**Q51. Write an SQL query to report the name, population, and area of the big countries. Return the result table in any order**

create table world

(name varchar(20),

continent varchar(20),

area int,

population int,

gdp bigint,

primary key(name)

)

insert into world values

('Afghanistan', 'Asia' ,652230 ,25500100 ,20343000000),

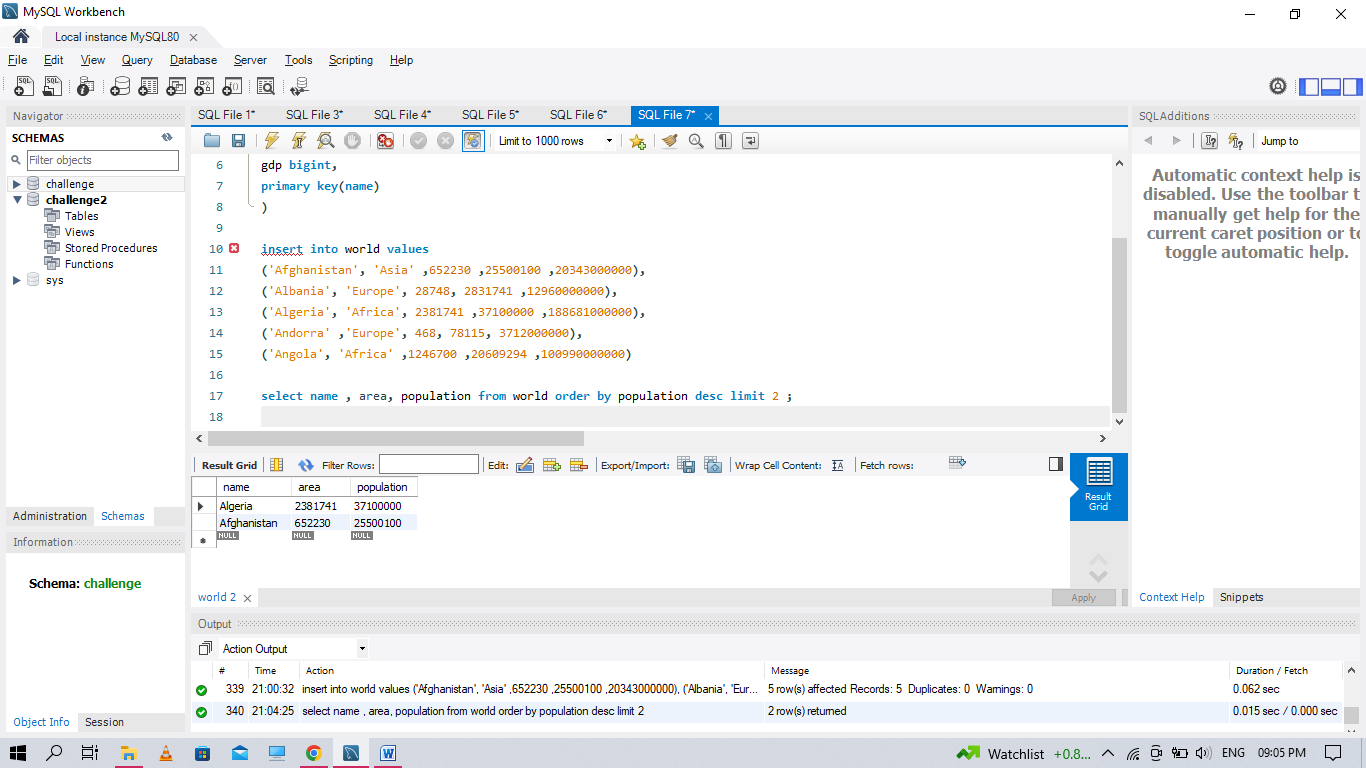
('Albania', 'Europe', 28748, 2831741 ,12960000000),

('Algeria', 'Africa', 2381741 ,37100000 ,188681000000),

('Andorra' ,'Europe', 468, 78115, 3712000000),

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

select name , area, population from world order by population desc limit 2 ;



**Q52. Write an SQL query to report the names of the customer that are not referred by the customer with id = 2**

create table customer

(id int,

name varchar(20),

referee\_id int,

primary key(id)

)

insert into customer values

(1 ,'Will' ,null),

(2 ,'Jane' ,null),

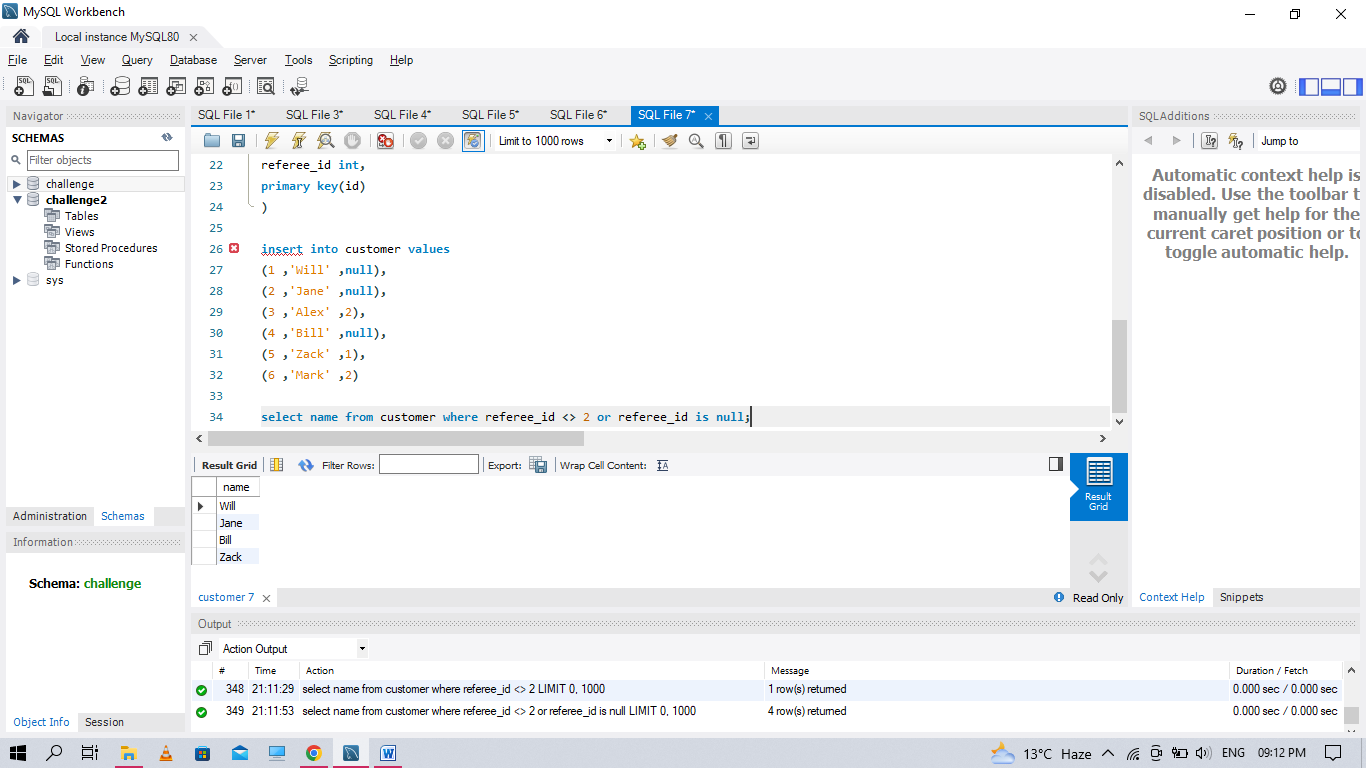
(3 ,'Alex' ,2),

(4 ,'Bill' ,null),

(5 ,'Zack' ,1),

(6 ,'Mark' ,2)

select name from customer where referee\_id <> 2 or referee\_id is null;



**Q53. Write an SQL query to report all customers who never order anything. Return the result table in any order.**

create table customers

(id int,

name varchar(20),

primary key(id))

create table orders

(id int,

customer\_id int,

primary key(id))

insert into customers values

(1 ,'Joe'),

(2,'Henry'),

(3 ,'Sam'),

(4 ,'Max')

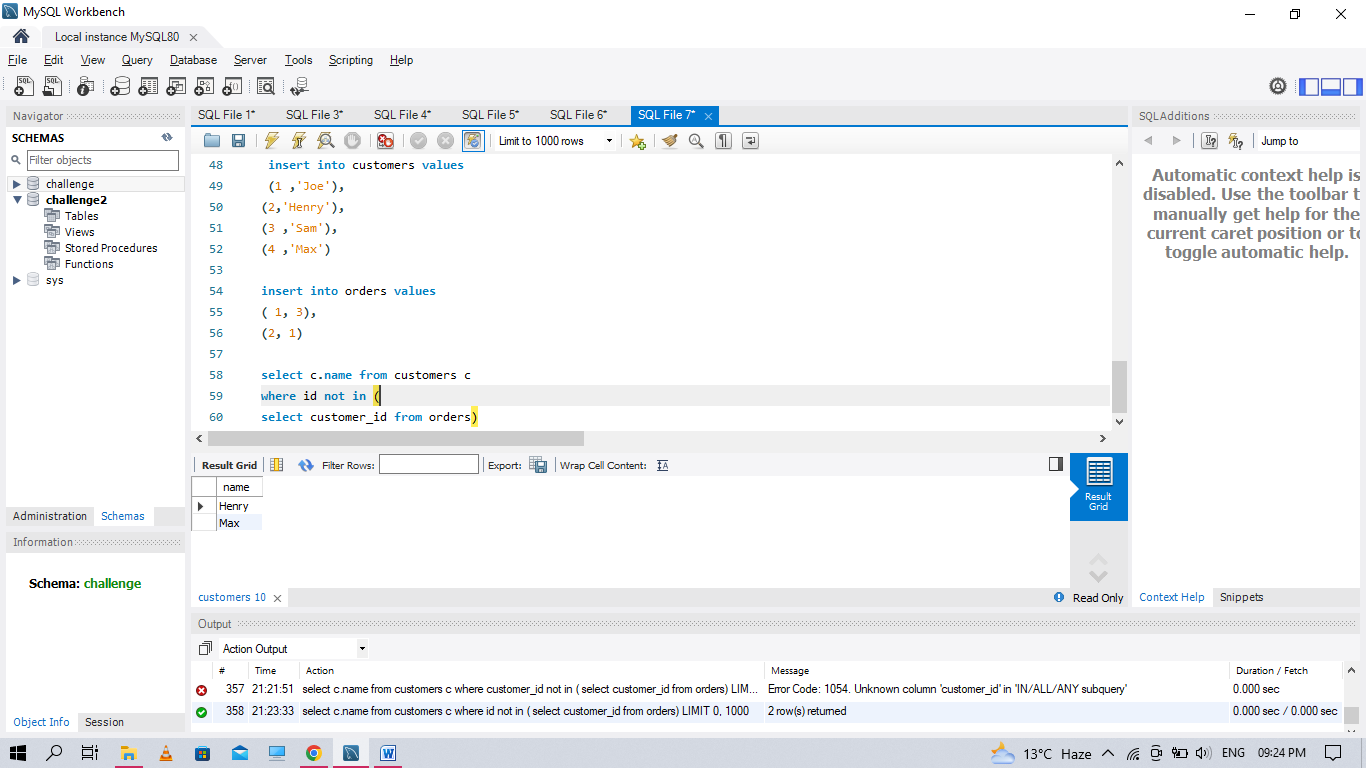
insert into orders values

( 1, 3),(2, 1)

select c.name from customers c

where id not in (

select customer\_id from orders)



**Q54. Write an SQL query to find the team size of each of the employees. Return result table in any order.**

create table employeee

(employee\_id int,

team\_id int,

primary key(employee\_id)

)

insert into employeee values

(1, 8),

(2, 8),

(3, 8),

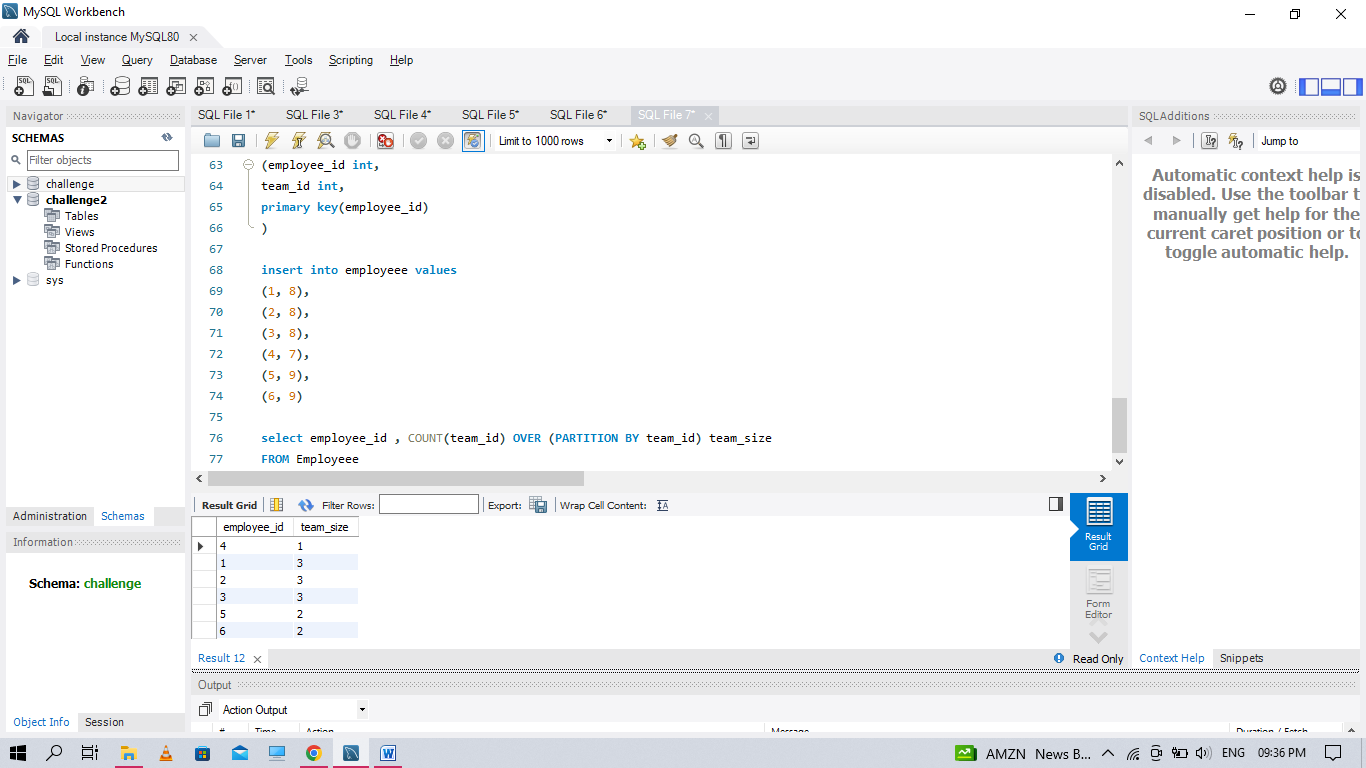
(4, 7),

(5, 9),

(6, 9)

select employee\_id , COUNT(team\_id) OVER (PARTITION BY team\_id) team\_size

FROM Employeee



Q55. Write an SQL query to find the countries where this company can invest. Return the result table in any order.

create table person

(

id int,

name varchar(20),

phone\_number varchar(20),

primary key(id)

)

create table country

(

name varchar(20),

country\_code varchar(20),

primary key(country\_code)

)

create table calls

(caller\_id int,

callee\_id int,

duration int

)

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')

insert into country values

('Peru' ,'51'),

('Israel' ,'972'),

('Morocco' ,'212'),

('Germany' ,'49'),

('Ethiopia' ,'251')

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)

select c.name as country

from Person p

inner join Country c

on left (p.phone\_number,3) = c.country\_code

inner join (select caller\_id as id, duration

from Calls

union all

select callee\_id as id, duration

from Calls) phn

on p.id = phn.id

group by country

having avg(duration) > (select avg(duration) from Calls)

**Q56. Write an SQL query to report the device that is first logged in for each player. Return the result table in any order.**

create table Activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

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),

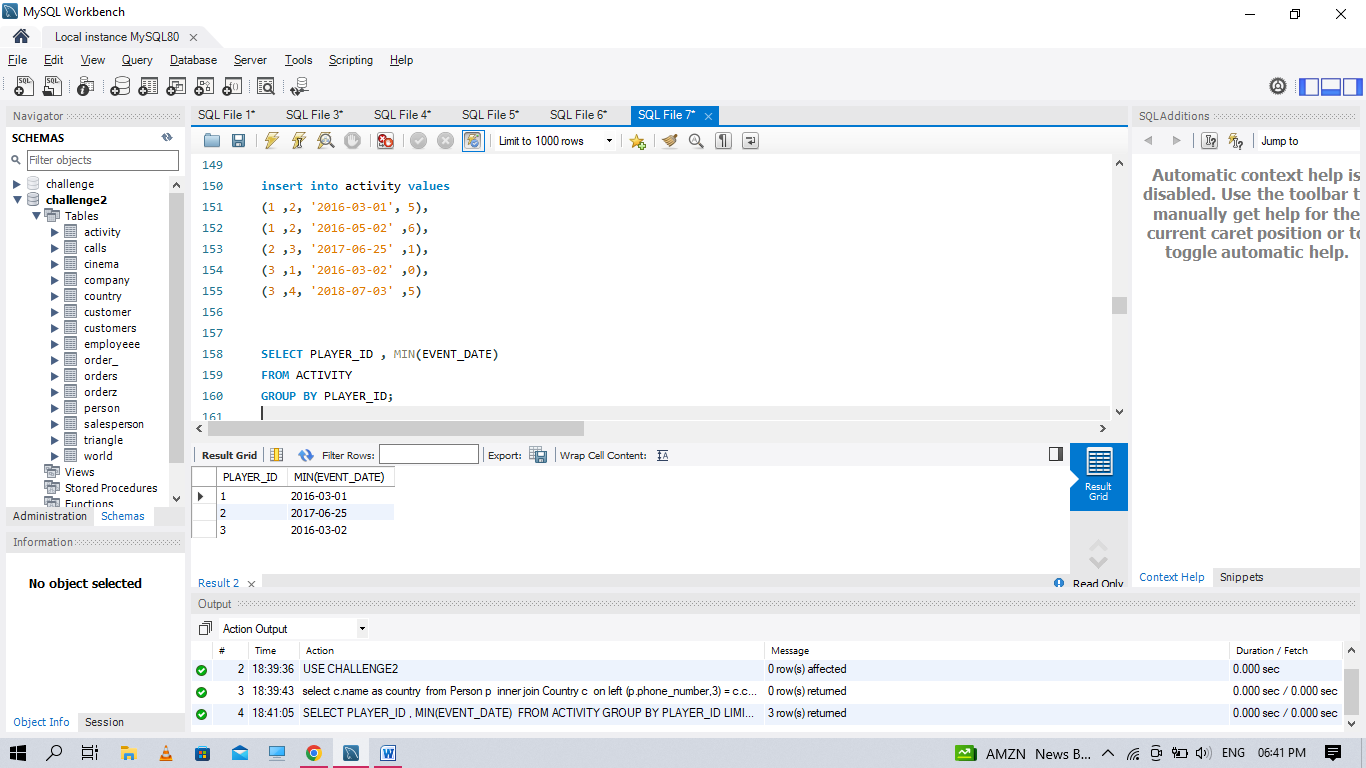
(3 ,1, '2016-03-02' ,0),

(3 ,4, '2018-07-03' ,5)

SELECT PLAYER\_ID , MIN(EVENT\_DATE)

FROM ACTIVITY

GROUP BY PLAYER\_ID;



**Q57 Write an SQL query to find the customer\_number for the customer who has placed the largest number of orders.**

CREATE TABLE ORDER\_

(order\_number int,

customer\_number int,

PRIMARY KEY(order\_number)

)

INSERT INTO ORDER\_ VALUES

(1, 1),

(2 ,2),

(3 ,3),

(4 ,3)

select

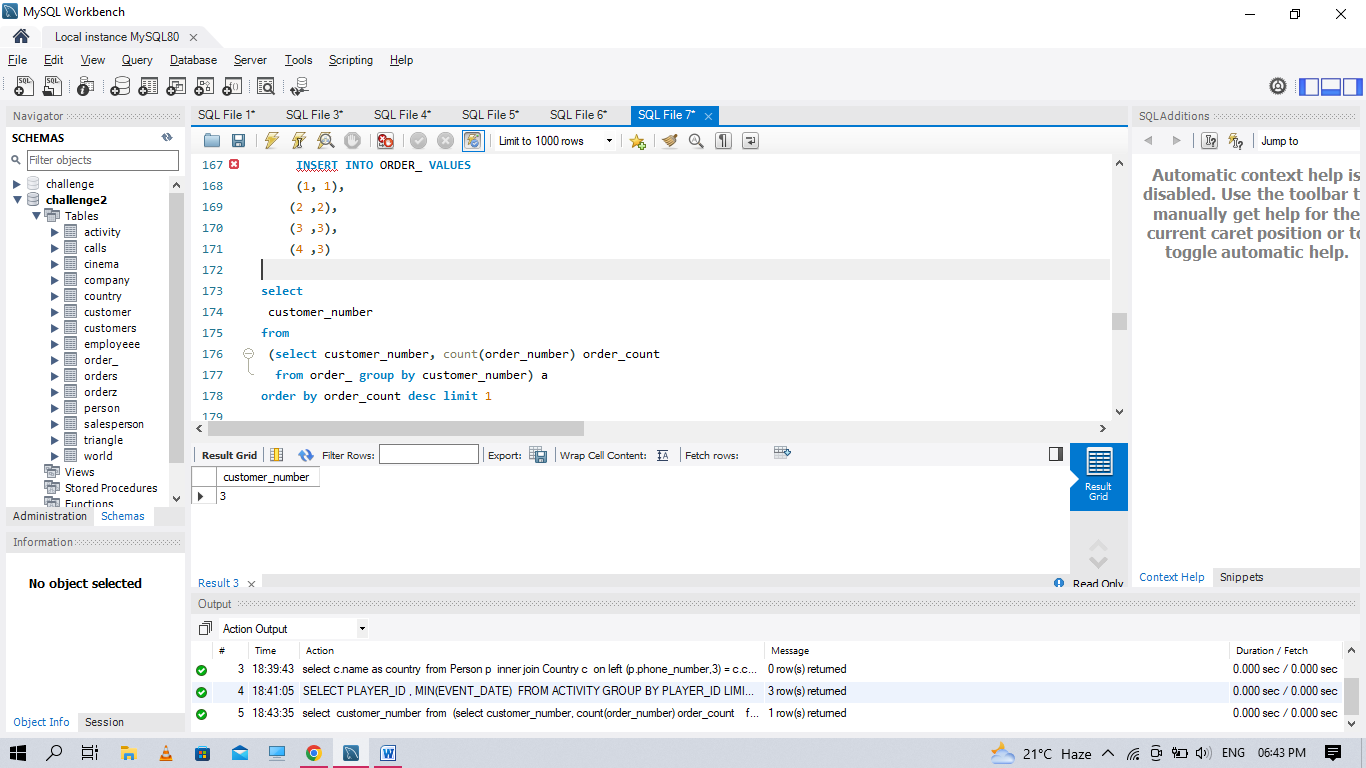
customer\_number

from

(select customer\_number, count(order\_number) order\_count

from order\_ group by customer\_number) a

order by order\_count desc limit 1



**Q58 Write an SQL query to report all the consecutive available seats in the cinema**

CREATE TABLE CINEMA

(seat\_id int NOT NULL AUTO\_INCREMENT ,

free bool,

PRIMARY KEY (seat\_id)

)

INSERT INTO CINEMA VALUES

(1 ,1),

(2 ,0),

(3 ,1),

(4 ,1),

(5 ,1)

SELECT

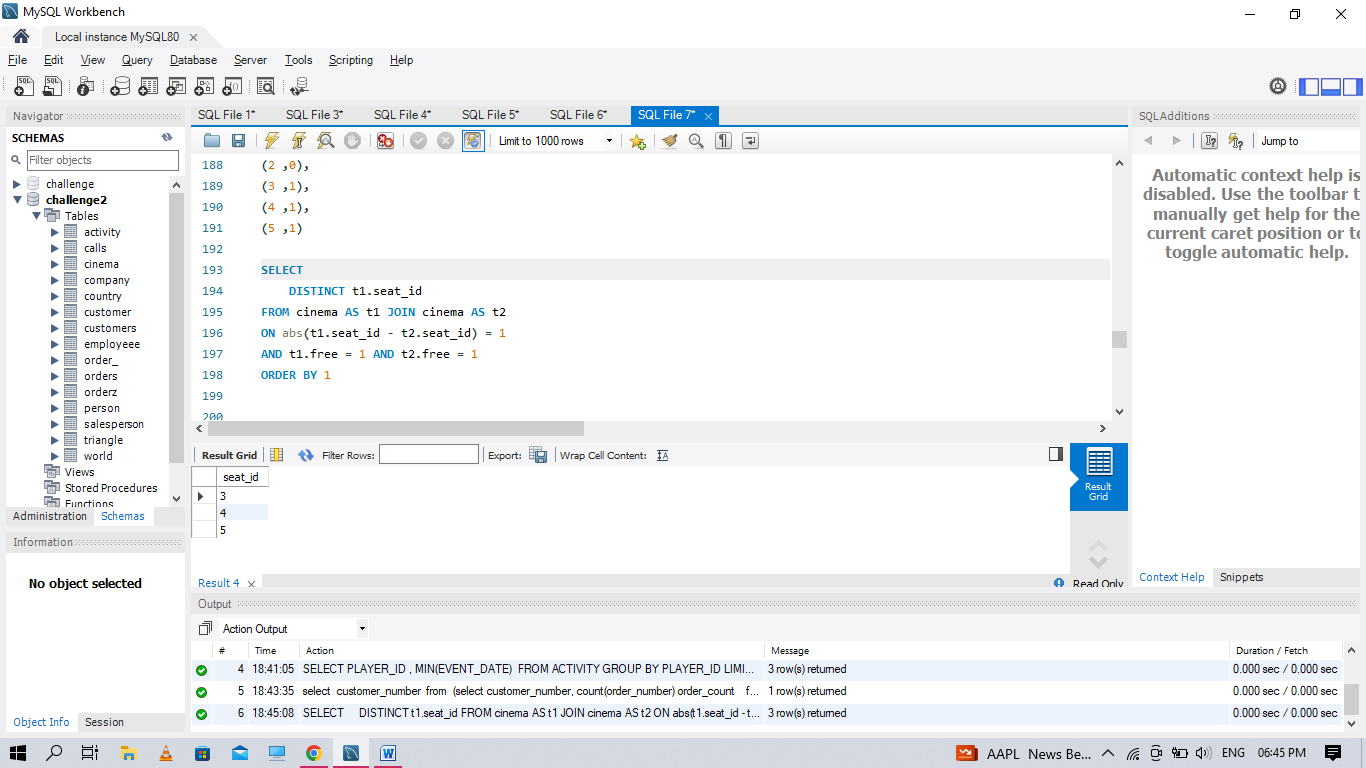
DISTINCT t1.seat\_id

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 1



**Q59 Write an SQL query to report the names of all the salespersons who did not have any orders related to the company with the name "RED".**

CREATE TABLE COMPANY

(com\_id int,

name varchar(30),

city varchar(30),

PRIMARY KEY(com\_id))

CREATE TABLE ORDERZ

(

order\_id int,

order\_date date,

com\_id int,

sales\_id int,

amount int,

PRIMARY KEY (order\_id),

FOREIGN KEY (com\_id) REFERENCES COMPANY(com\_id)

)

CREATE TABLE SALESPERSON

(sales\_id int,

name varchar(30),

salary int,

commission\_rate int,

hire\_date date,

PRIMARY KEY(sales\_id)

)

INSERT INTO SALESPERSON VALUES

(1, 'John' ,100000, 6 ,'2006-01-04'),

(2, 'Amy' ,12000 ,5 ,'2010-01-05'),

(3, 'Mark' ,65000 ,12, '2008-12-25'),

(4, 'Pam' ,25000 ,25 ,'2005-01-01'),

(5, 'Alex' ,5000 ,10 ,'2007-03-02')

INSERT INTO COMPANY VALUES

(1, 'RED', 'Boston'),

(2, 'ORANGE', 'New York'),

(3, 'YELLOW' ,'Boston'),

(4, 'GREEN' ,'Austin')

INSERT INTO ORDERZ VALUES

(1, '2014-01-01', 3, 4, 10000),

(2, '2014-01-02', 4, 5, 5000),

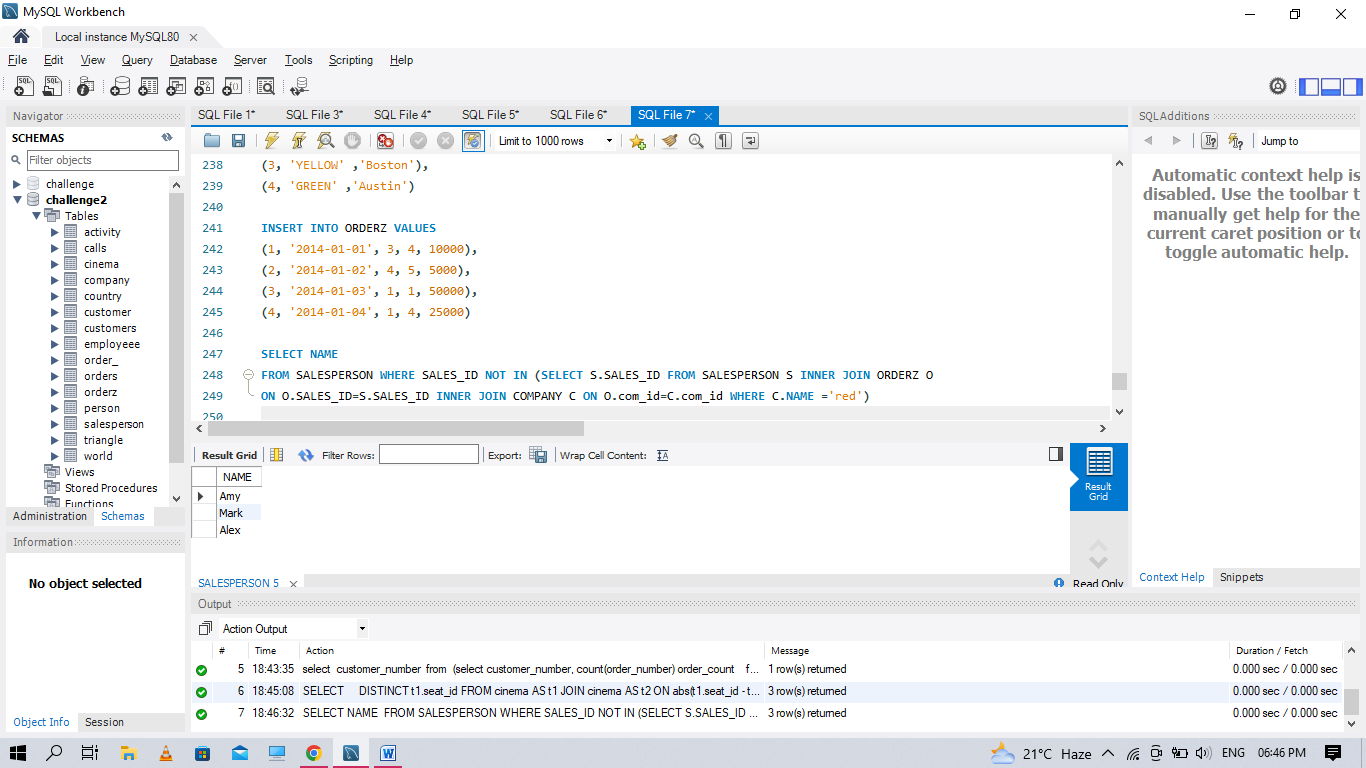
(3, '2014-01-03', 1, 1, 50000),

(4, '2014-01-04', 1, 4, 25000)

SELECT NAME

FROM SALESPERSON WHERE SALES\_ID NOT IN (SELECT S.SALES\_ID FROM SALESPERSON S INNER JOIN ORDERZ O

ON O.SALES\_ID=S.SALES\_ID INNER JOIN COMPANY C ON O.com\_id=C.com\_id WHERE C.NAME ='red')



**Q60. Write an SQL query to report for every three line segments whether they can form a triangle. Return the result table in any order.**

CREATE TABLE TRIANGLE

(x int,

y int,

z int,

PRIMARY KEY(X,Y,Z))

INSERT INTO TRIANGLE VALUES

(13, 15, 30),

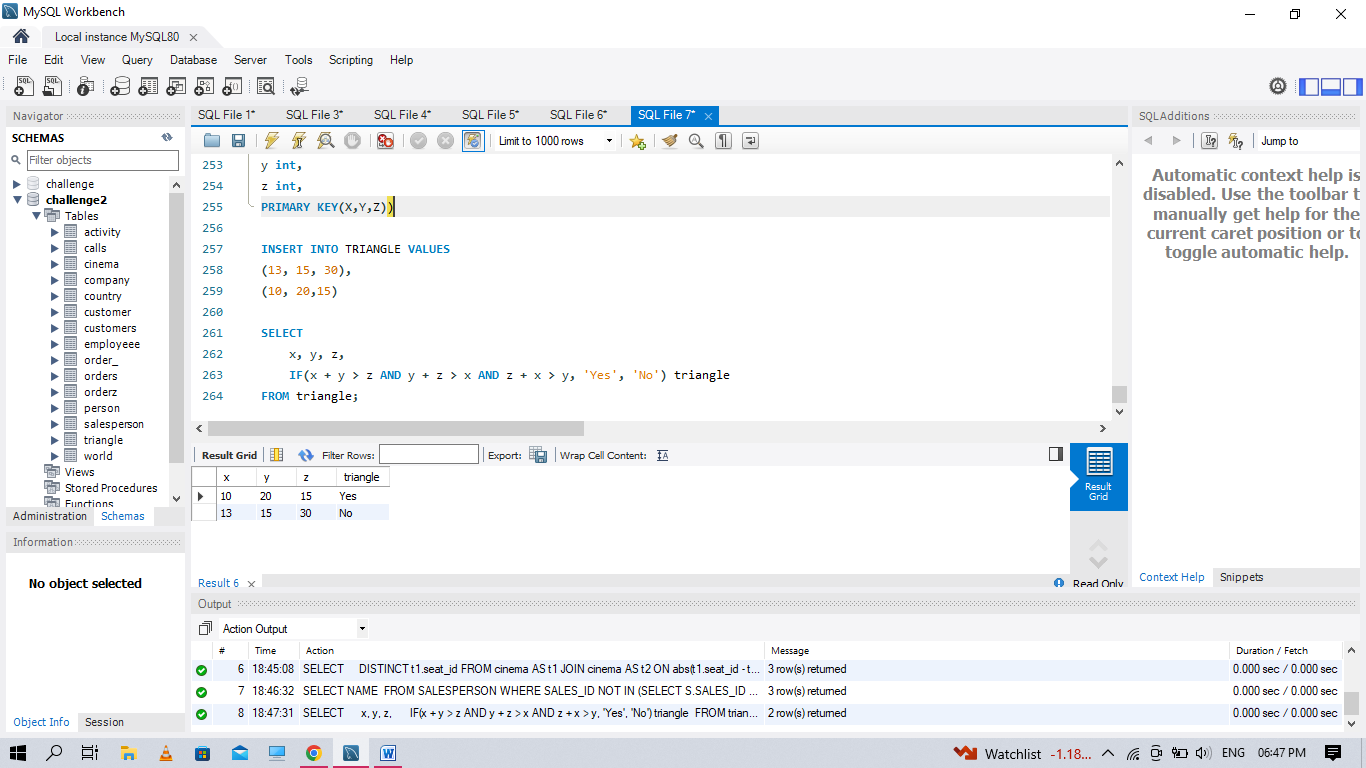
(10, 20,15)

SELECT

x, y, z,

IF(x + y > z AND y + z > x AND z + x > y, 'Yes', 'No') triangle

FROM triangle;



**Q61. Write an SQL query to report the shortest distance between any two points from the Point table. The query result format is in the following example.**

CREATE TABLE POINT(

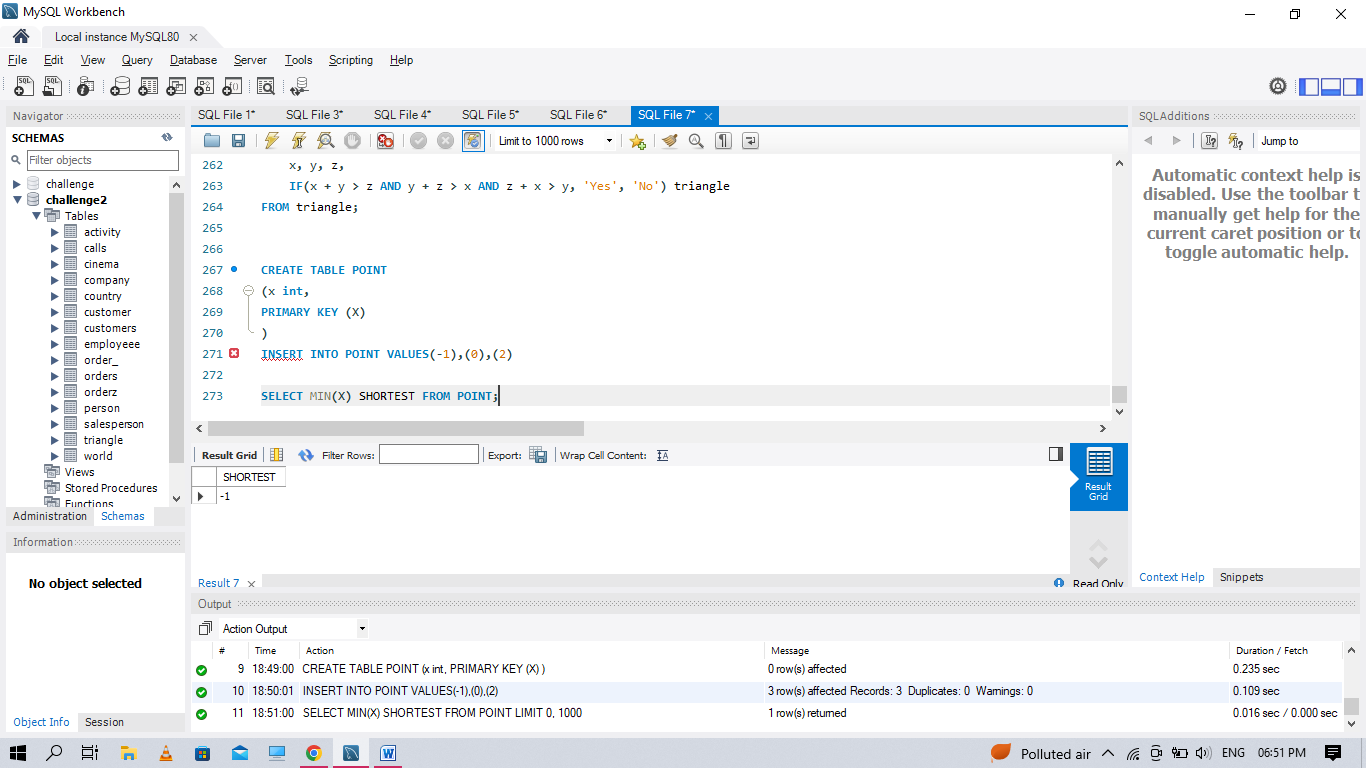
x int,

PRIMARY KEY (X)

)

INSERT INTO POINT VALUES(-1),(0),(2)

SELECT MIN(X) SHORTEST FROM POINT;



**Q62 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.**

CREATE TABLE ACTORDIRETOR

(actor\_id int,

director\_id int,

timestamp int,

PRIMARY KEY(timestamp))

INSERT INTO ACTORDIRETOR VALUES

(1, 1, 0),

(1, 1, 1),

(1, 1, 2),

(1, 2, 3),

(1, 2, 4),

(2, 1, 5),

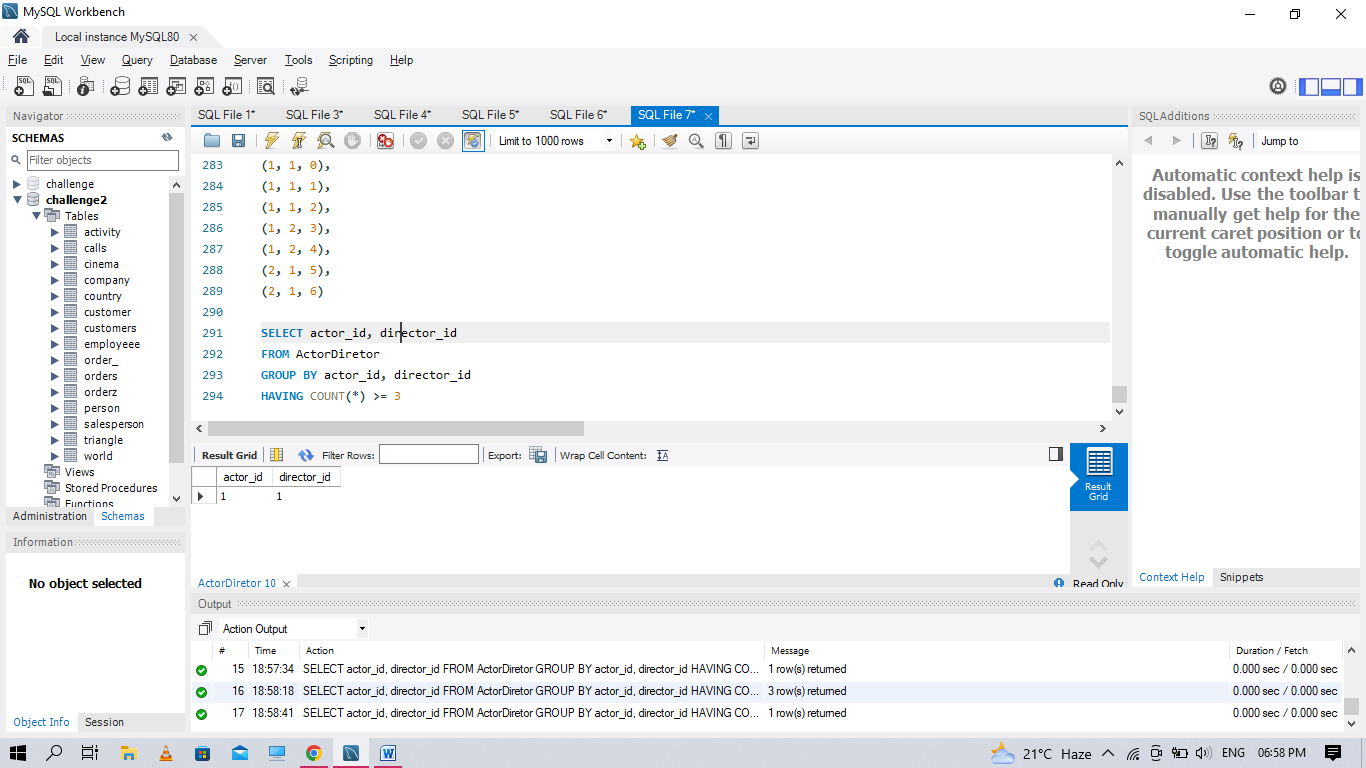
(2, 1, 6)

SELECT actor\_id, director\_id

FROM ActorDiretor

GROUP BY actor\_id, director\_id

HAVING COUNT(\*) >= 3



**Q63. 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**

CREATE TABLE SALES

(sale\_id int,

product\_id int,

year int,

quantity int,

price int,

PRIMARY KEY (sale\_id, year) )

CREATE TABLE PRODUCT(

product\_id int,

product\_name varchar(30),

PRIMARY KEY (PRODUCT\_ID))

INSERT INTO SALES VALUES

(1, 100 ,2008 ,10 ,5000),

(2, 100 ,2009 ,12 ,5000),

(7, 200 ,2011 ,15 ,9000)

INSERT INTO PRODUCT VALUES

(100, 'Nokia'),

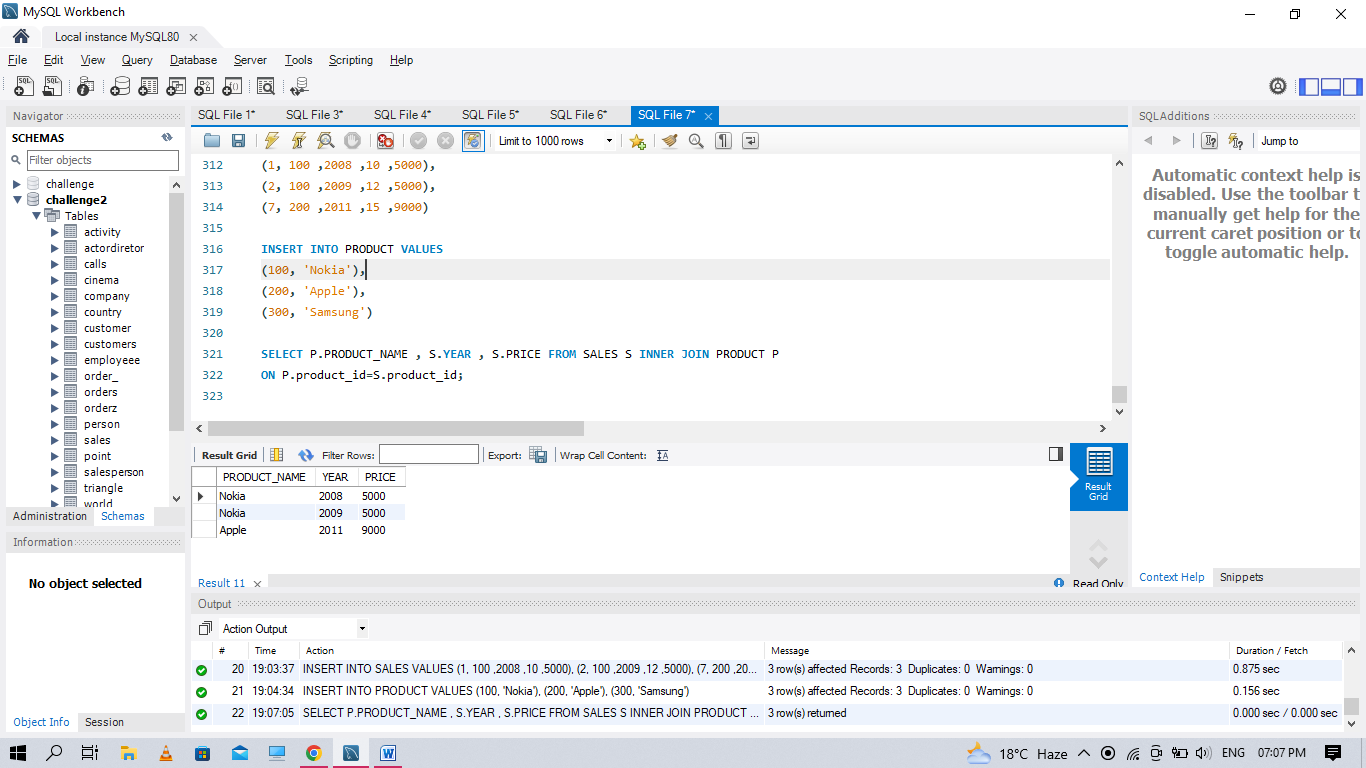
(200, 'Apple'),

(300, 'Samsung')

SELECT P.PRODUCT\_NAME , S.YEAR , S.PRICE FROM SALES S

INNER JOIN PRODUCT P

ON P.product\_id=S.product\_id;



**Q64. Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits**

CREATE TABLE PROJECT

(project\_id int,

employee\_id int,

PRIMARY KEY (project\_id,employee\_id))

CREATE TABLE EMPLOYEE

(employee\_id int,

name varchar(30),

experience\_years int,

PRIMARY KEY(employee\_id)

)

INSERT INTO PROJECT VALUES

(1, 1),

(1, 2),

(1 ,3),

(2, 1),

(2, 4)

INSERT INTO EMPLOYEE VALUES

(1, 'Khaled' ,3),

(2, 'Ali',2),

(3, 'John', 1),

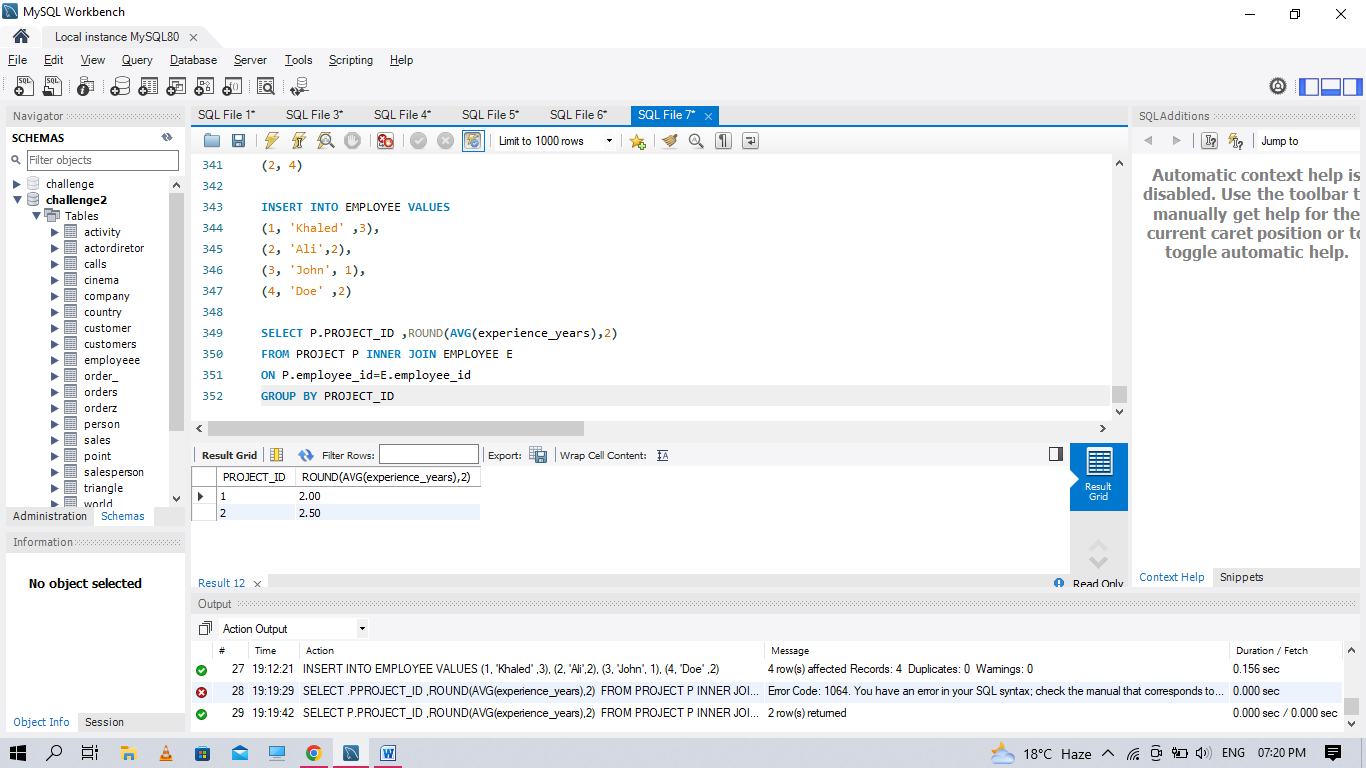
(4, 'Doe' ,2)

SELECT P.PROJECT\_ID ,ROUND(AVG(experience\_years),2)

FROM PROJECT P INNER JOIN EMPLOYEE E

ON P.employee\_id=E.employee\_id

GROUP BY PROJECT\_ID



**Q65. Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all. Return the result table in any order.**

CREATE TABLE PRODUCTS

(product\_id int,

product\_name varchar(30),

unit\_price int,

PRIMARY KEY(product\_id) )

CREATE TABLE SALE

(seller\_id int,

product\_id int,

buyer\_id int,

sale\_date date,

quantity int,

price int,

FOREIGN KEY (product\_id) REFERENCES PRODUCTS(product\_id))

INSERT INTO PRODUCTS VALUES

(1, 'S8' ,1000),

(2, 'G4' ,800),

(3, 'iPhone', 1400)

INSERT INTO SALE 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)

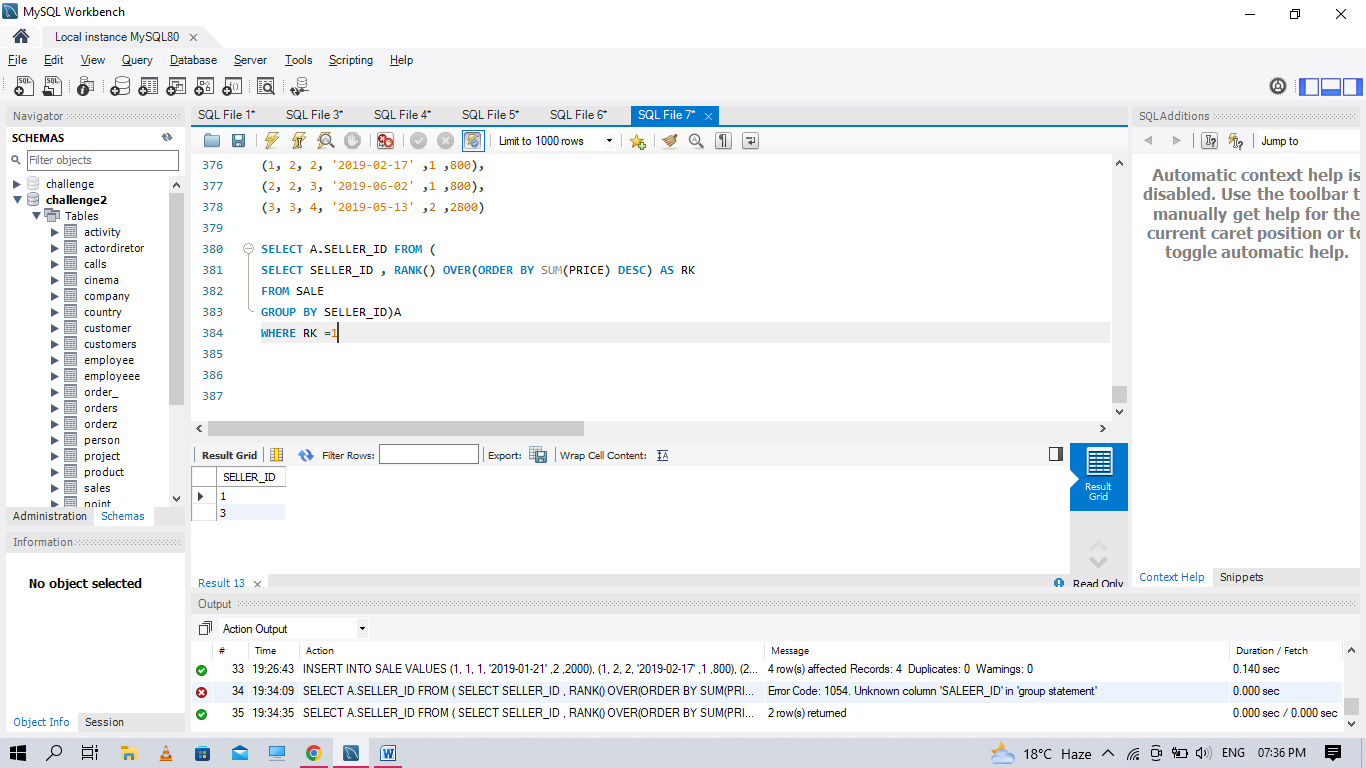
SELECT A.SELLER\_ID FROM (

SELECT SELLER\_ID , RANK() OVER(ORDER BY SUM(PRICE) DESC) AS RK

FROM SALE

GROUP BY SELLER\_ID)A

WHERE RK =1

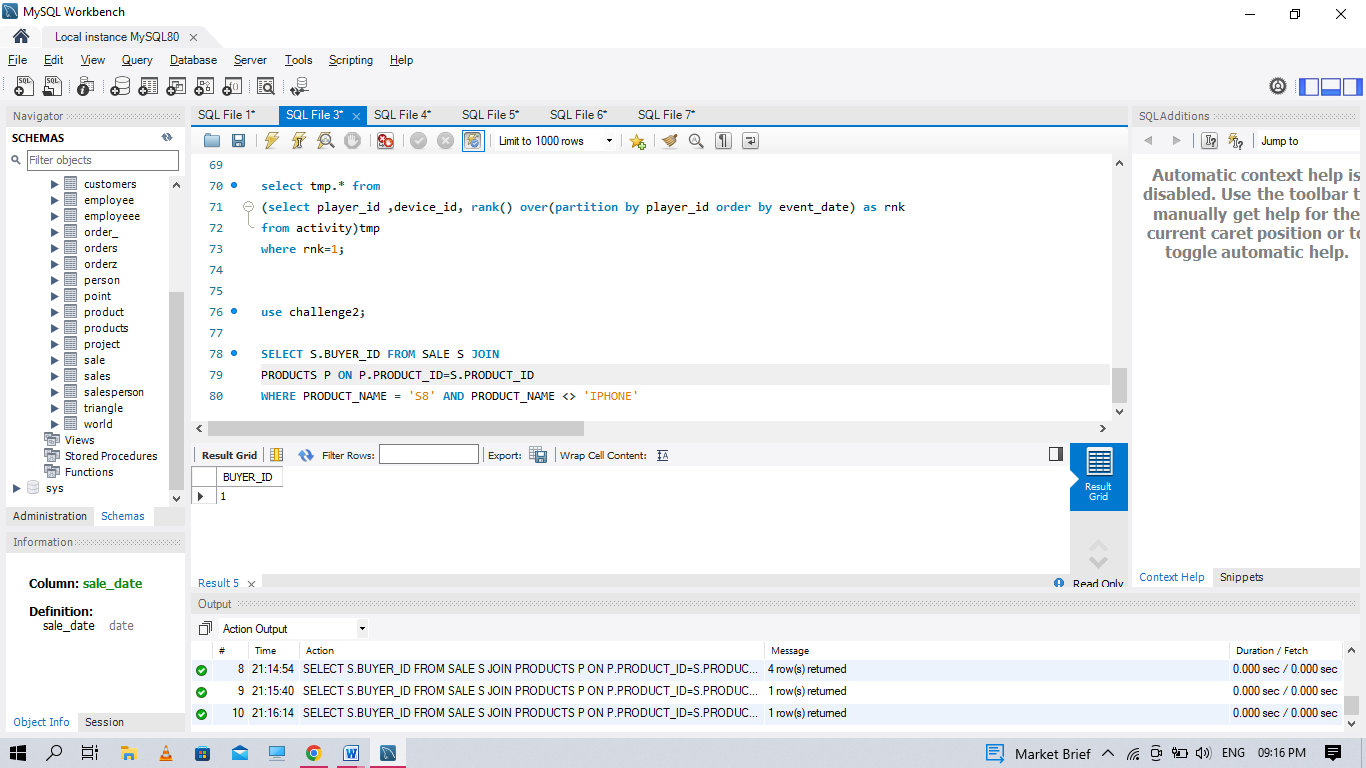


**Q66 Write an SQL query that reports the buyers who have bought S8 but not iPhone. Note that S8 and iPhone are products present in the Product table.**

SELECT S.BUYER\_ID FROM SALE S JOIN

PRODUCTS P ON P.PRODUCT\_ID=S.PRODUCT\_ID

WHERE PRODUCT\_NAME = 'S8' AND PRODUCT\_NAME <> 'IPHONE'



**Q68. Write an SQL query to find the total score for each gender on each day**

CREATE TABLE SCORES

(player\_name varchar(30),

gender varchar(30),

day date,

score\_points int,

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),

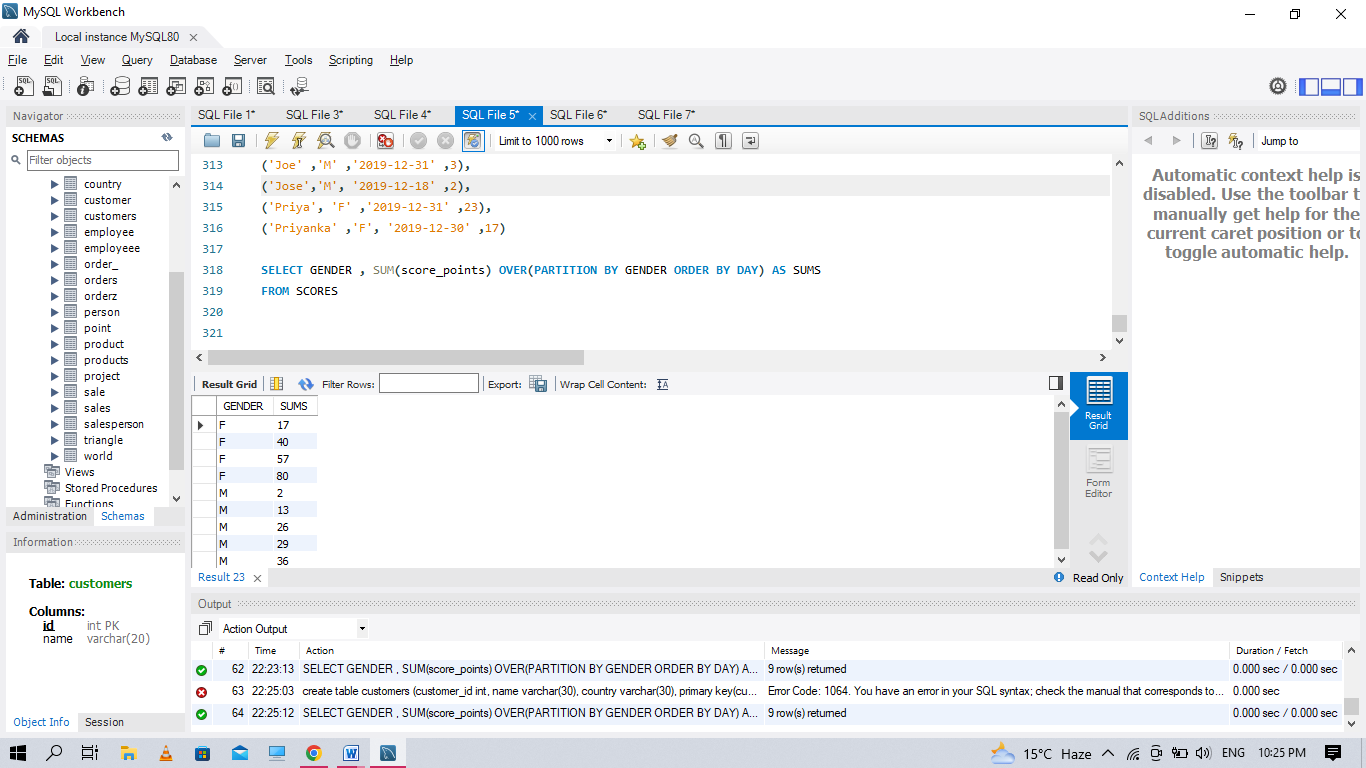
('Jose','M', '2019-12-18' ,2),

('Priya', 'F' ,'2019-12-31' ,23),

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

SELECT GENDER , SUM(score\_points) OVER(PARTITION BY GENDER ORDER BY DAY) AS SUMS

FROM SCORES



**Q69. Write an SQL query to find the start and end number of continuous ranges in the table Logs. Return the result table ordered by start\_id.**

CREATE TABLE LOGS(

log\_id int,

PRIMARY KEY (log\_id)

)

INSERT INTO LOGS VALUE

(1),

(2),

(3),

(7),

(8),

(10)

SELECT

MIN(log\_id) AS start\_id,

MAX(log\_id) AS end\_id

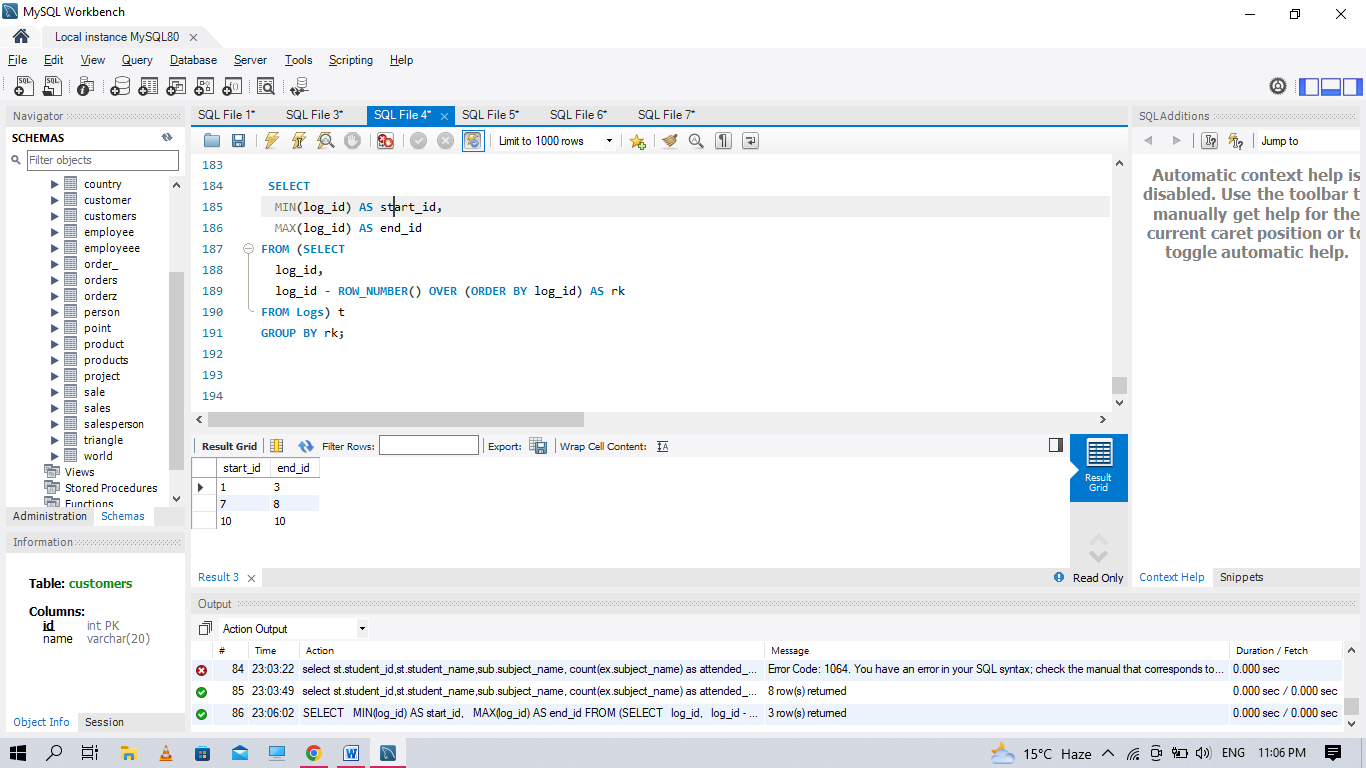
FROM (SELECT

log\_id,

log\_id - ROW\_NUMBER() OVER (ORDER BY log\_id) AS rk

FROM Logs) t

GROUP BY rk;



**Q70 .Write an SQL query to find the number of times each student attended each exam**

CREATE TABLE STUDENTS

(student\_id int,

student\_name varchar(30),

primary key(student\_id))

create table subjects(

subject\_name varchar(30),

primary key (subject\_name)

)

create table examinations

(student\_id int,

subject\_name varchar(30))

insert into STUDENTS values

(1, 'Alice'),

(2, 'Bob'),

(13,'John'),

(6, 'Alex')

insert into subjects values

( 'Math'),

( 'Physics'),

('Programming')

insert into examinations values

(1, 'Math'),

(1, 'Physics'),

(1, 'Programming'),

(2, 'Programming'),

(1, 'Physics'),

(1, 'Math'),

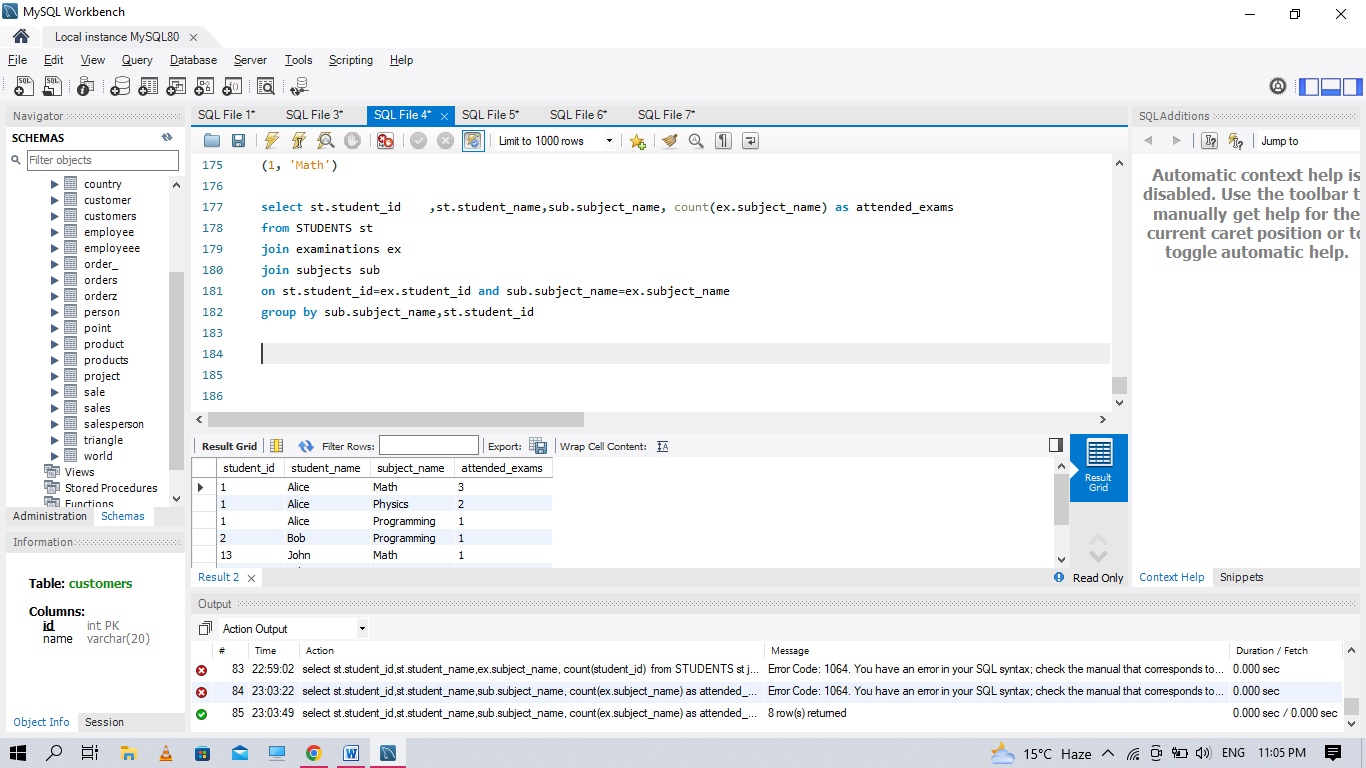
(13, 'Math'),

(13, 'Programming'),

(13, 'Physics'),

(2, 'Math'),

(1, 'Math')



**Q71. Write an SQL query to find employee\_id of all employees that directly or indirectly report their work to the head of the company**

create table employees

(employee\_id int,

employee\_name varchar(30),

manager\_id int,

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)

select a.employee\_id as EMPLOYEE\_ID

from

Employees as a

join

Employees as b on a.manager\_id = b.employee\_id

join

Employees as c on b.manager\_id = c.employee\_id

join

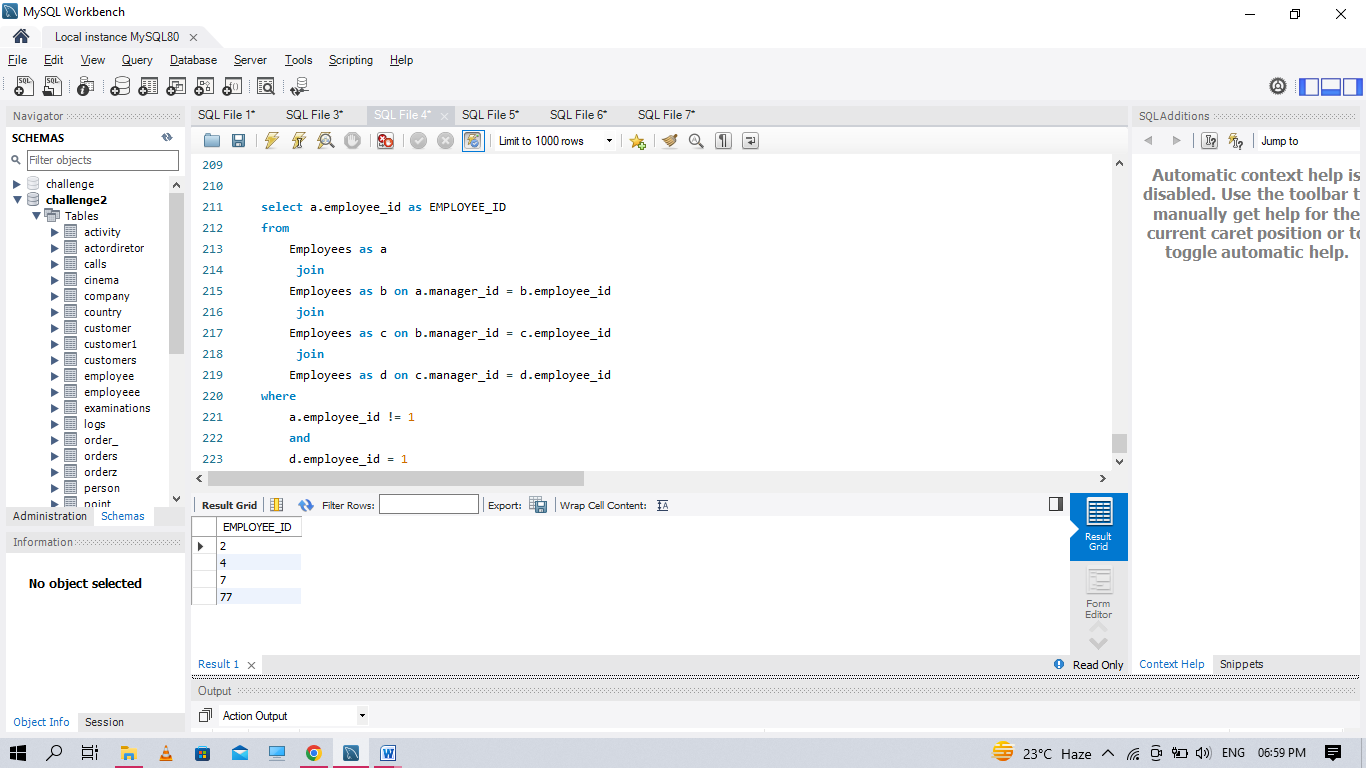
Employees as d on c.manager\_id = d.employee\_id

where

a.employee\_id != 1

and

d.employee\_id = 1



**Q72. Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.**

create table transactions

(id int,

country varchar(20),

state enum("approved", "declined"),

amount int,

trans\_date date,

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 date\_format(trans\_date,"%Y-%m") as month, country,

count(id) as trans\_count,

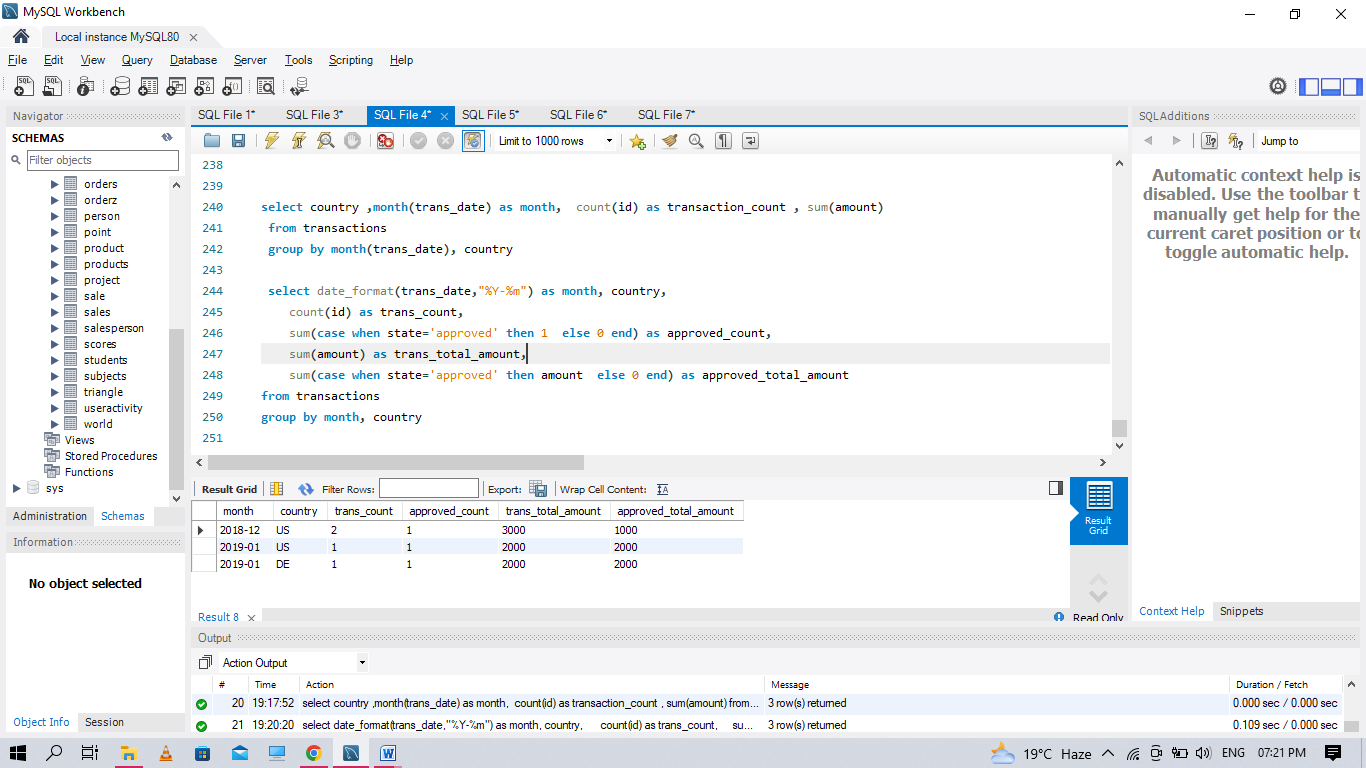
sum(case when state='approved' then 1 else 0 end) as approved\_count,

sum(amount) as trans\_total\_amount,

sum(case when state='approved' then amount else 0 end) as approved\_total\_amount

from transactions

group by month, country



**Q73. Write an SQL query to find the average daily percentage of posts that got removed after being reported as spam, rounded to 2 decimal places.**

create table actions

(user\_id int,

post\_id int,

action\_date date,

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

extra varchar(20)

)

create table removal

(post\_id int,

remove\_date date,

primary key(post\_id)

)

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')

insert into removal values

(2, '2019-07-20'),

(3 ,'2019-07-18')

select round(avg(daily\_count), 2) as average\_daily\_percent

from (select count(distinct b.post\_id)/count(distinct a.post\_id)\*100 as daily\_count

from actions a

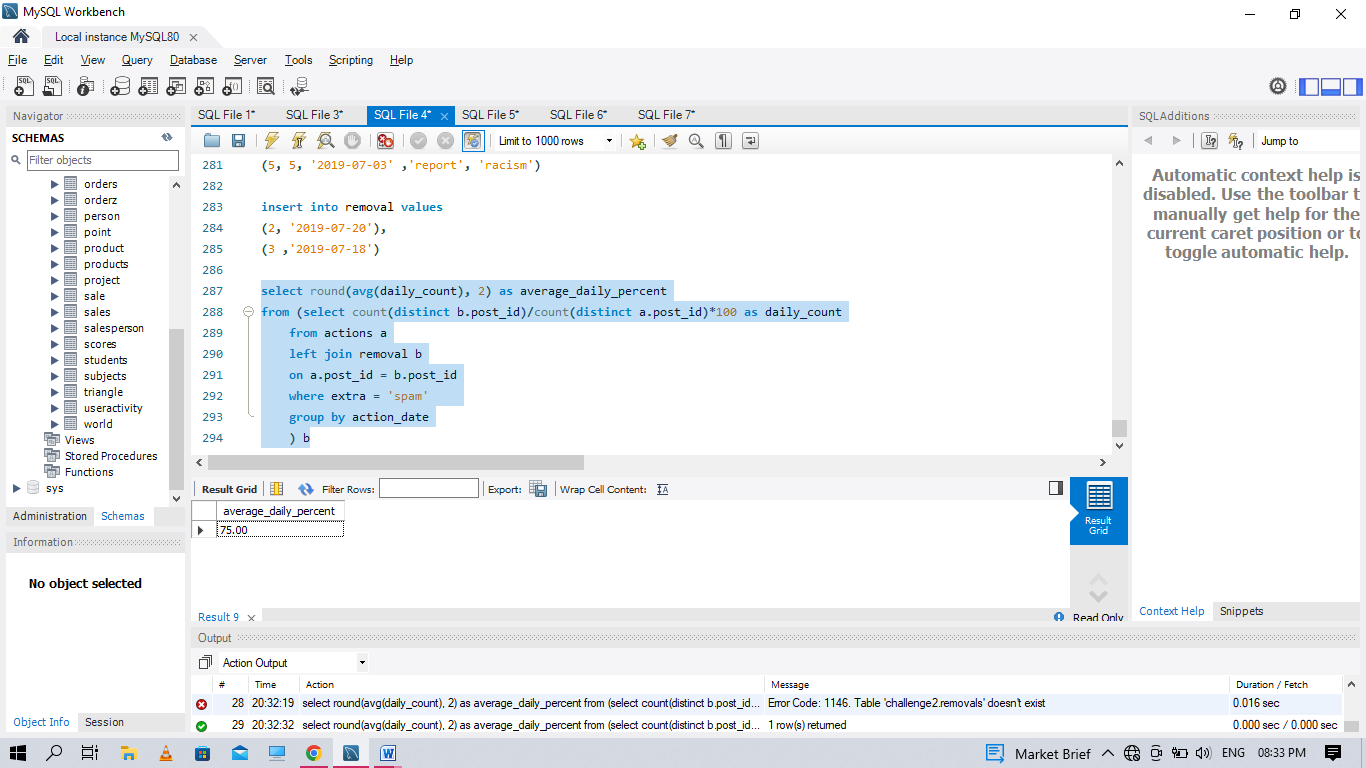
left join removal b

on a.post\_id = b.post\_id

where extra = 'spam'

group by action\_date

) b



**Q74. Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players**

create table activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

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)

WITH cte AS (

SELECT player\_id, MIN(event\_date) as first\_login

FROM Activity1

GROUP BY player\_id

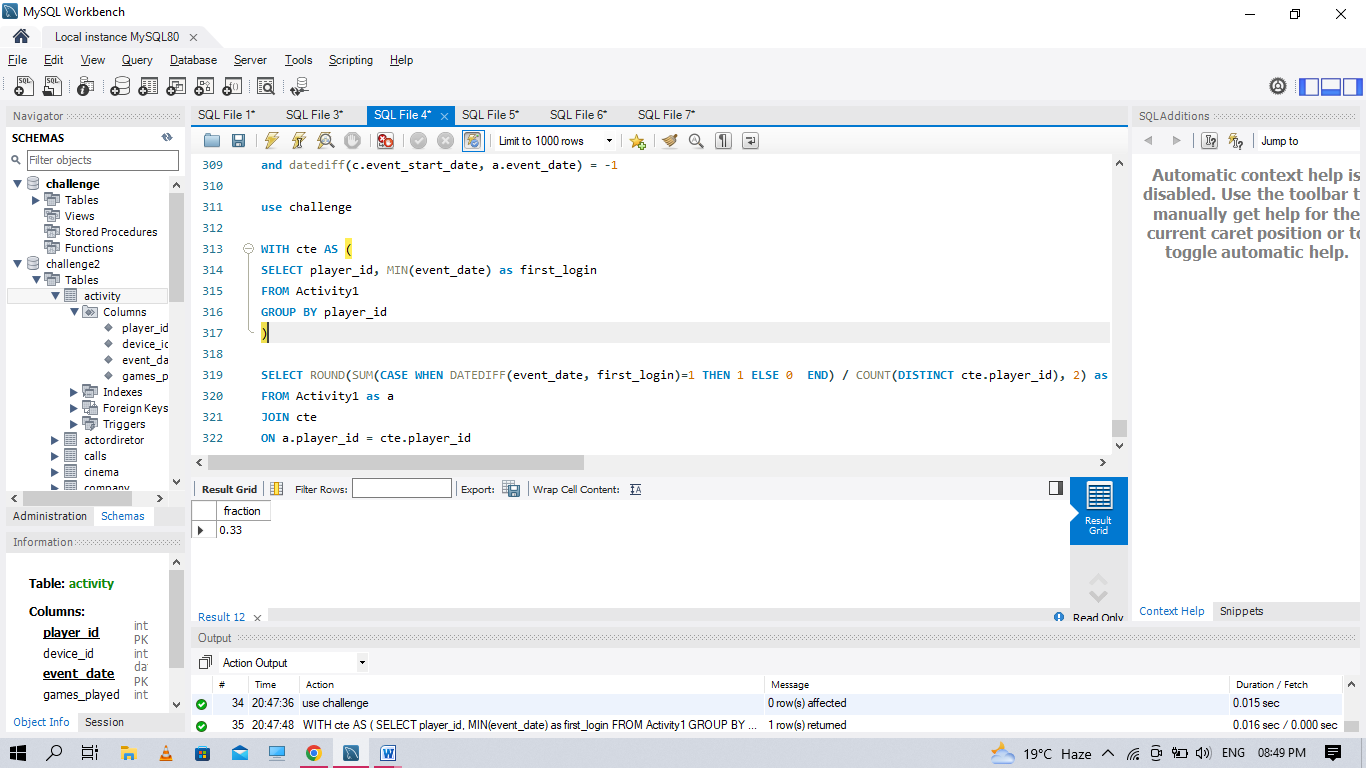
)

SELECT ROUND(SUM(CASE WHEN DATEDIFF(event\_date, first\_login)=1 THEN 1 ELSE 0 END) / COUNT(DISTINCT cte.player\_id), 2) as fraction

FROM Activity1 as a

JOIN cte

ON a.player\_id = cte.player\_id



**Q75. Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players**

create table activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

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)

WITH cte AS (

SELECT player\_id, MIN(event\_date) as first\_login

FROM Activity1

GROUP BY player\_id

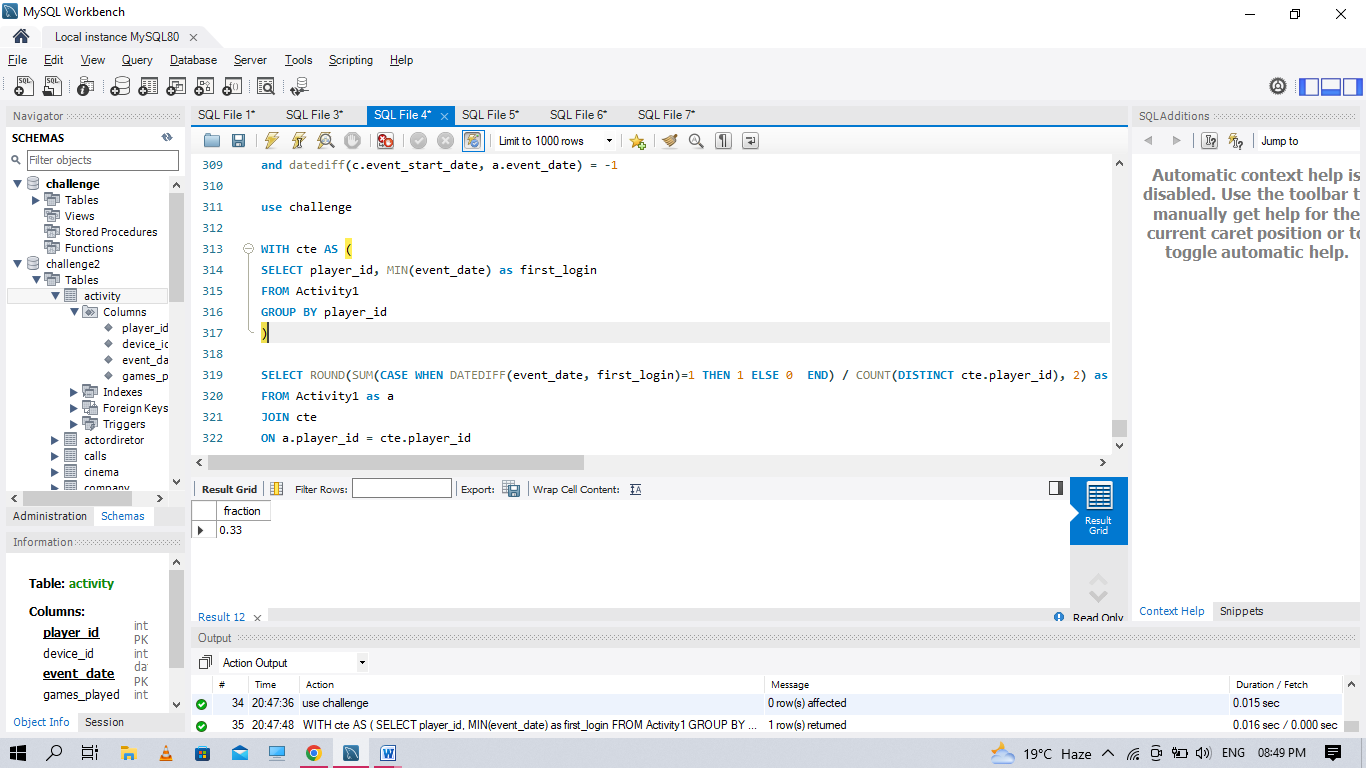
)

SELECT ROUND(SUM(CASE WHEN DATEDIFF(event\_date, first\_login)=1 THEN 1 ELSE 0 END) / COUNT(DISTINCT cte.player\_id), 2) as fraction

FROM Activity1 as a

JOIN cte

ON a.player\_id = cte.player\_id



**Q76 Write an SQL query to find the salaries of the employees after applying taxes. Round the salary to the nearest integer**

create table salaries

(company\_id int,

employee\_id int,

employee\_name varchar(30),

salary int,

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 company\_id, employee\_id, employee\_name, round(salary - salary\*tax, 0) as salary

from

(

select \*,

case when max(salary) over(partition by company\_id) < 1000 then 0

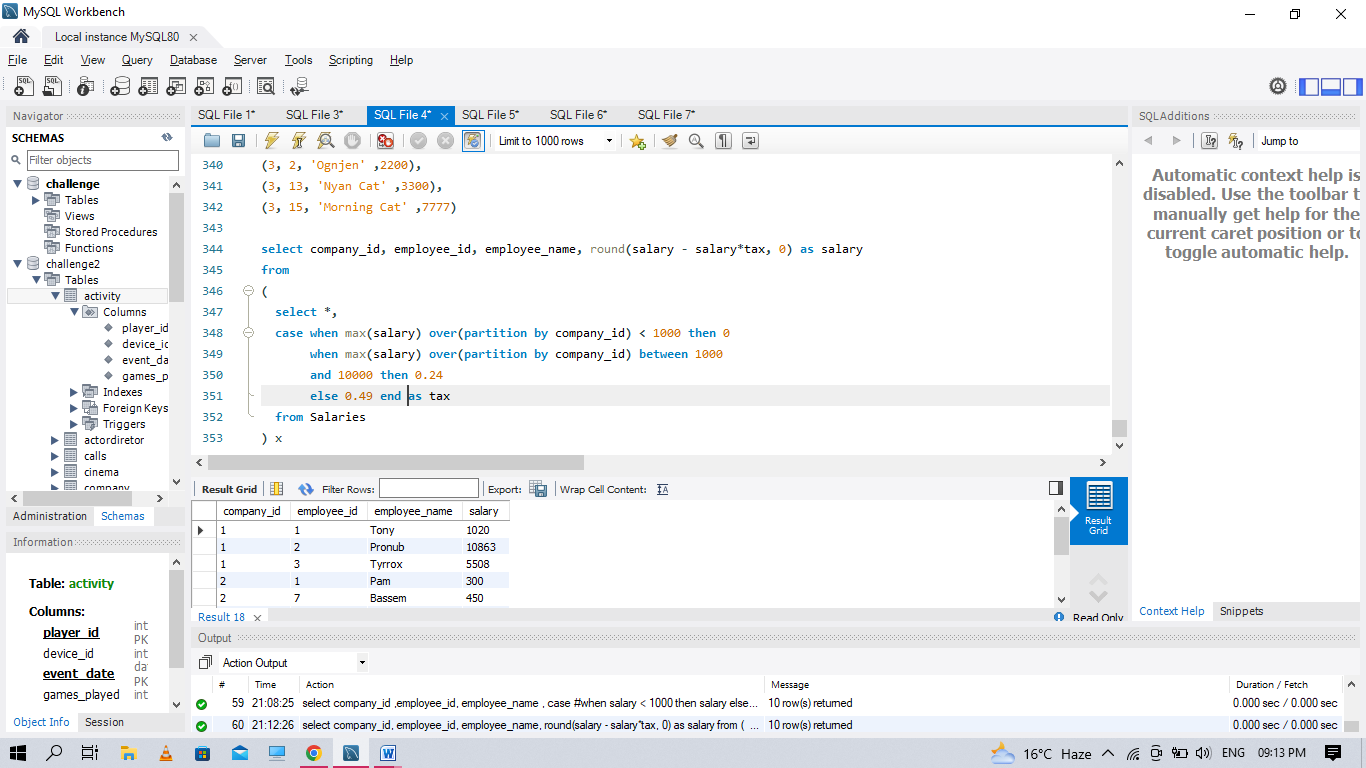
when max(salary) over(partition by company\_id) between 1000

and 10000 then 0.24

else 0.49 end as tax

from Salaries

) x



**Q77 Write an SQL query to evaluate the boolean expressions in Expressions table**

CREATE TABLE EXPRESSIONS

(left\_operand varchar(5),

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

right\_operand varchar(5),

PRIMARY KEY (left\_operand, operator, right\_operand))

CREATE TABLE VARIABLE

(name varchar(10),

value int,

primary key(NAME)

)

INSERT INTO VARIABLE VALUES

('x' ,66),

('y' ,77)

INSERT INTO EXPRESSIONS VALUES

('x', '>', 'y' ),

('x', '<','y' ),

('x', '=', 'y' ),

('y', '>', 'x'),

('y', '<', 'x'),

('x', '=', 'x')

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

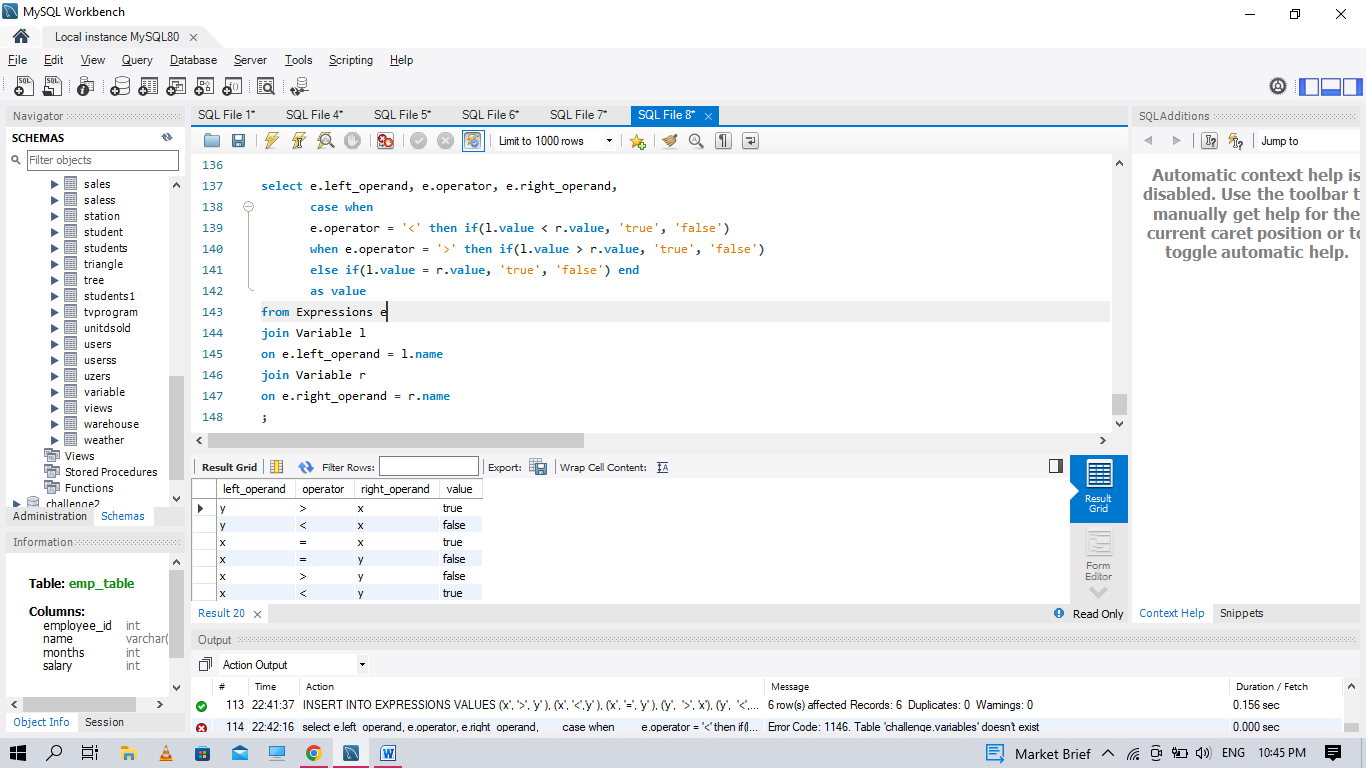
join Variable l

on e.left\_operand = l.name

join Variable r

on e.right\_operand = r.name

;



**Q78. Write an SQL query to find the countries where this company can invest. Return the result table in any order.**

create table person

(

id int,

name varchar(20),

phone\_number varchar(20),

primary key(id)

)

create table country

(

name varchar(20),

country\_code varchar(20),

primary key(country\_code)

)

create table calls

(caller\_id int,

callee\_id int,

duration int

)

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')

insert into country values

('Peru' ,'51'),

('Israel' ,'972'),

('Morocco' ,'212'),

('Germany' ,'49'),

('Ethiopia' ,'251')

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)

select c.name as country

from Person p

inner join Country c

on left (p.phone\_number,3) = c.country\_code

inner join (select caller\_id as id, duration

from Calls

union all

select callee\_id as id, duration

from Calls) phn

on p.id = phn.id

group by country

having avg(duration) > (select avg(duration) from Calls)

**Q79. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.**

SELECT NAME FROM EMP\_TABLE ORDER BY NAME ASC;

create table emp\_table

(employee\_id int,

name varchar(30),

months int,

salary int)

insert into emp\_table values

(12228,'rose',15,1968),

(33645,'angelina',1,3443),

(45692,'frank',17,1608),

(56118,'patrik',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)

