**101. Write an SQL query to show the second most recent activity of each user**

CREATE TABLE USERACTIVITY

(username varchar(30),

activity varchar(30),

startDate Date,

endDate Date

)

INSERT INTO USERACTIVITY VALUES

('Alice' ,'Travel', '2020-02-12', '2020-02-20'),

('Alice', 'Dancing', '2020-02-21' ,'2020-02-23'),

('Alice', 'Travel', '2020-02-24', '2020-02-28'),

('Bob', 'Travel' ,'2020-02-11', '2020-02-18')

select distinct username, activity, startDate, endDate

from

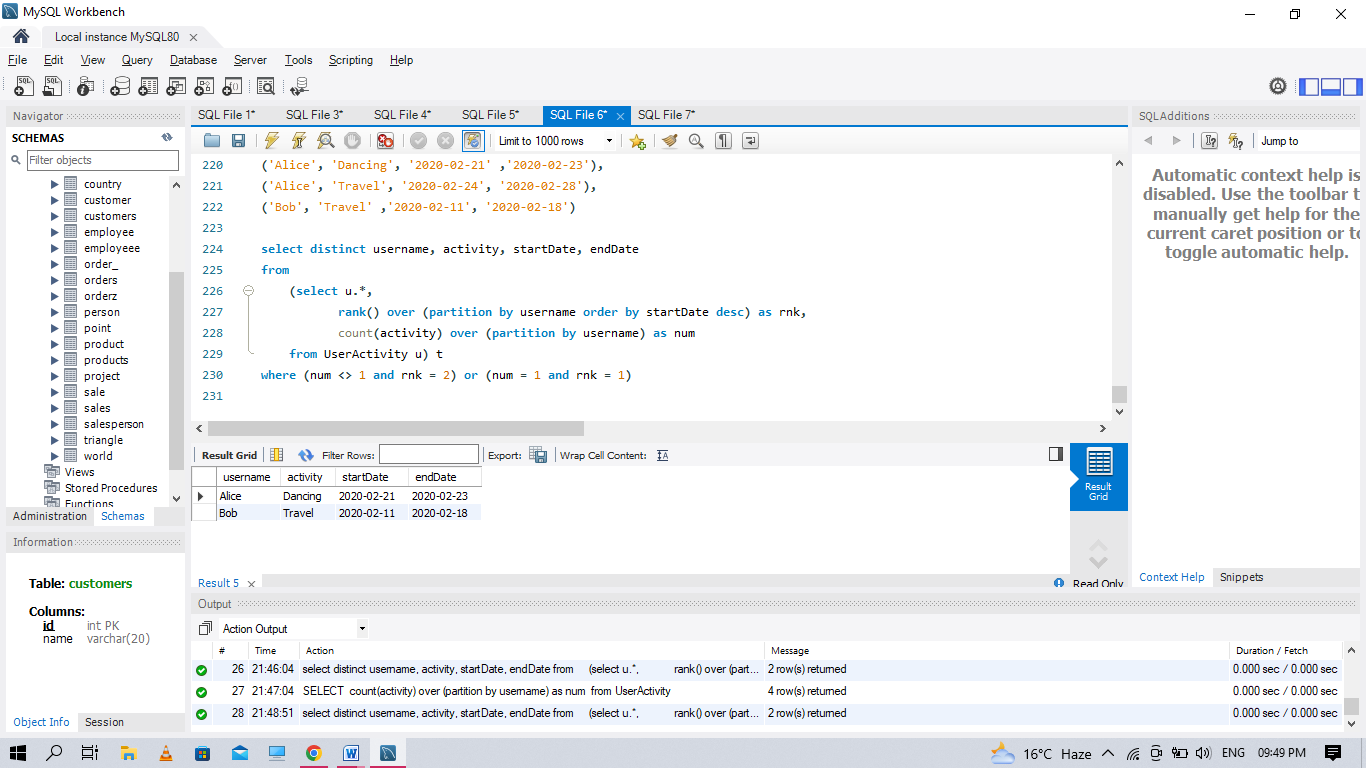
(select u.\*,

rank() over (partition by username order by startDate desc) as rnk,

count(activity) over (partition by username) as num

from UserActivity u) t

where (num <> 1 and rnk = 2) or (num = 1 and rnk = 1)



**102. Write an SQL query to show the second most recent activity of each user**

CREATE TABLE USERACTIVITY

(username varchar(30),

activity varchar(30),

startDate Date,

endDate Date

)

INSERT INTO USERACTIVITY VALUES

('Alice' ,'Travel', '2020-02-12', '2020-02-20'),

('Alice', 'Dancing', '2020-02-21' ,'2020-02-23'),

('Alice', 'Travel', '2020-02-24', '2020-02-28'),

('Bob', 'Travel' ,'2020-02-11', '2020-02-18')

select distinct username, activity, startDate, endDate

from

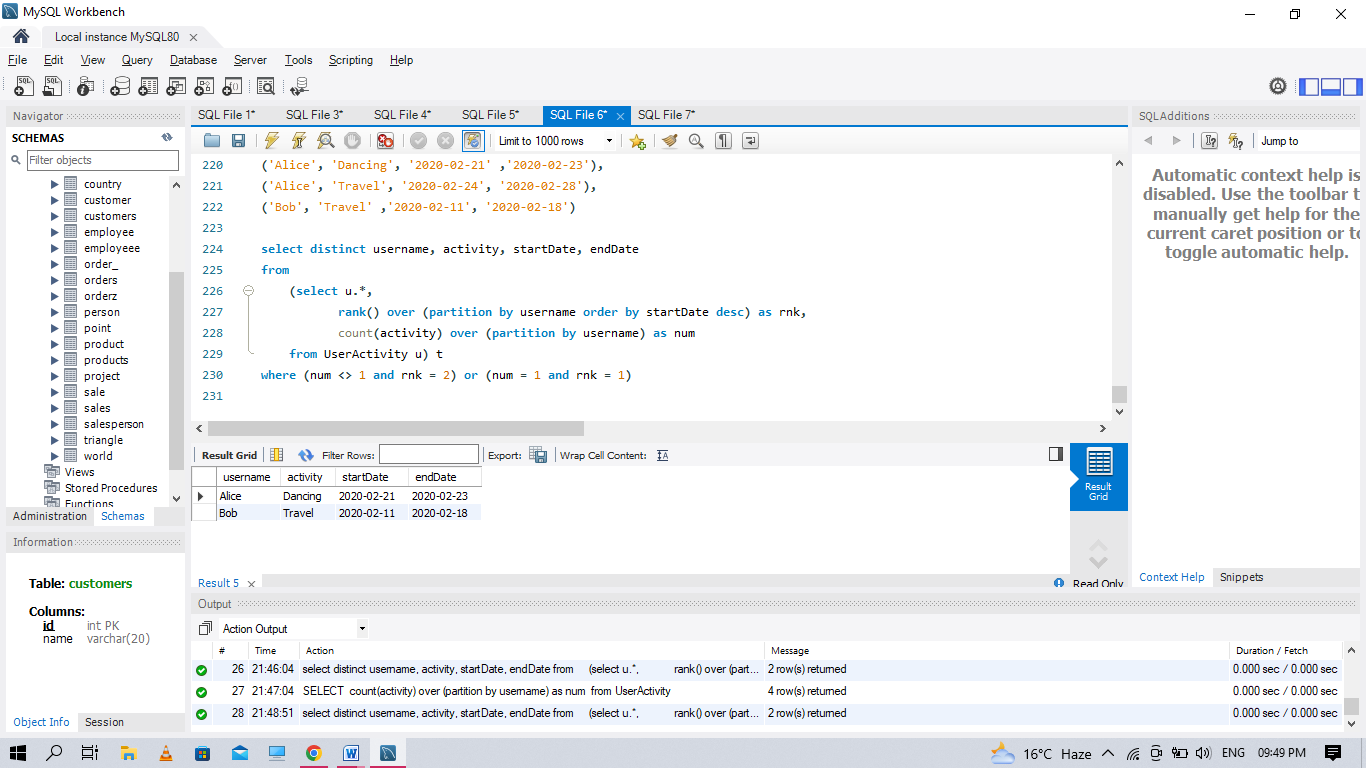
(select u.\*,

rank() over (partition by username order by startDate desc) as rnk,

count(activity) over (partition by username) as num

from UserActivity u) t

where (num <> 1 and rnk = 2) or (num = 1 and rnk = 1)



**103. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same**

create table students1

(id int,

name varchar(50),

marks int

)

insert into students1 values

(1, 'ashley' ,81),

(2, 'samantha' ,75),

(3, 'julia' ,76),

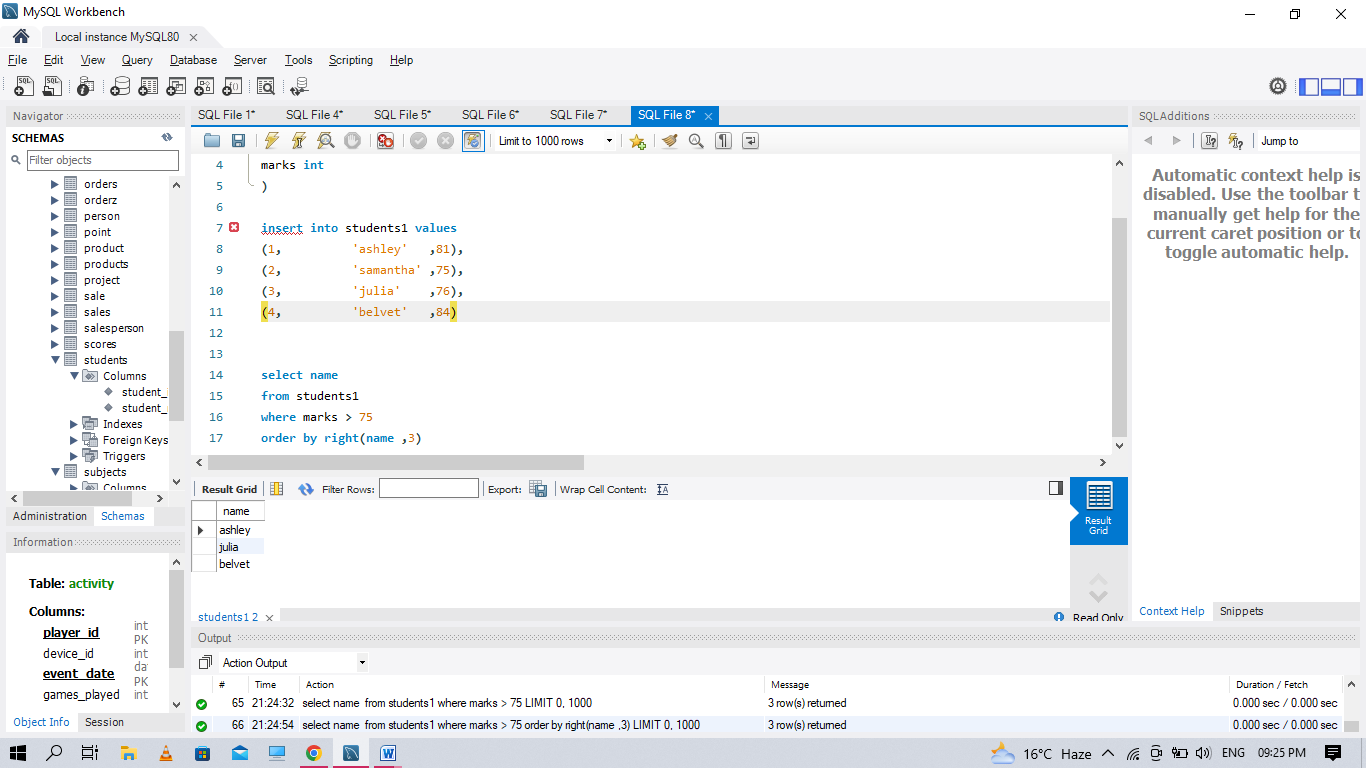
(4, 'belvet' ,84)

select name

from students1

where marks > 75

order by right(name ,3)



**Q104. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id**.

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),

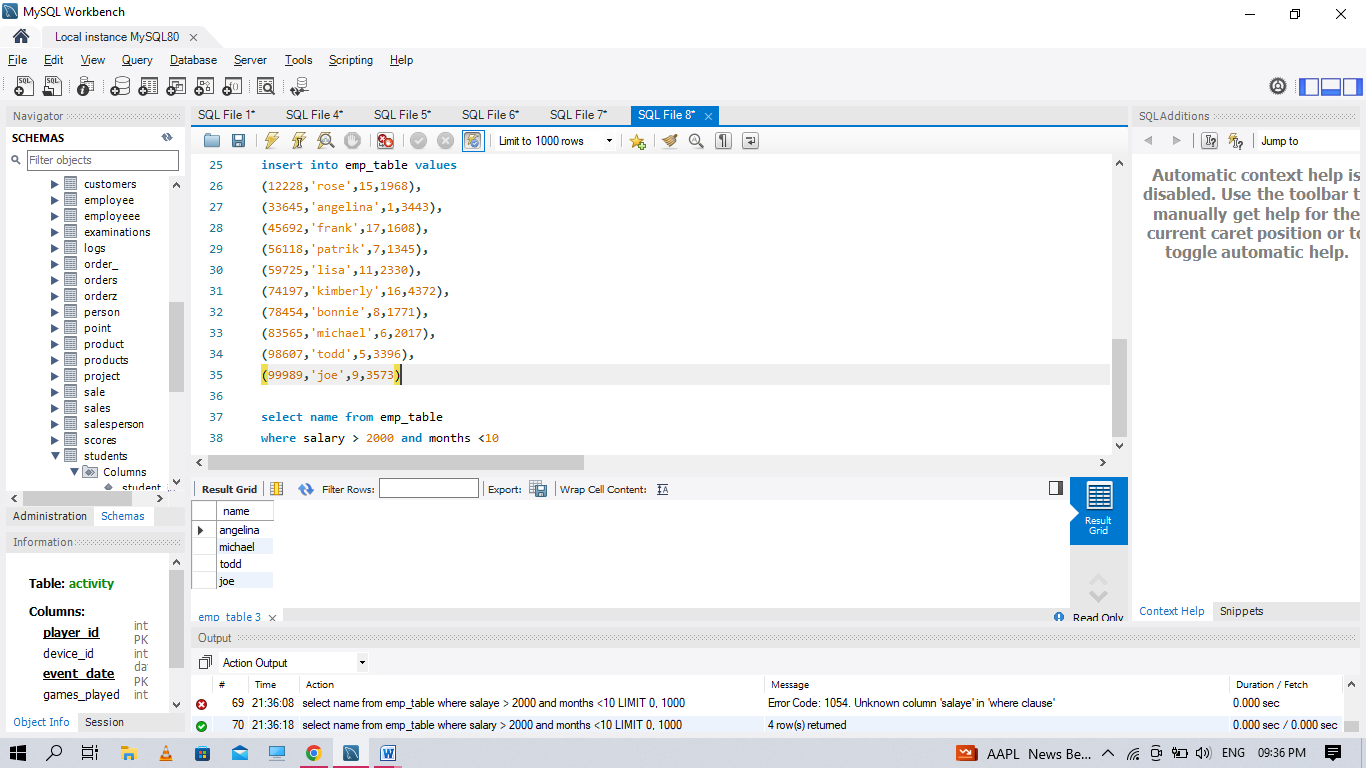
(83565,'michael',6,2017),

(98607,'todd',5,3396),

(99989,'joe',9,3573)

select name from emp\_table

where salary > 2000 and months <10



**Q105 Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table: ● Equilateral: It's a triangle with sides of equal length.**

**● Isosceles: It's a triangle with sides of equal length.**

**● Scalene: It's a triangle with sides of differing lengths.**

**● Not A Triangle: The given values of A, B, and C don't form a triangle**

create table triangle

(a int,

b int,

c int)

insert into triangle values

(20,20,23),(20,20,20),(20,21,22),(13,14,30)

select

case

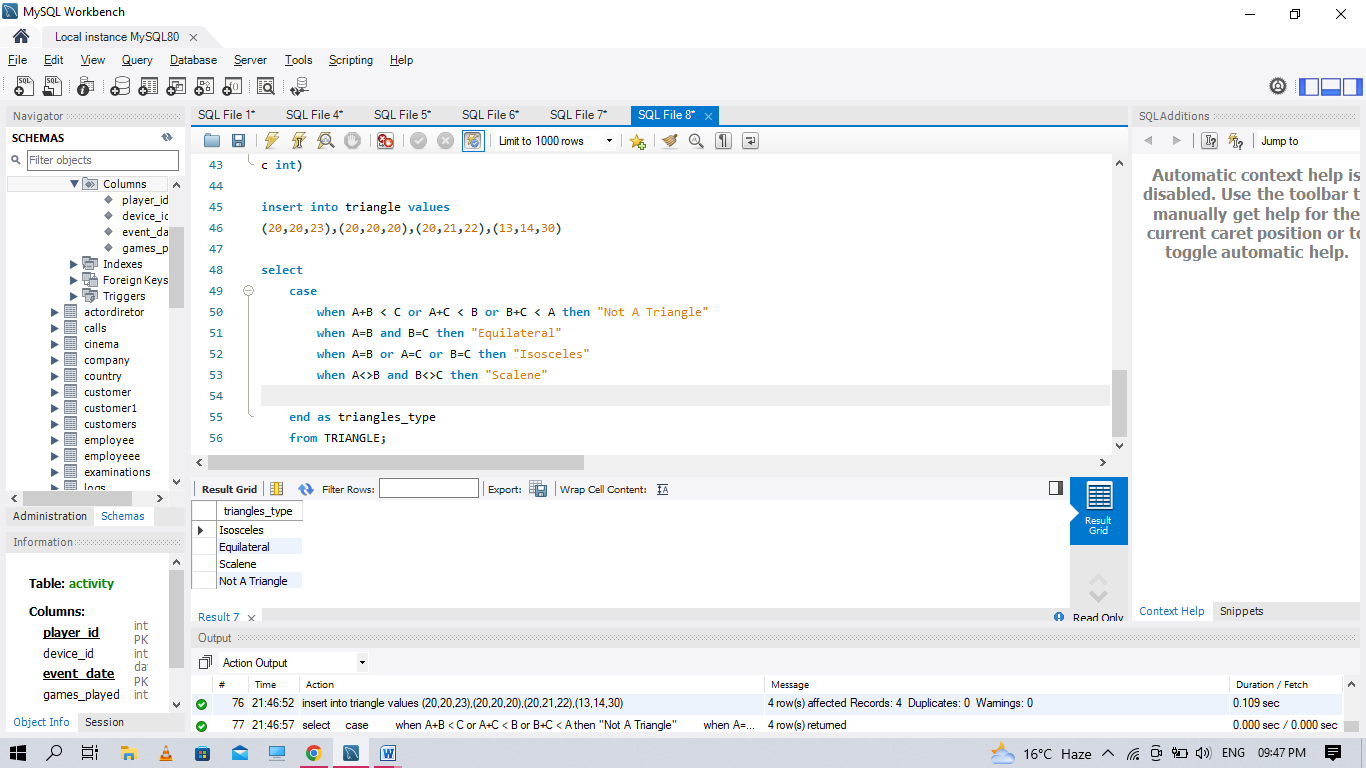
when A+B < C or A+C < B or B+C < A then "Not A Triangle"

when A=B and B=C then "Equilateral"

when A=B or A=C or B=C then "Isosceles"

when A<>B and B<>C then "Scalene"

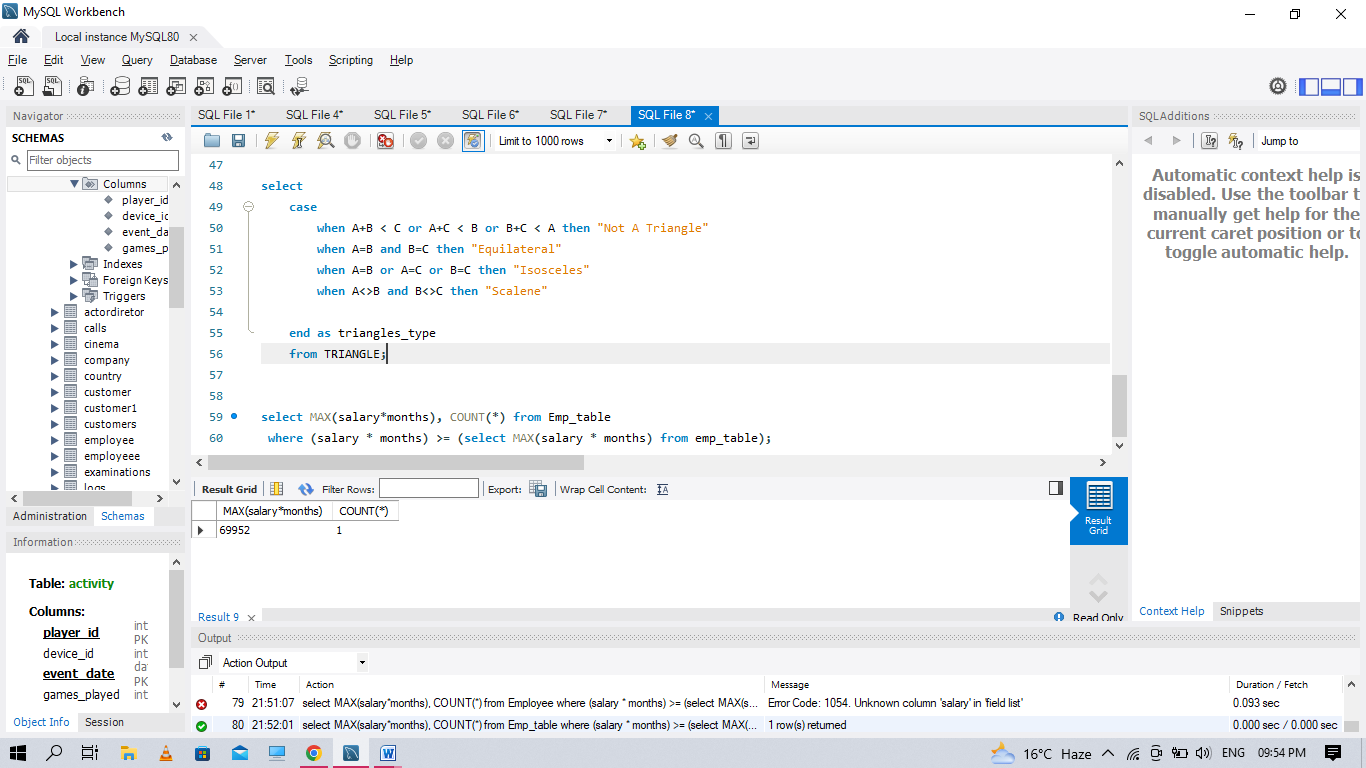
end as triangles\_type



**Q107 Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings.**

select MAX(salary\*months), COUNT(\*) from Emp\_table

where (salary \* months) >= (select MAX(salary \* months) from emp\_table);



**Q108. Generate the following two result sets:**

1. **Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S). Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format: Level - Medium There are a total of [occupation\_count] [occupation]s**
2. **. 2. where [occupation\_count] is the number of occurrences of an occupation in OCCUPATIONS and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation\_count], they should be ordered alphabetically.**

create table occupation

(

name varchar(20),

occupation varchar(30)

)

insert into occupation values

('samantha','doctor'),

('julia','actor'),

('maria','actor'),

('meera','singer'),

('ashely','professor'),

('ketty','professor'),

('christeen','professor'),

('jane','actor'),

('jenny','doctor'),

('priya','singer')

(

SELECT CONCAT(NAME, '(', SUBSTRING(OCCUPATION, 1, 1), ')') as THETEXT, '1' as SELECTNUMBER

FROM OCCUPATION

)

UNION ALL

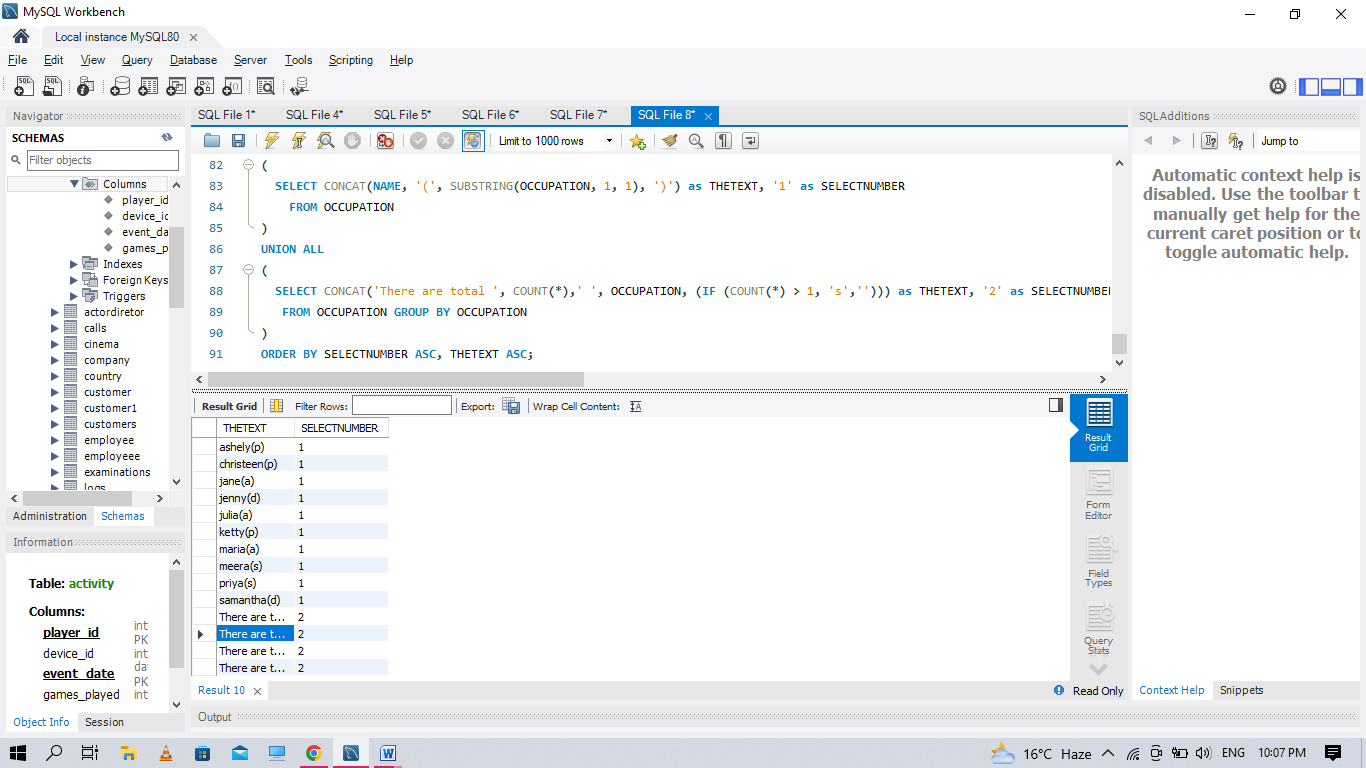
(

SELECT CONCAT('There are total ', COUNT(\*),' ', OCCUPATION, (IF (COUNT(\*) > 1, 's',''))) as THETEXT, '2' as SELECTNUMBER

FROM OCCUPATION GROUP BY OCCUPATION

)

ORDER BY SELECTNUMBER ASC, THETEXT ASC;



**Q110. You are given a table, BST, containing two columns: N and P, where N represents the value of a node in Binary Tree, and P is the parent of N.**

create table tree

(n int,

p int

)

insert into tree values

(1,2),(3,2),(6,8),(9,8),(2,5),(8,5),(5,null)

select n,

case

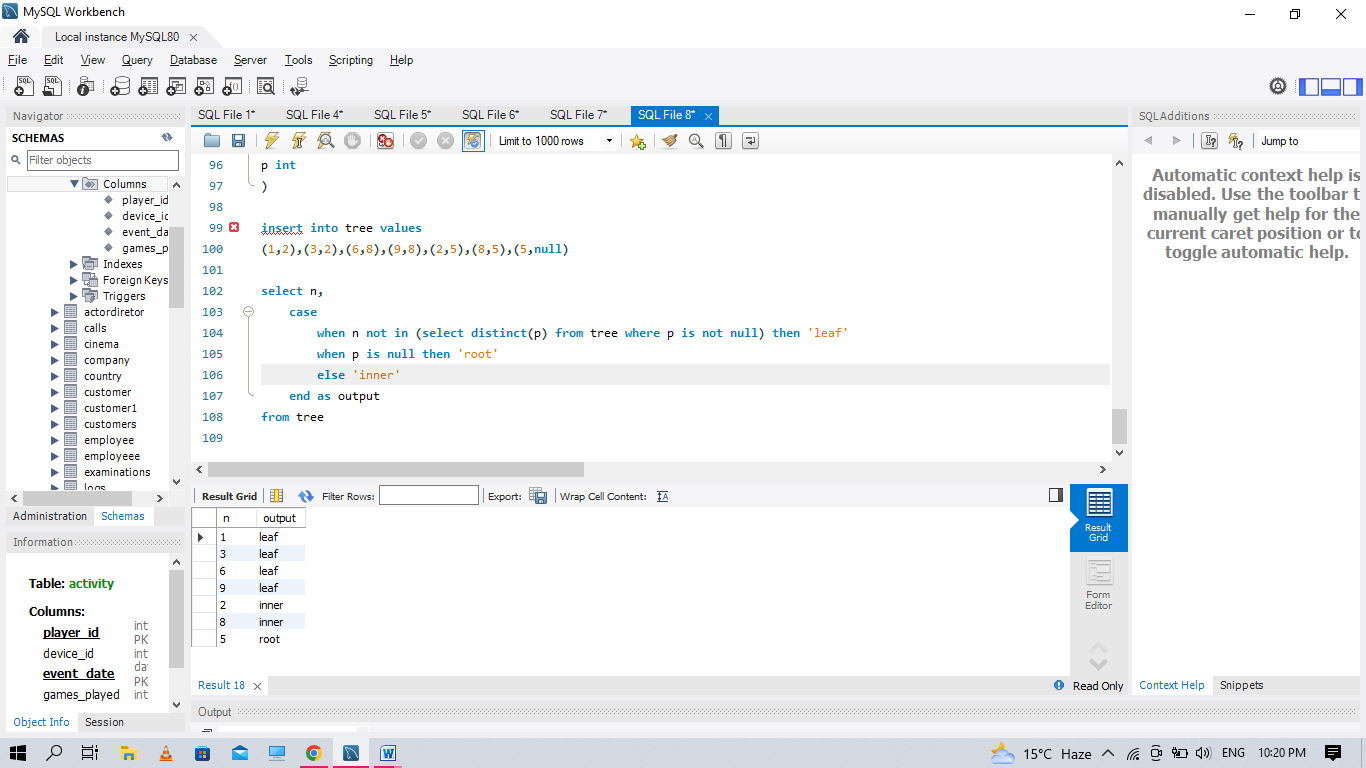
when n not in (select distinct(p) from tree where p is not null) then 'leaf'

when p is null then 'root'

else 'inner'

end as output

from tree



**Q115** . **Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same**

create table students1

(id int,

name varchar(50),

marks int

)

insert into students1 values

(1, 'ashley' ,81),

(2, 'samantha' ,75),

(3, 'julia' ,76),

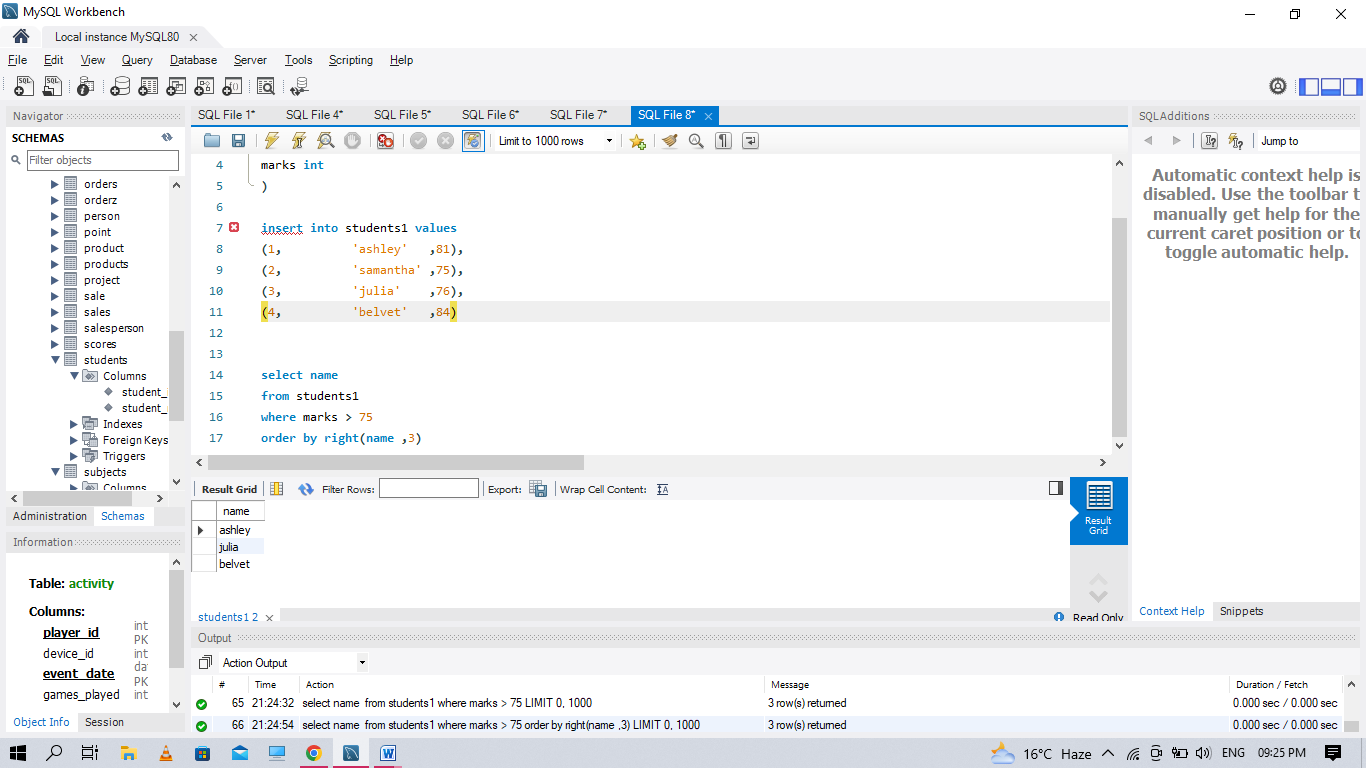
(4, 'belvet' ,84)

select name

from students1

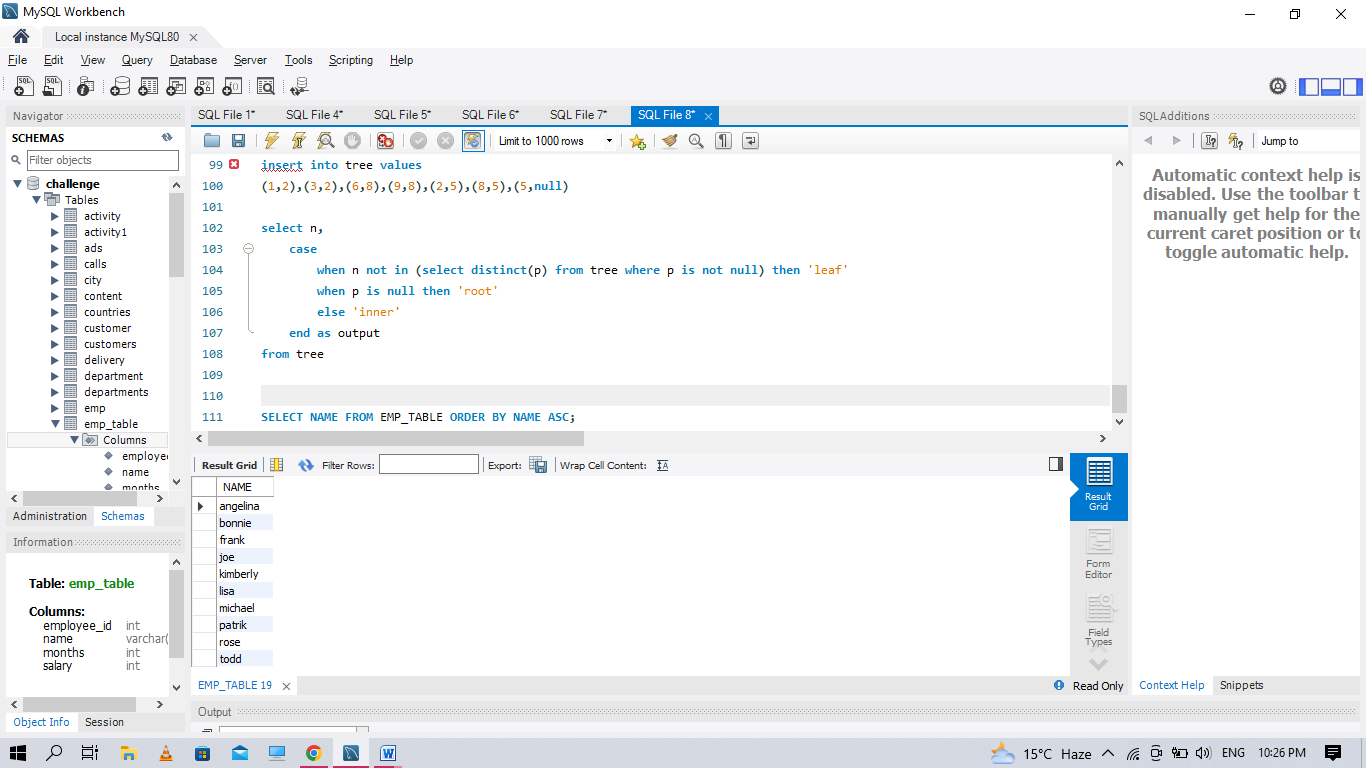
where marks > 75

order by right(name ,3)



**Q116. 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;



**Q117 Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id**.

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),

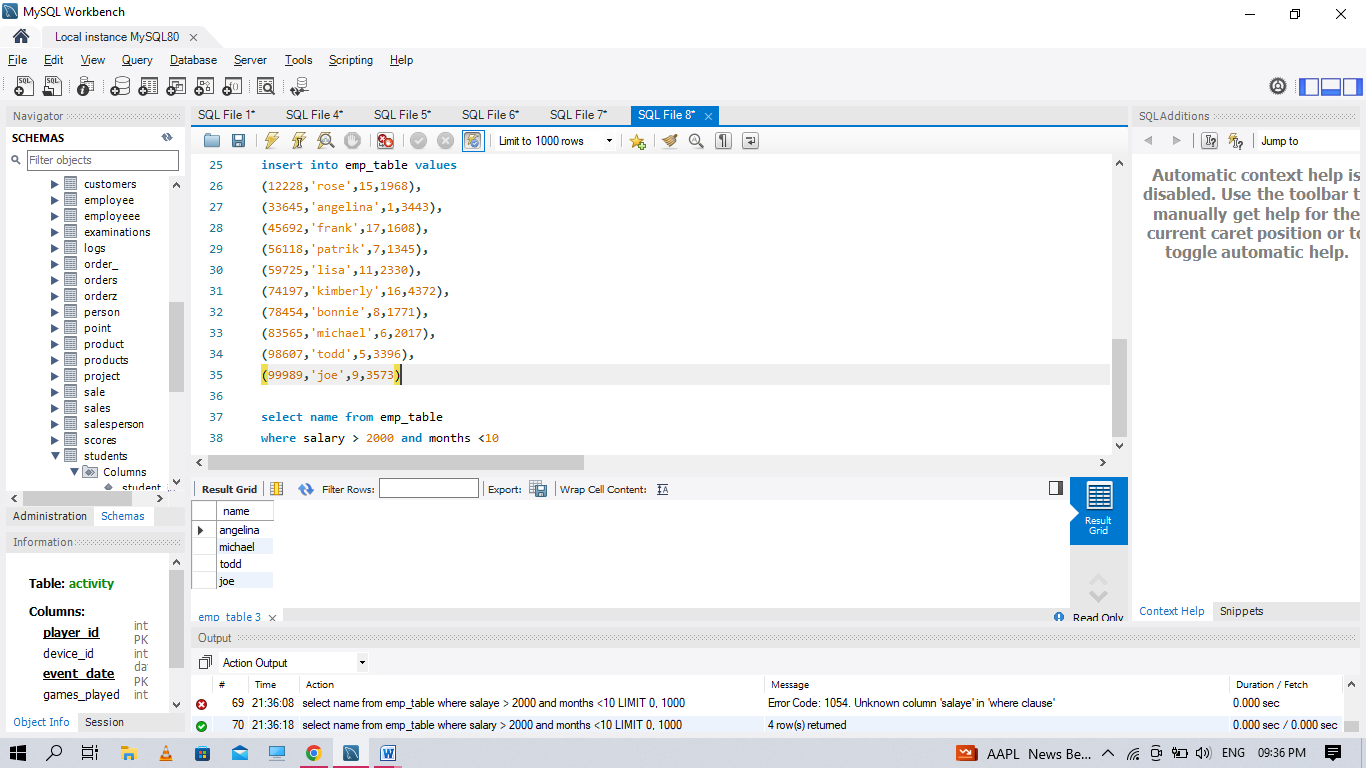
(83565,'michael',6,2017),

(98607,'todd',5,3396),

(99989,'joe',9,3573)

select name from emp\_table

where salary > 2000 and months <10



**Q118 Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table: ● Equilateral: It's a triangle with sides of equal length.**

**● Isosceles: It's a triangle with sides of equal length.**

**● Scalene: It's a triangle with sides of differing lengths.**

**● Not A Triangle: The given values of A, B, and C don't form a triangle**

create table triangle

(a int,

b int,

c int)

insert into triangle values

(20,20,23),(20,20,20),(20,21,22),(13,14,30)

select

case

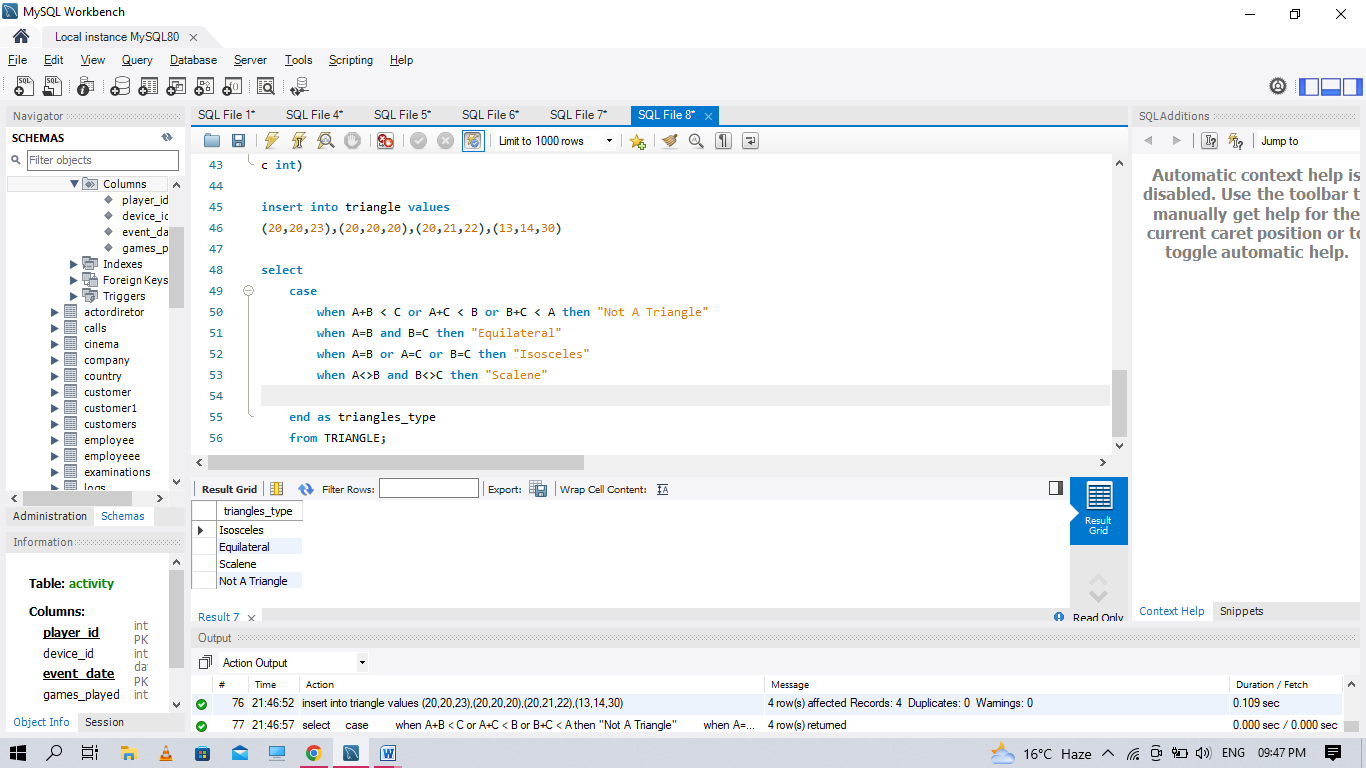
when A+B < C or A+C < B or B+C < A then "Not A Triangle"

when A=B and B=C then "Equilateral"

when A=B or A=C or B=C then "Isosceles"

when A<>B and B<>C then "Scalene"

end as triangles\_type



**Q127 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),

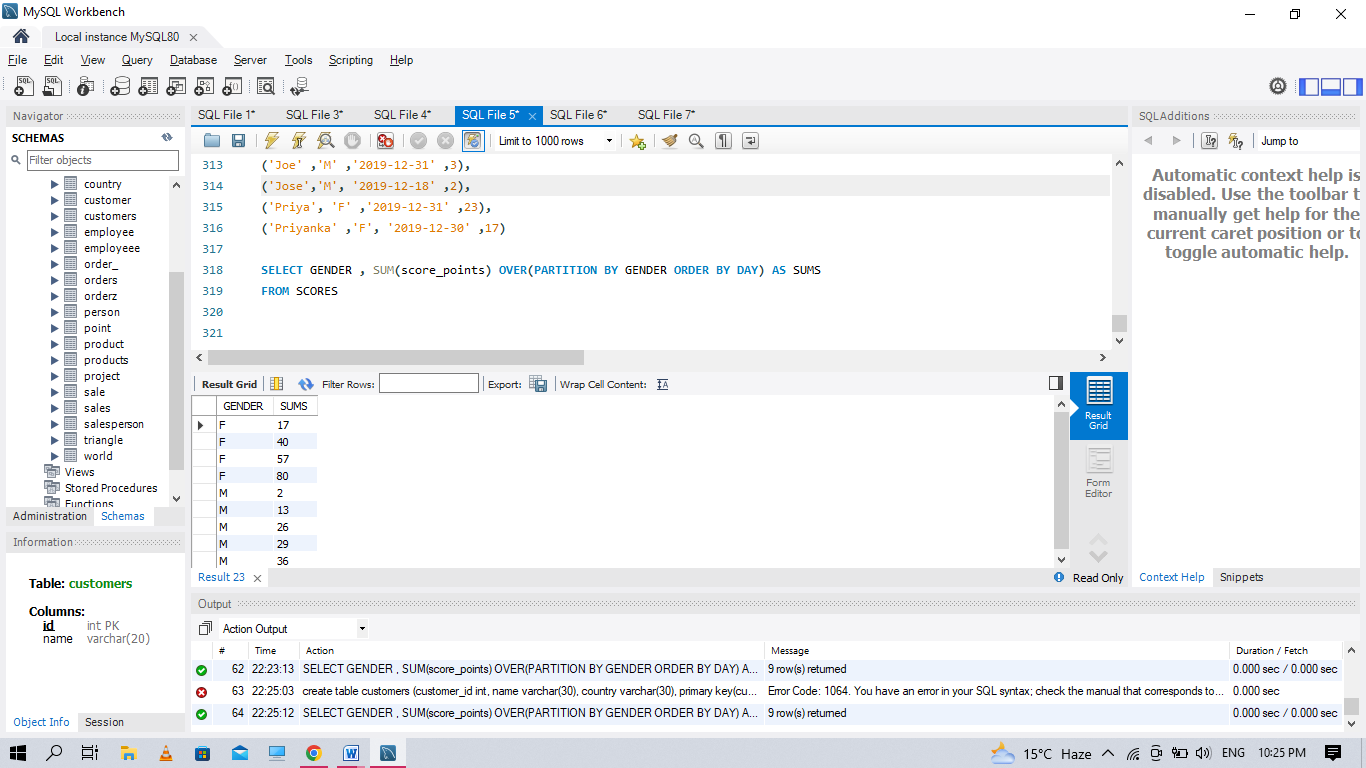
('Jose','M', '2019-12-18' ,2),

('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



**Q128 . 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)

**Q131. Write an SQL query to report for each install date, the number of players that installed the game on that day, and the day one retention**

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)

select a1.event\_date as install\_dt, count(a1.player\_id) as installs, round(count(a3.player\_id) / count(a1.player\_id), 2) as Day1\_retention

from Activity a1 left join Activity a2

on a1.player\_id = a2.player\_id and a1.event\_date > a2.event\_date

left join Activity a3

on a1.player\_id = a3.player\_id and datediff(a3.event\_date, a1.event\_date) = 1

where a2.event\_date is null

group by a1.event\_date;