;  Q54)  Ans-   |  | | --- | | create table if not exists Employee | |  | ( | |  | employee\_id int, | |  | team\_id int, | |  | constraint pk PRIMARY KEY (employee\_id) | |  | ); | |  |  | |  | insert into Employee VALUES (1,8),(2,8),(3,8),(4,7),(5,9),(6,9); | |  |  | |  | select \* from Employee; | |  |  | |  | --Write an SQL query to find the team size of each of the employees. | |  | --Return result table in any order. | |  | SELECT employee\_id, COUNT(team\_id) OVER (PARTITION BY team\_id) team\_size | |  | FROM Employee; | |

Q55)

Ans-

create table if not exists Person

(

id int,

name VARCHAR(50),

phone\_number VARCHAR(50),

constraint pk PRIMARY KEY (id)

);

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');

create table if not exists Country

(

name VARCHAR(50),

country\_code VARCHAR(50),

constraint pk PRIMARY KEY (country\_code)

);

insert into Country VALUES ('Peru',51),('Israel',972),('Morocco',212),('Germany',49),('Ethiopia',251);

create table if not exists Calls

(

caller\_id int,

callee\_id int,

duration int

);

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);

>With joined\_table as (select con.country\_code,con.name,c.duration from Calls c

inner join Person p on c.caller\_id = p.id or c.callee\_id=p.id

inner join Country con on cast(con.country\_code as decimal) = substr(p.phone\_number,1,3))

select name from joined\_table group by name having avg(duration) > (select avg(duration) from joined\_table);

Q56)

create table if not exists Activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

constraint pk PRIMARY KEY (player\_id, event\_date)

);

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);

select player\_id,device\_id from Activity A where event\_date = (select min(B.event\_date) from Activity B where B.player\_id = A.player\_id)

Q57)

Ans-

create table if not exists Orders

(

order\_number int,

customer\_number int,

constraint pk PRIMARY KEY (order\_number)

);

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

* with orders\_sum as (select customer\_number,count(order\_number) order\_count from Orders group

by customer\_number)

select customer\_number from orders\_sum where order\_count = (select max(order\_count) from orders\_sum);

----Another way ----------

>select customer\_number from Orders group by customer\_number order by count(\*) desc limit 1;

Q58)

Ans-

create table if not exists Cinema

(

seat\_id int AUTO\_INCREMENT,

free bool,

constraint pk PRIMARY KEY (seat\_id)

);

insert into Cinema VALUES (1,1),(2,0),(3,1),(4,1),(5,1);

>select \* from Cinema;

>select seat\_id from (select seat\_id,free,coalesce(lag(free,1) over(),0) pre,coalesce(lead(free,1) over(),0) next from Cinema) tmp where free = 1 and (pre = 1 or next = 1) order by seat\_id ;

Q59)

create table if not exists SalesPerson

(

sales\_id int,

name VARCHAR(50),

salary int,

commission\_rate int,

constraint pk PRIMARY KEY (sales\_id)

);

INSERT into SalesPerson VALUES (1,'John',100000,6),(2,'Amy',12000,5),(3,'Mark',65000,12),(4,'Pam',25000,25),(5,'Alex',5000,10);

create table if not exists Company

(

com\_id int,

name VARCHAR(50),

city VARCHAR(50),

constraint pk PRIMARY KEY (com\_id)

);

insert into Company VALUES (1,'RED','Boston'),(2,'ORANGE','New York'),(3,'YELLOW','Boston'),(4,'GREEN','Austin');

create table if not exists Orders

(

order\_id int,

com\_id int,

sales\_id int,

amount int,

constraint pk PRIMARY KEY (order\_id),

constraint fk1 FOREIGN KEY (com\_id) REFERENCES Company(com\_id),

constraint fk FOREIGN KEY (sales\_id) REFERENCES SalesPerson(sales\_id)

);

insert into Orders VALUES (1,3,4,10000),(2,4,5,5000),(3,1,1,50000),(4,1,4,25000);

select name from SalesPerson where sales\_id not in(select sales\_id from Orders o inner join Company c on o.com\_id = c.com\_id and name = 'RED');

Q60)

Ans-

|  |
| --- |
| create table if not exists Triangle |
|  | ( |
|  | x int, |
|  | y int, |
|  | z int, |
|  | constraint pk PRIMARY KEY (x,y,z) |
|  | ); |
|  |  |
|  | insert into Triangle VALUES (13,15,30),(10,20,15); |
|  |  |
|  | select \* from Triangle; |
|  |  |
|  | --Write an SQL query to report for every three line segments whether they can form a triangle. |
|  | --Return the result table in any order. |
|  |  |
|  | SELECT |
|  | x, |
|  | y, |
|  | z, |
|  | CASE WHEN x + y > z AND y + z > x AND z + x > y THEN 'Yes' |
|  | ELSE 'No' END AS triangle |
|  | FROM Triangle; |

Q61)

Ans-

create table Points(x int);

insert into Points values(-1),(0),(1);

select \* from Points;

>select diff from (select abs(x-lag(x) over(order by x desc)) diff from Points)tmp where diff is not null order by diff limit 1;

Q62)

Ans-

|  |
| --- |
| create table if not exists ActorDirector |
|  | ( |
|  | actor\_id int, |
|  | director\_id int, |
|  | timestamp int, |
|  | constraint pk PRIMARY KEY (timestamp) |
|  | ); |
|  |  |
|  |  |
|  | 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); |
|  |  |
|  | select \* from ActorDirector; |
|  |  |
|  | --Write a SQL query for a report that provides the pairs (actor\_id, director\_id) where the actor has cooperated with the director at least three times. |
|  | --Return the result table in any order. |
|  | >SELECT actor\_id, director\_id |
|  | FROM ActorDirector |
|  | GROUP BY actor\_id, director\_id |
|  | HAVING COUNT(\*) >= 3; |

Q63)

Ans-

|  |
| --- |
| --Q63. |
|  |  |
|  | --Table: Product |
|  | create table if not exists Product |
|  | ( |
|  | product\_id int, |
|  | product\_name varchar(50), |
|  | constraint pk PRIMARY KEY (product\_id) |
|  | ); |
|  |  |
|  | insert into Product VALUES (100,'Nokia'),(200,'Apple'),(300,'Samsung'); |
|  |  |
|  | select \* from Product; |
|  |  |
|  | --Table: Sales |
|  | create table if not exists Sales |
|  | ( |
|  | sale\_id int, |
|  | product\_id int, |
|  | year int, |
|  | quantity int, |
|  | price int, |
|  | constraint pk PRIMARY KEY (sale\_id, year), |
|  | constraint fk FOREIGN KEY (product\_id) REFERENCES Product(product\_id) |
|  | ); |
|  |  |
|  | insert into Sales VALUES (1,100,2008,10,5000),(2,100,2009,12,5000),(7,200,2011,15,9000); |
|  |  |
|  | select \* from Sales; |
|  |  |
|  | --Write an SQL query that reports the product\_name, year, and price for each sale\_id in the Sales table. |
|  | --Return the resulting table in any order. |
|  | >select p.product\_name, s.year, s.price |
|  | from Product p |
|  | join Sales s |
|  | on s.product\_id = p.product\_id; |

Q64)

Ans-

|  |
| --- |
|  |
|  | --Table: Employee |
|  | create table if not exists Employee |
|  | ( |
|  | employee\_id int, |
|  | name varchar(50), |
|  | experience\_years int, |
|  | constraint pk PRIMARY KEY (employee\_id) |
|  | ); |
|  |  |
|  | insert into Employee VALUES (1,'Khaled',3),(2,'Ali',2),(3,'John',1),(4,'Doe',2); |
|  |  |
|  | select \* from Employee; |
|  |  |
|  | --Table: Project |
|  | create table if not exists Project |
|  | ( |
|  | project\_id int, |
|  | employee\_id int, |
|  | constraint pk PRIMARY KEY (project\_id, employee\_id), |
|  | constraint fk FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id) |
|  |  |
|  | ); |
|  |  |
|  | insert into Project VALUES (1,1),(1,2),(1,3),(2,1),(2,4); |
|  |  |
|  | select \* from Project; |
|  |  |
|  | --Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits. |
|  | --Return the result table in any order. |
|  | select project\_id , round(avg(experience\_years), 2) as average\_years |
|  | from Project as p |
|  | left join Employee as e |
|  | on p.employee\_id = e.employee\_id |
|  | group by project\_id; |

Q65)

Ans-

>create table if not exists Product

(

product\_id int,

product\_name VARCHAR(50),

unit\_price int,

constraint pk PRIMARY KEY (product\_id)

);

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

>create table if not exists Sales

(

seller\_id int,

product\_id int,

buyer\_id int,

sale\_date date,

quantity int,

price int,

constraint fk FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

>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);

>with Sum\_seller as (select seller\_id,sum(price) total\_sum from Sales group by seller\_id)

select seller\_id from Sum\_seller where total\_sum = (select max(total\_sum) from Sum\_seller);

Q66)

Ans-

create table if not exists Product

(

product\_id int,

product\_name VARCHAR(50),

unit\_price int,

constraint pk PRIMARY KEY (product\_id)

);

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

create table if not exists Sales

(

seller\_id int,

product\_id int,

buyer\_id int,

sale\_date date,

quantity int,

price int,

constraint fk FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

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);

with Joined\_table as(select s.buyer\_id,p.product\_id,p.product\_name from Sales s inner join Product p on p.product\_id = s.product\_id)

select buyer\_id from Joined\_table where product\_name = 'S8' and buyer\_id not in (select buyer\_id from Joined\_table where product\_name = 'iPhone');

Q67)

create table if not exists Customer

(

customer\_id int,

name VARCHAR(50),

visited\_on date,

amount int,

constraint pk PRIMARY KEY (customer\_id, visited\_on)

);

INSERT into Customer 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);

>select visited\_on,sum(amount) over( order by visited\_on range between interval '6' day preceding and current row)/7 as avg

from Customer;

Q68)

Ans-

create table if not exists Scores

(

player\_name VARCHAR(50),

gender varchar(50),

day date,

score\_points int,

constraint pk PRIMARY KEY (gender, day)

);

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);

>select gender,sum(score\_points) over(order by day) from Scores where gender = 'F'

union

select gender,sum(score\_points) over(order by day) from Scores where gender = 'M'

Q69)

create table if not exists Logs

(

log\_id int,

constraint pk PRIMARY KEY (log\_id)

);

insert into Logs VALUES (1),(2),(3),(7),(8),(10);

select \* from Logs;

select min(log\_id) as start\_id, max(log\_id) as end\_id

from (select l.log\_id, (l.log\_id - l.row\_num) as diff

from (select log\_id, row\_number() over() as row\_num from Logs) l

) l2

group by diff;

Q70)

Ans-

create table if not exists Students

(

student\_id int,

student\_name VARCHAR(50),

constraint pk PRIMARY KEY (student\_id)

);

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

create table if not exists Subjects

(

subject\_name VARCHAR(50),

constraint pk PRIMARY KEY (subject\_name)

);

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

create table if not exists Examinations

(

student\_id int,

subject\_name VARCHAR(50)

);

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');

>with Joined\_table as(select s.student\_id,sub.subject\_name,s.student\_name from Students s,Subjects sub)

select j.student\_id,j.subject\_name,count(e.student\_id) from Joined\_table j left outer join Examinations e

on j.student\_id = e.student\_id and j.subject\_name = e.subject\_name

group by j.student\_id,j.subject\_name;

Q71)

Ans-

create table if not exists Employees

(

employee\_id int,

employee\_name VARCHAR(50),

manager\_id int,

constraint pk PRIMARY KEY (employee\_id)

);

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);

with recursive reporting as(

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

union

select e.employee\_id from reporting r inner join Employees e on e.manager\_id = r.employee\_id

)

select \* from reporting;

Q72)

Ans-

create table if not exists Transactions

(

id int,

country VARCHAR(50),

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

amount int,

trans\_date date,

constraint pk PRIMARY KEY (id)

);

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');

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

count(id) total\_count,count(case when state='approved' then id else null end)approved\_count,sum(amount) total\_amount,sum(case when state = 'approved' then amount

else 0 end)approve\_amt from Transactions group by country,date\_format(trans\_date,'%Y-%m')

Q73)

create table if not exists Actions

(

user\_id int,

post\_id int,

action\_date date,

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

extra VARCHAR(50)

);

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');

select \* from Actions;

create table if not exists Removals

(

post\_id int,

remove\_date date,

constraint pk PRIMARY KEY (post\_id)

);

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

select round(avg(daily\_avg),2) avg\_removal from (select count(remove\_date)/count(\*)\*100 daily\_avg from Actions a left outer join Removals r on a.post\_id = r.post\_id

where a.extra = 'spam' group by action\_date) tmp;

Q74)

Ans-

create table if not exists Activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

constraint pk PRIMARY KEY (player\_id, event\_date)

);

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);

select round(count(temp.player\_id)/count(distinct a.player\_id)\*100,2) fraction from Activity a left outer join (select player\_id,min(event\_date) event\_date from Activity group by player\_id) temp on a.player\_id = temp.player\_id

and datediff(a.event\_date,temp.event\_date) = 1;

Q76)

Ans-

create table if not exists Salaries

(

company\_id int,

employee\_id int,

employee\_name VARCHAR(50),

salary int,

constraint pk PRIMARY KEY (company\_id, employee\_id)

);

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);

select s.company\_id,employee\_id,employee\_name,

round((case

when max\_salary<1000 then salary

when max\_salary>=1000 and max\_salary<=10000 then salary - salary\*24/100

else salary - salary\*49/100

end)) salary

from Salaries s

inner join

(select company\_id,max(salary) max\_salary from Salaries

group by company\_id) cms

on s.company\_id = cms.company\_id;

Q77)

Ans-

|  |
| --- |
|  |
|  | --Table Variables: |
|  | create table if not exists Variables |
|  | ( |
|  | name varchar(50), |
|  | value int, |
|  | constraint pk PRIMARY KEY (name) |
|  | ); |
|  |  |
|  | insert into Variables VALUES ('x',66),('y',77); |
|  |  |
|  | select \* from Variables; |
|  |  |
|  | --Table Expressions: |
|  | create table if not exists Expressions |
|  | ( |
|  | left\_operand varchar(50), |
|  | operator enum ('<', '>', '='), |
|  | right\_operand VARCHAR(50), |
|  | constraint pk PRIMARY KEY (left\_operand, operator, right\_operand) |
|  | ); |
|  |  |
|  |  |
|  | insert into Expressions VALUES ('x','>','y'),('x','<','y') ,('x','=','y'),('y','>','x'),('y','<','x'),('x','=','x'); |
|  |  |
|  | select \* from Expressions; |
|  |  |
|  | --Write an SQL query to evaluate the boolean expressions in Expressions table. |
|  | --Return the result table in any order. |
|  |  |
|  | select e.left\_operand, e.operator, e.right\_operand, |
|  | case |
|  | when e.operator = '<' then if(l.value < r.value,'true','false') |
|  | when e.operator = '>' then if(l.value > r.value,'true','false') |
|  | else if(l.value = r.value,'true','false') |
|  | end as value |
|  | from Expressions e |
|  | left join Variables l on e.left\_operand = l.name |
|  | left join Variables r on e.right\_operand = r.name; |

Q80)

Ans-

create table if not exists user\_transactions

(

transaction\_id int,

product\_id int,

spend decimal (8,2),

transaction\_date DATETIME

);

insert into user\_transactions values (1341,123424,1500.60,'2019-12-31 12:00:00'),(1423,123424,1000.20,'2020-12-31 12:00:00'),(1623,123424,1246.44,'2021-12-31 12:00:00'),(1322,123424,2145.32,'2022-12-31 12:00:00');

select

product\_id,transaction\_date,round((spend-prev\_year\_trans)/prev\_year\_trans\*100,2) year\_growth

from (select

transaction\_id,

product\_id,

transaction\_date,

spend,

(lag(u.spend) over(order by transaction\_date)) prev\_year\_trans

from user\_transactions u) temp

Q82)

Ans-

create table if not exists user\_actions

(

user\_id int,

event\_id int,

event\_type enum ("sign-in", "like", "comment"),

event\_date DATETIME

);

insert into user\_actions VALUES (445,7765,'sign-in','2022-05-31 12:00:00'),(742,6458,'sign-in','2022-06-03 12:00:00'),(445,3634,'like','2022-06-05 12:00:00'),(742,1374,'comment','2022-06-05 12:00:00'),(648,3124,'like','2022-06-18 12:00:00');

select extract(month from u2.event\_date) month, count(distinct u2.user\_id) total\_count from user\_actions u1 inner join user\_actions u2 on u1.user\_id = u2.user\_id

and extract(month from u2.event\_date) - extract(month from u1.event\_date) = 1 and date\_format(u2.event\_date,'%y')=date\_format(u1.event\_date,'%y') group by u2.event\_date

Q84)

Ans-

create table if not exists advertiser

(

user\_id VARCHAR(50),

status VARCHAR(50)

);

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

create table if not exists daily\_pay

(

user\_id VARCHAR(50),

paid decimal

);

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

>with joined\_table as (

select ad.user\_id,ad.status,dp.paid from advertiser ad left outer join daily\_pay dp

on ad.user\_id = dp.user\_id

)

select user\_id,'CHURN' status from joined\_table

where paid is null

union

select user\_id,if(status = 'CHURN','RESURRECT','EXISTING') status from joined\_table

where paid is not null;

Q85)

Ans-

create table if not exists server\_utilization

(

server\_id int,

status\_time TIMESTAMP,

session\_status VARCHAR(50)

);

insert into server\_utilization VALUES(1,'2022-08-02 10:00:00','start'),(1,'2022-08-04 10:00:00','stop'),(2,'2022-08-17 10:00:00','start'),(2,'2022-08-24 10:00:00','stop');

with joined\_table as (

select server\_id,

status\_time stop\_time,

session\_status,

lag(status\_time) over(partition by server\_id order by status\_time) start\_time

from server\_utilization

)

select

sum(datediff(stop\_time,start\_time)) running\_time

from joined\_table where session\_status = 'stop';

Q86)

Ans-

create table if not exists transactions

(

transaction\_id int,

merchant\_id int,

credit\_card\_id INT,

amount int,

transaction\_timestamp datetime

);

insert into transactions values (1,101,1,100,'2022-09-25 12:00:00'),(2,101,1,100,'2022-09-25 12:08:00'),(3,101,1,100,'2022-09-25 12:28:00'),(4,102,2,300,'2022-09-25 12:00:00'),(6,102,2,400,'2022-09-25 14:00:00');

select datediff('2022-09-24','2022-09-25');

with cte as (

select \*,

lag(transaction\_timestamp) over(partition by merchant\_id,credit\_card\_id,amount order by transaction\_timestamp) prev\_time

from transactions

)

select count(transaction\_id) duplicates from cte where timestampdiff(Minute,prev\_time,transaction\_timestamp)<10;

Q87)

Ans-

|  |
| --- |
| use sql\_challenge; |
|  | create table orders6( |
|  | order\_id int, |
|  | customer\_id int, |
|  | trip\_id int, |
|  | status enum('completed successfully', 'completed incorrectly', 'never received'), |
|  | order\_timestamp timestamp); |
|  | create table trips( |
|  | dasher\_id int, |
|  | trip\_id int, |
|  | estimated\_delivery\_timestamp timestamp, |
|  | actual\_delivery\_timestamp timestamp); |
|  |  |
|  | create table customer5( |
|  | customer\_id int, |
|  | signup\_timestamp timestamp); |
|  |  |
|  | -- Q87.Write a query to find the bad experience rate in the first 14 days for new users who signed up in June |
|  | -- 2022. Output the percentage of bad experience rounded to 2 decimal places. |
|  |  |
|  | select \* from orders6; |
|  | select \* from customer5; |
|  | select \* from trips; |
|  | with cte as( |
|  | select o.customer\_id,o.status,c.signup\_timestamp,o.order\_timestamp |
|  | from orders6 o inner join trips t on o.trip\_id=t.trip\_id |
|  | inner join customer5 c on c.customer\_id=o.customer\_id |
|  | where extract(year from c.signup\_timestamp)=2022 and month(c.signup\_timestamp)=6 and datediff(o.order\_timestamp,c.signup\_timestamp)<=14 |
|  | ) |
|  | select round((100.0\*count(case when status in('completed incorrectly','never received') then 1 else null end)/count(customer\_id)),2) |
|  | as bad\_experience\_pct |
|  | from cte; |

Q90)

Ans-

create table if not exists Numbers

(

x int,

frequency int,

constraint pk PRIMARY KEY (x)

);

insert into Numbers VALUES (0,7),(1,1),(2,3),(3,1);

with recursive tmp as (

select x,frequency from Numbers

union

select x,frequency-1 from tmp where frequency>1),

cte as (select count(\*) n from tmp),

row\_tmp as (select x,row\_number() over(order by x) term\_no from tmp),

odd\_median as (select x from row\_tmp where term\_no = (select n from cte)/2),

even\_median as (select (x+pre)/2 from (select x,term\_no, lag(x) over() pre from row\_tmp)t where term\_no = (select n from cte)/2 + 1)

select (case

when (select n from cte)%2 = 0 then (select \* from even\_median)

else (select \* from odd\_median)

end )median

from dual;

Q91)

Ans-

create table if not exists Salary

(

id int,

employee\_id int,

amount int,

pay\_date date,

constraint pk PRIMARY KEY (id)

);

insert into Salary VALUES (1,1,9000,'2017/03/31'),(2,2,6000,'2017/03/31'),(3,3,10000,'2017/03/31'),(4,1,7000,'2017/02/28'),(5,2,6000,'2017/02/28'),(6,3,8000,'2017/02/28');

select \* from Salary;

create table if not exists Employee

(

employee\_id int,

department\_id int,

constraint pk PRIMARY KEY (employee\_id)

);

insert into Employee VALUES (1,1),(2,2),(3,2);

select department\_id,pay\_date, case

when department\_avg<total\_avg then 'lower'

when department\_avg>total\_avg then 'higher'

else 'same'

end comparison from (select distinct department\_id,pay\_date, avg(s.amount) over(partition by department\_id,pay\_date) department\_avg, avg(amount) over(partition by pay\_date) total\_avg from Employee e inner join Salary s on e.employee\_id = s.employee\_id) t;

Q92)

Ans-

create table if not exists Activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

constraint pk PRIMARY KEY (player\_id, event\_date)

);

insert INTO Activity VALUES (1,2,'2016-03-01',5),(1,2,'2016-03-02',6),(2,3,'2017-06-25',1),(3,1,'2016-03-01',0),(3,4,'2016-07-03',5);

with first\_login as (

select

player\_id,

min(event\_date) event\_date

from Activity group by player\_id

)

select

fl.event\_date event\_date,

round(count(a.player\_id)/count(fl.player\_id),2) retention

from first\_login fl

left outer join

Activity a on fl.player\_id= a.player\_id and datediff(fl.event\_date,a.event\_date) = -1

group by fl.event\_date;

Q94)

Ans-

create table if not exists Student

(

student\_id int,

student\_name VARCHAR(50),

constraint pk PRIMARY KEY (student\_id)

);

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

create table if not exists Exam

(

exam\_id int,

student\_id int,

score int,

constraint pk PRIMARY KEY (exam\_id, student\_id)

);

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);

Select e.student\_id,s.student\_name from Exam e

inner join

Student s on e.student\_id = s.student\_id

and e.score > (select min(score) from Exam e2 where e2.exam\_id = e.exam\_id group by e2.exam\_id )

and e.score < (select max(score) from Exam e2 where e2.exam\_id = e.exam\_id group by e2.exam\_id )

group by e.student\_id,s.student\_name having count(e.student\_id) = (select count(student\_id) from Exam where student\_id = e.student\_id);

Q96)

Ans-

create table songs\_history(history\_id int,

user\_id int,song\_id int,song\_plays int);

create table songs\_weekly(user\_id int,song\_id int);

insert into songs\_history values(1001,777,1238,11),(12452,695,4520,1);

insert into songs\_weekly values(777,1238),(695,4520),(125,9630)

,(695,9852);

select sw.song\_id,sw.user\_id,(count(\*) + coalesce(sh.song\_plays,0)) c from songs\_weekly sw left outer join songs\_history sh

on sw.user\_id = sh.user\_id and sw.song\_id = sh.song\_id group by sw.song\_id,sw.user\_id,sh.song\_plays;