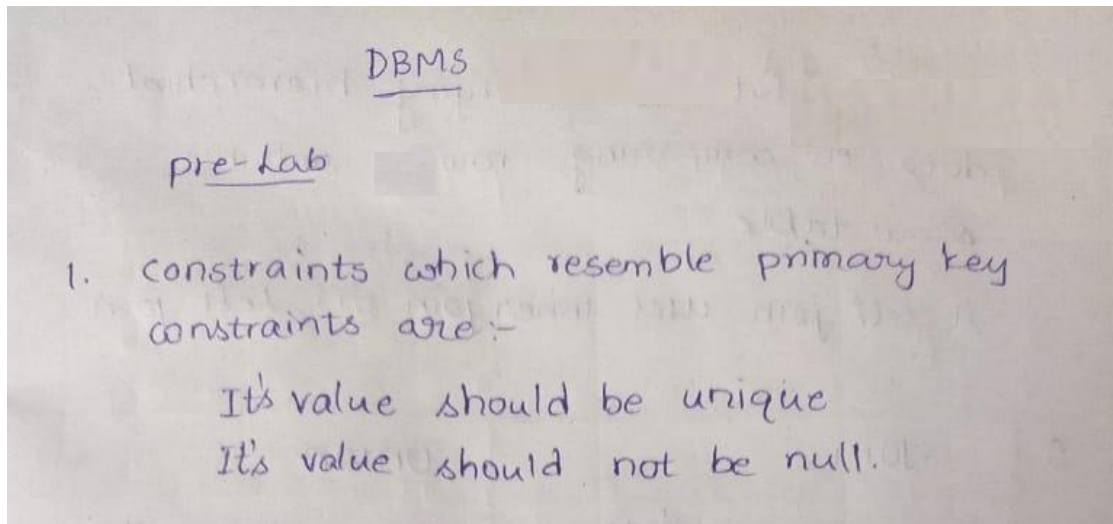


DBMS SKILL-5

PRE-LAB

1. The properties of a primary key are already known. A combination of which individual constraints resembles "Primary Key" constraint?



2. Consider a database table T containing two columns X and Y each of type integer. After the creation of the table, one record (X=1, Y=1) is inserted in the table. Let M_X and M_Y denote the respective maximum values of X and Y among all records in the table at any point in time. Using M_X and M_Y , new records are inserted in the table 128 times with X and Y values being M_X+1 , $2*M_Y+1$ respectively. It may be noted that each time after the insertion, values of M_X and M_Y change. What will be the output of the following SQL query after the steps mentioned above are carried out? Explain.

2.

X	Y
1	1
2	3
3	7
4	15
5	31
6	63
7	127

After performing the operations the output pattern will be of this form

3. Consider the set of relations shown below and the SQL query that follows.
Students: (Roll_number, Name, Date_of_birth)

Courses: (Course number, Course_name, Instructor) Grades: (Roll_number, Course_number, Grade) What is the output of the given SQL query?

select distinct Name **from** Students, Courses, Grades **where** Students. Roll_number = Grades.Roll_number **and** Courses.Instructor = 'Korth' **and** Courses.Course_number = Grades.Course_number **and** Grades.grade = 'A'

3. The output will be the name of students who have got an A grade in atleast one of the course taught by korth.

4. What self join and why it is required?

4. A self join is a regular join, which joins data from the same table (or) it joins a table with itself.

It is useful for querying hierarchial data on comparing row within same table

A self join uses inner join (or) left join

5. State the difference between UNION clause and JOIN ?

5	Join	Union
	Join combines data from many tables based on a matched condition b/w them	SQL combines the result set of two or more select statements
	It combines data into new columns	It combines data into new rows
	No of columns selected from each table may not be same	No of columns selected from each tables should be same
	Datatypes of corresponding columns selected from each table can be different	Datatypes of corresponding columns selected from each table should be same
	It may not return distinct values	It returns distinct values.

6. Classify Outer join operations and explain briefly.

6. The outer join is classified into 3 types :-

- a) Left outer join
- b) Right outer join
- c) Full outer join

Left outer join

It contains the set of tuples of all combinations in R and S that are equal on their common attribute names

In the left outer join, tuples in R have no matching tuples in S

It is denoted by \bowtie

Right Outer join

It contains the set of tuples of all combinations in R and S that are equal on their common attributes names

In right outer join, tuples in S have no matching tuples in R.

It is denoted by \bowtie

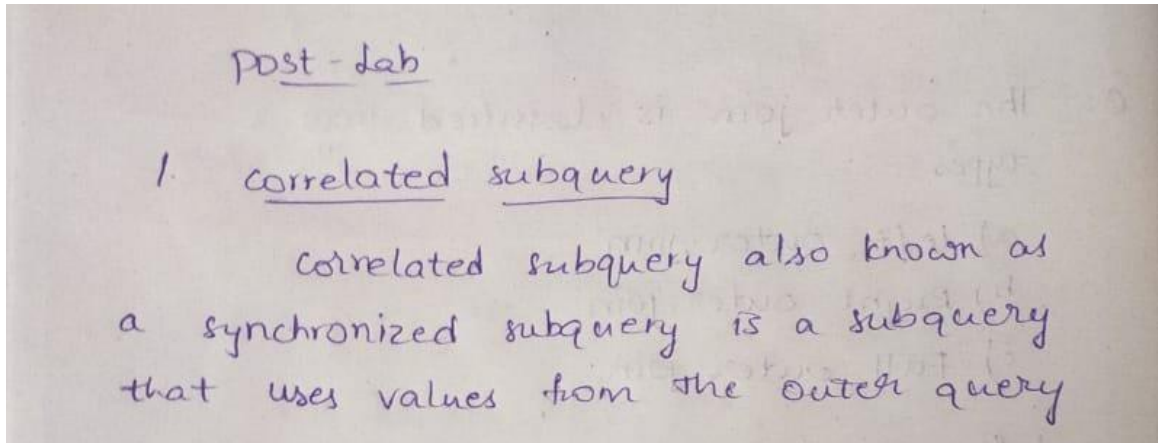
Full outer join

It is like a left (or) right join except that it contains all rows from both tables.

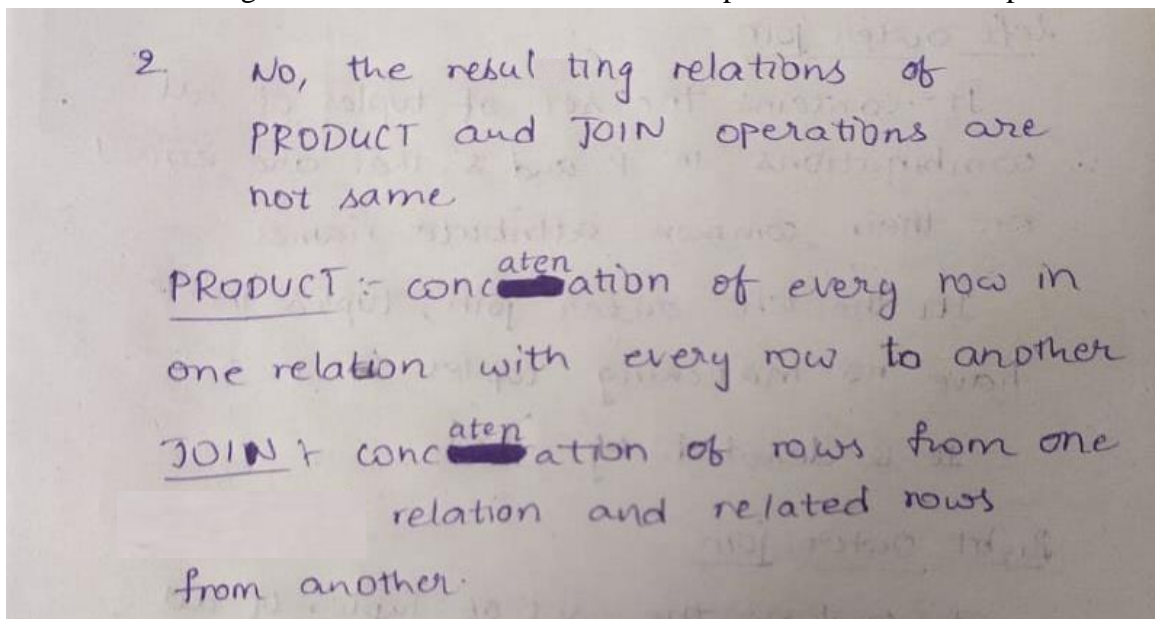
It is denoted by \bowtie

POST LAB 1.

What do you mean by Correlated subquery?



2. Are the resulting relations of PRODUCT and JOIN operation the same? Explain.



3. Explain a join between tables

3. An SQL join clause - corresponding to a join operation in relational algebra combines columns from one or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A JOIN is meant for combining columns from one or more tables by using values common to each.

4. Describe the difference between embedded and dynamic SQL.

4. Embedded SQL
are SQL statements in an application that do not change at runtime and there fans can be hard-coded into the application

Dynamic SQL

Is SQL statements that are construct-ed at runtime. For example, the application may allow users to enter their own queries.

5. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus?

5. A Tuple relational calculus is a non procedural query language which specifies to select the tuples in a relation. It can select the tuples with range of values or tuples for certain attribute values etc. The resulting relation can have one or more tuples.

DBMS SKILL-5

INLAB

- 1) Create tables with the required constraints for the given case study
- 2) Insert 10 records into the created tables

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following SQL statements:

```

1 CREATE TABLE STAFF(STAFF_NO INT,NAME VARCHAR(25),SALARY INT,CITY VARCHAR(25),STATE VARCHAR(25),PHONE BIGINT)
2
3 INSERT INTO STAFF VALUES(50012,'Surya',45000,'Hyderabad','Telangana',6074331464,'sj@gmail.com');
4 INSERT INTO STAFF VALUES(50013,'raju',50000,'Banglore','Karnataka',6158984565,'raju@yahoo.co.in');
5 INSERT INTO STAFF VALUES(50014,'virat',55000,'Vijayawada','Andhra Pradesh',6243637666,'v@yahoo.com');
6 INSERT INTO STAFF VALUES(50015,'Iya',60000,'Chennai','Tamil nadu',6328290767,'pooja@tcs.com');

```

The Result Grid shows the following data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50012	Surya	45000	Hyderabad	Telangana	6074331464	sj@gmail.com
50013	raju	50000	Banglore	Karnataka	6158984565	raju@yahoo.co.in
50014	virat	55000	Vijayawada	Andhra Pradesh	6243637666	v@yahoo.com
50015	Iya	60000	Chennai	Tamil nadu	6328290767	pooja@tcs.com
50016	pooja	65000	kochi	kerala	6412943868	ab@gmail.com
50017	anil	70000	Hyderabad	Telangana	6497596969	anil@gmail.com

The Output pane shows the execution results of the SQL statements, including error messages for invalid group function usage.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following SQL statements:

```

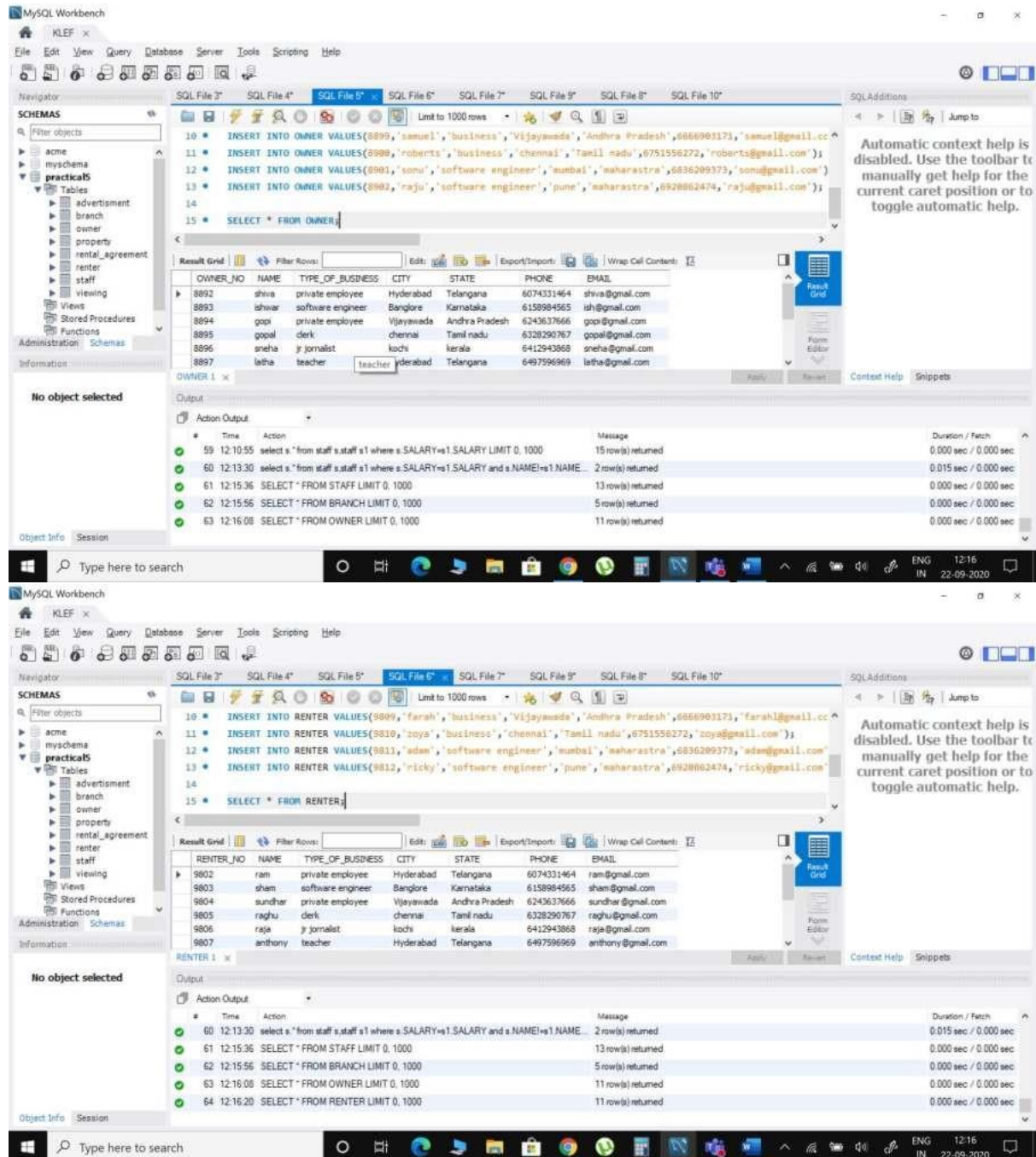
4 INSERT INTO BRANCH VALUES(6623,'Banglore','Karnataka',50018,6075337465,'banglore@gmail.com');
5 INSERT INTO BRANCH VALUES(6624,'Vijayawada','Andhra Pradesh',50014,6076337466,'ap@gmail.com');
6 INSERT INTO BRANCH VALUES(6625,'chennai','Tamil nadu',50015,6077337467,'tn@gmail.com');
7 INSERT INTO BRANCH VALUES(6626,'kochi','kerala',50016,6078337468,'kochi@gmail.com');
8
9 SELECT * FROM BRANCH;

```

The Result Grid shows the following data:

BRANCH_NO	CITY	STATE	MANAGER	BRANCH_PHONE	BRANCH_EMAIL
6622	Hyderabad	Telangana	50017	6074337464	Hyderabad@gmail.com
6623	Banglore	Karnataka	50018	6075337465	banglore@gmail.com
6624	Vijayawada	Andhra Pradesh	50014	6076337466	ap@gmail.com
6625	chennai	Tamil nadu	50015	6077337467	tn@gmail.com
6626	kochi	kerala	50016	6078337468	kochi@gmail.com

The Output pane shows the execution results of the SQL statements, including error messages for invalid group function usage.



MySQL Workbench

Navigator

SCHEMAS

Filter objects

myschema

practica5

Tables

advertisement

branch

owner

property

rental_agreement

renter

staff

viewing

Views

Stored Procedures

Functions

Administration

Schemas

Information

No object selected

SQL File 3

SQL File 4

SQL File 5

SQL File 6

SQL File 7

SQL File 8

SQL File 10

Limit to 1000 rows

10 * INSERT INTO PROPERTY VALUES(154589,'vijayawada',8899,50018);

11 * INSERT INTO PROPERTY VALUES(167589,'chennai',8900,50019);

12 * INSERT INTO PROPERTY VALUES(180589,'mumbai',8901,50013);

13 * INSERT INTO PROPERTY VALUES(193589,'pune',8902,50018);

14

15 * SELECT * FROM PROPERTY;

Result Grid

PROPERTY_NO	CITY	OWNED_BY	OVERSEEN_BY
63589	Hyderabad	8892	50012
76589	Bangalore	8893	50013
89589	Vijayawada	8894	50014
102589	chennai	8895	50015
115589	kochi	8896	50016
128589	Hyderabad	8897	50017

Output

Action Output

#	Time	Action	Message	Duration / Fetch
61	12-15-36	SELECT * FROM STAFF LIMIT 0, 1000	13 row(s) returned	0.000 sec / 0.000 sec
62	12-15-56	SELECT * FROM BRANCH LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
63	12-16-08	SELECT * FROM OWNER LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
64	12-16-20	SELECT * FROM RENTER LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
65	12-16-32	SELECT * FROM PROPERTY LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec

Object Info

Session

MySQL Workbench

Navigator

SCHEMAS

Filter objects

myschema

practica5

Tables

advertisement

branch

owner

property

rental_agreement

renter

staff

viewing

Views

Stored Procedures

Functions

Administration

Schemas

Information

No object selected

SQL File 3

SQL File 4

SQL File 5

SQL File 6

SQL File 7

SQL File 9

SQL File 8

SQL File 10

Limit to 1000 rows

10 * INSERT INTO ADVERTISEMENT VALUES(29,'08-Jun-20','times',154589);

11 * INSERT INTO ADVERTISEMENT VALUES(30,'09-Jun-20','sakshi',167589);

12 * INSERT INTO ADVERTISEMENT VALUES(31,'10-Jun-20','dc',180589);

13 * INSERT INTO ADVERTISEMENT VALUES(32,'11-Jun-20','hinda',193589);

14

15 * SELECT * FROM ADVERTISEMENT;

Result Grid

AD_ID	AD_DATE	PAPER	PROPERTY_NO
22	01-Jun-20	hindu	63589
23	02-Jun-20	eenadu	76589
24	03-Jun-20	times	89589
25	04-Jun-20	sakshi	102589
26	05-Jun-20	dc	115589
27	06-Jun-20	hindu	128589

Output

Action Output

#	Time	Action	Message	Duration / Fetch
62	12-15-56	SELECT * FROM BRANCH LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
63	12-16-08	SELECT * FROM OWNER LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
64	12-16-20	SELECT * FROM RENTER LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
65	12-16-32	SELECT * FROM PROPERTY LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
66	12-17-13	SELECT * FROM ADVERTISEMENT LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec

Object Info

Session

The first screenshot shows a MySQL Workbench interface with a SQL query in SQL File 10. The query is:

```

10 * INSERT INTO VIEWING VALUES(154589,9889,'31-Jul-20');
11 * INSERT INTO VIEWING VALUES(167589,9889,'01-Aug-20');
12 * INSERT INTO VIEWING VALUES(189589,9811,'02-Aug-20');
13 * INSERT INTO VIEWING VALUES(193589,9812,'03-Aug-20');
14
15 * SELECT * FROM VIEWING;

```

The result grid shows the following data:

PROPERTY_NO	RENTER_NO	VIEWING_DATE
63589	9802	24-Jul-20
76589	9812	25-Jul-20
85589	9804	26-Jul-20
102589	9811	27-Jul-20
115589	9806	28-Jul-20
128589	9807	29-Jul-20

The second screenshot shows the same MySQL Workbench interface with a new SQL query in SQL File 10. The query is:

```

10 * INSERT INTO RENTAL_AGREEMENT VALUES(2363,154589,'31-Aug-20','11-Sep-20','11-Sep-22',9889);
11 * INSERT INTO RENTAL_AGREEMENT VALUES(2364,167589,'01-Sep-20','12-Sep-20','12-Sep-22',9889);
12 * INSERT INTO RENTAL_AGREEMENT VALUES(2365,189589,'02-Sep-20','13-Sep-20','13-Sep-22',9811);
13 * INSERT INTO RENTAL_AGREEMENT VALUES(2366,193589,'03-Sep-20','14-Sep-20','14-Sep-22',9812);
14
15 * select * from rental_agreement;

```

The result grid shows the following data:

RENTAL_NO	PROPERTY_NO	SIGNING_DATE	START_DATE	END_DATE	RENTER_NO
2356	63589	24-Aug-20	04-Sep-20	04-Sep-22	9802
2357	76589	25-Aug-20	05-Sep-20	05-Sep-22	9812
2358	89589	26-Aug-20	06-Sep-20	06-Sep-22	9804
2359	102589	27-Aug-20	07-Sep-20	07-Sep-22	9811
2360	115589	28-Aug-20	08-Sep-20	08-Sep-22	9806
2361	128589	29-Aug-20	09-Sep-20	09-Sep-22	9807

3) Display renter details which are unique.

MySQL Workbench interface showing a query result grid. The query executed is `SELECT DISTINCT * FROM RENTER;`. The result grid displays the following data:

RENTER_NO	NAME	TYPE_OF_BUSINESS	CITY	STATE	PHONE	EMAIL
9802	ram	private employee	Hyderabad	Telangana	6074331464	ram@gmail.com
9803	sham	software engineer	Bangalore	Karnataka	6158984565	sham@gmail.com
9804	sundhar	private employee	Vijayawada	Andhra Pradesh	6243637666	sundhar@gmail.com
9805	raghu	clerk	Chennai	Tamil Nadu	6328290767	raghu@gmail.com
9806	raja	journalist	Kochi	Kerala	6412943868	raja@gmail.com
9807	anthony	teacher	Hyderabad	Telangana	6497596969	anthony@gmail.com

The Action Output pane shows the execution of several queries, including the current one, with messages indicating the number of rows returned.

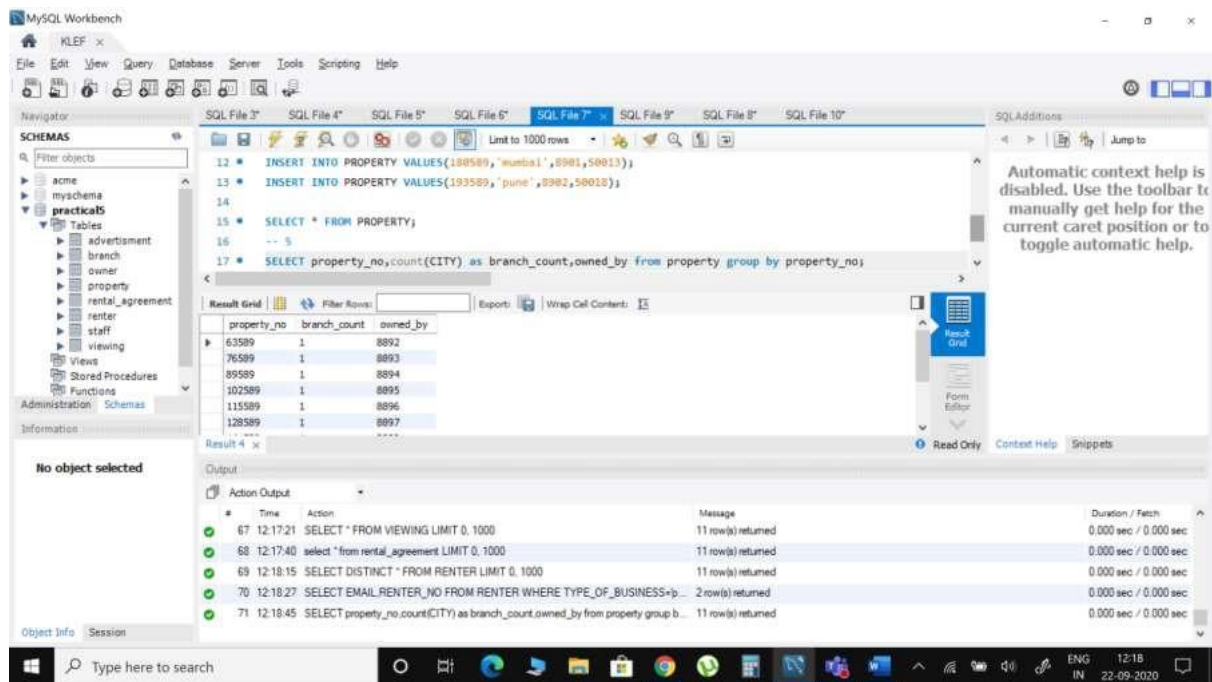
- 4) Give the email addresses and the renter number for all the private renters. Please, sort them by the renter number.

MySQL Workbench interface showing a query result grid. The query executed is `SELECT EMAIL, RENTER_NO FROM RENTER WHERE TYPE_OF_BUSINESS='private employee' ORDER BY RENTER_NO;`. The result grid displays the following data:

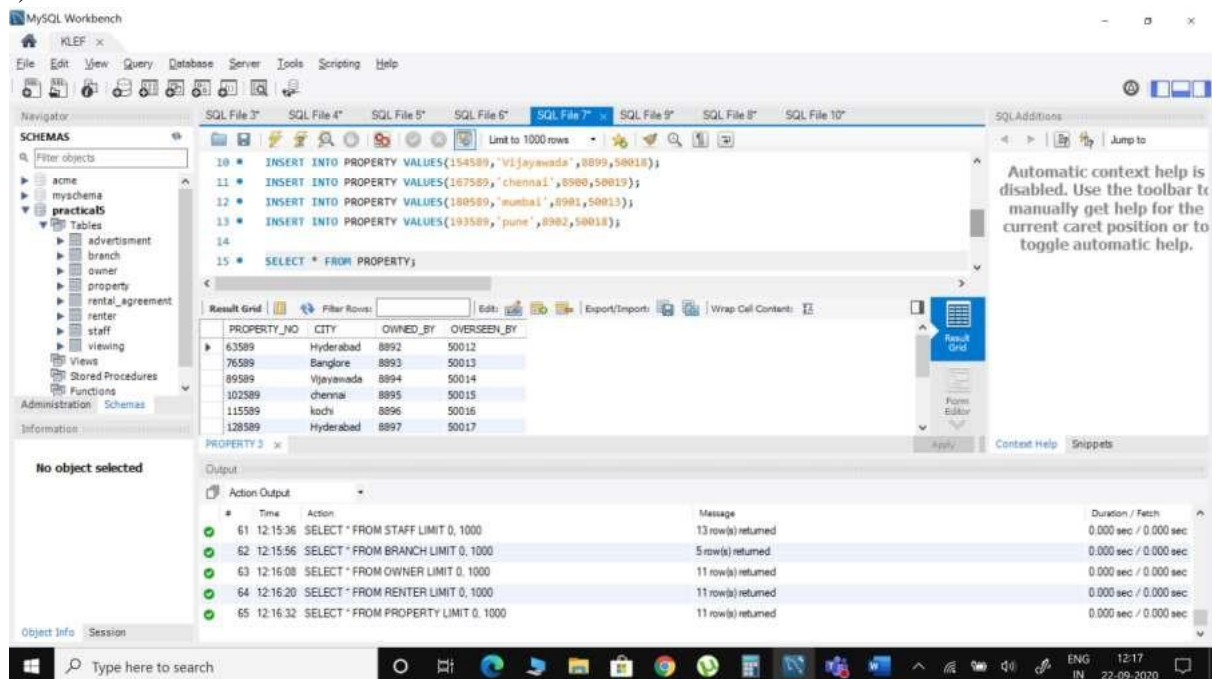
EMAIL	RENTER_NO
ram@gmail.com	9802
sundhar@gmail.com	9804

The Action Output pane shows the execution of several queries, including the current one, with messages indicating the number of rows returned.

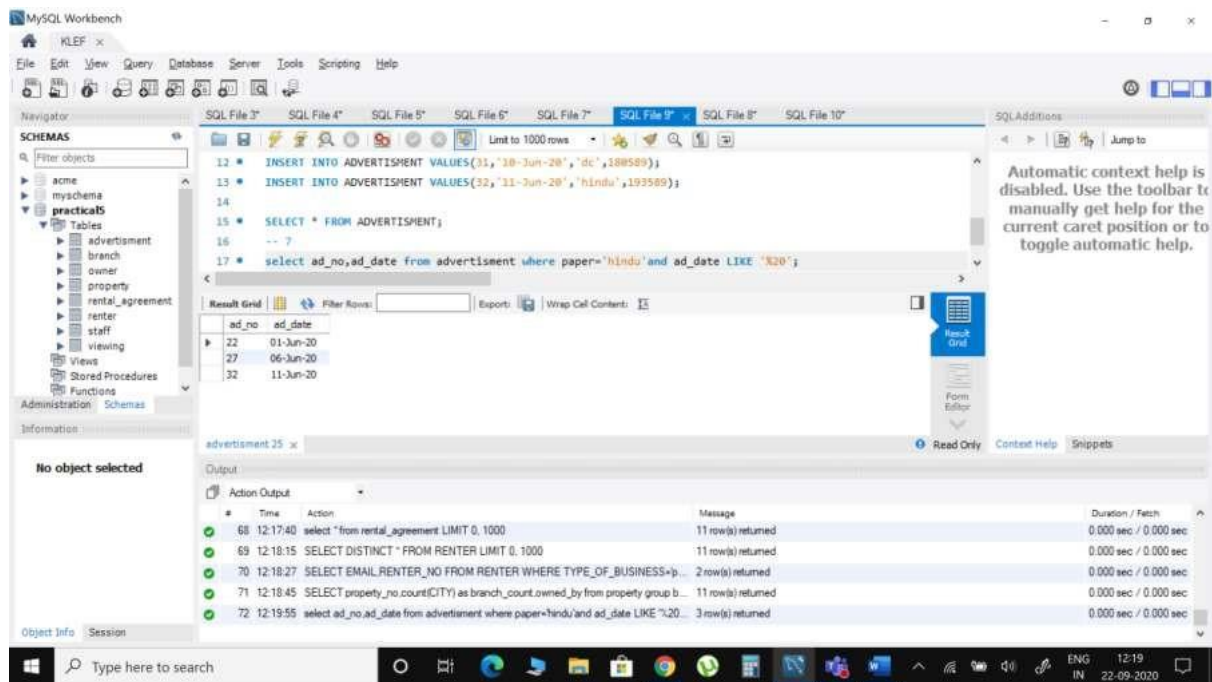
- 5) Find unique property name and number of branches for each property.



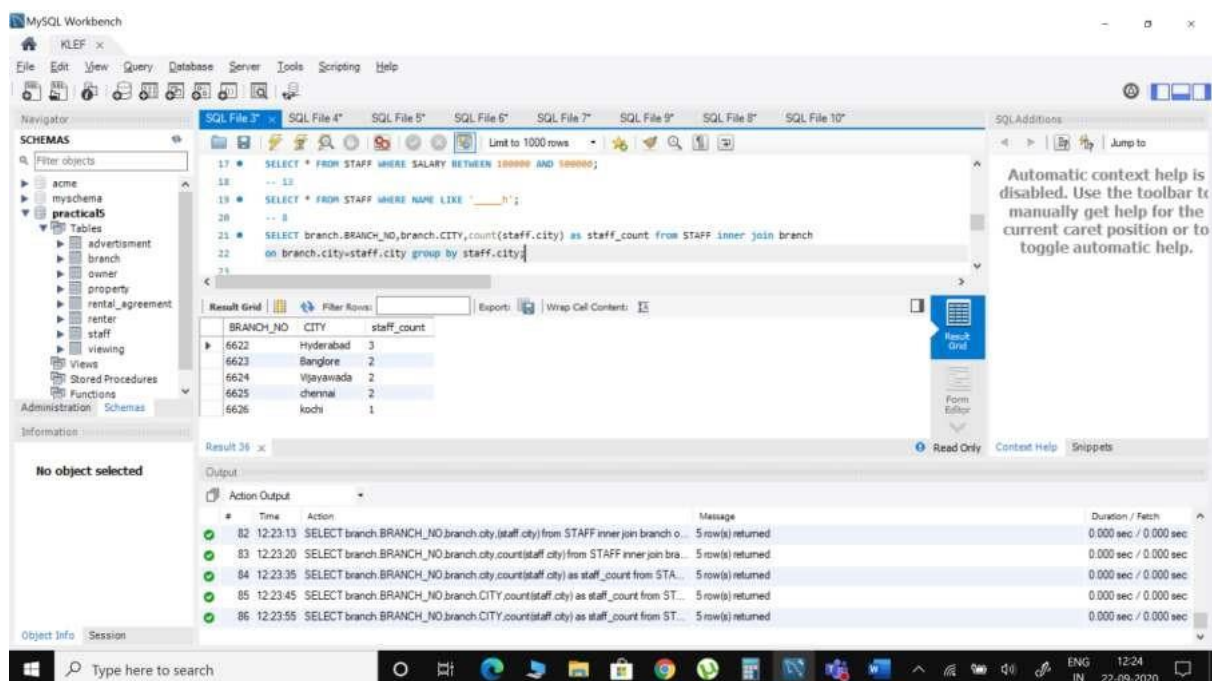
6) Create table for staff member and insert all the details of the staff members.



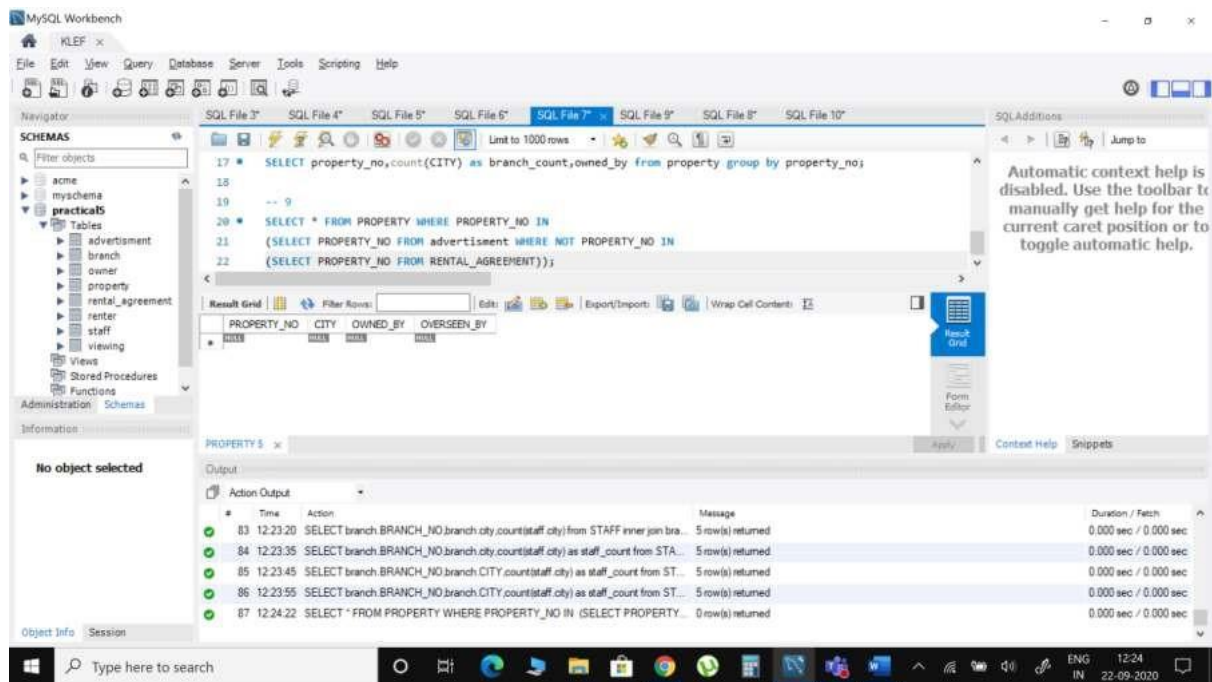
7) Give the dates of all the advertisements posted in THE GLOBE AND MAIL in 2005.



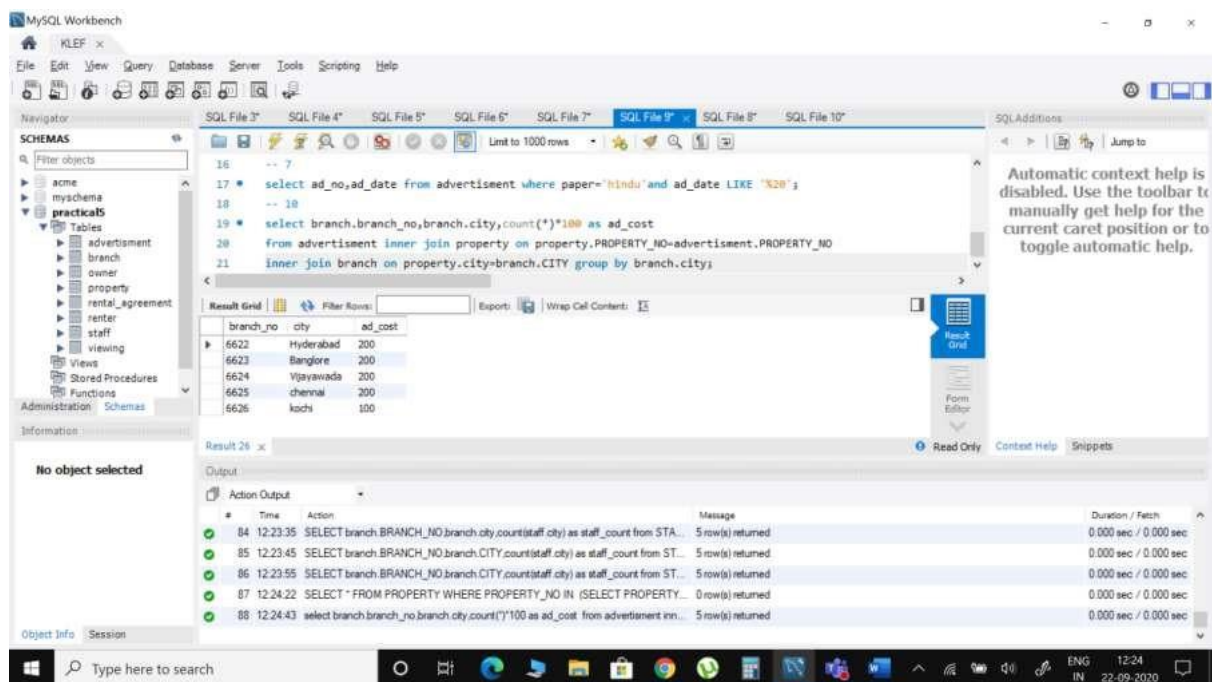
8) Display the count of staff in each branch and display them in descending order on count.



9) Find the properties that are already advertised but not yet rented.



- 10) Assuming that each advertisement costs 100 dollars, give the branch number and the amount spent on the advertisements for each branch. Name the branch number as Branch no, and the amount as ad cost.



- 11) Display the details of staff who are working in a particular branch.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

SELECT branch.BRANCH_NO, branch.CITY, count(staff.city) as staff_count from STAFF inner join branch
on branch.city=staff.city group by staff.city;

SELECT STAFF.* from STAFF inner join BRANCH on STAFF.city=BRANCH.city and BRANCH.CITY='chennai';

```

The Result Grid displays the following data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50015	laya	60000	Chennai	Tamil nadu	6328290767	poorja@fcs.com
50020	sowmya	40000	Chennai	Tamil nadu	6751556272	sowmya@gmail.com

The Action Output pane shows the execution of the queries with their respective durations and messages.

12) Write down a query to find out the Name, Address and Position of the branch staff whose salary is the second highest without using TOP or limit method.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

SELECT STAFF.* from STAFF inner join BRANCH on STAFF.city=BRANCH.city and BRANCH.CITY='chennai';

SELECT * from staff where salary in (select MAX(SALARY) from staff where salary not in (select MAX(salary) from staff));

```

The Result Grid displays the following data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50022	mujahed	95000	pune	maharashtra	6920862474	mujahed@gmail.com

The Action Output pane shows the execution of the queries with their respective durations and messages.

13) Write a query to print details of the staff whose Name ends with 'h' and contains six alphabets.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

17 SELECT * FROM STAFF WHERE SALARY BETWEEN 100000 AND 500000;
18 -- 13
19 SELECT * FROM STAFF WHERE NAME LIKE '____h';
20 -- 8
21 SELECT branch.BRANCH_NO,branch.CITY,count(staff.city) as staff_count from STAFF inner join branch
22 on branch.city=staff.city group by staff.city;

```

The Result Grid shows the following data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50023	rakesh	100000	pune	maharashtra	6863209733	rakesh@gmail.com

The Action Output pane shows the execution of the query:

#	Time	Action	Message	Duration / Fetch
87	12:24:22	SELECT * FROM PROPERTY WHERE PROPERTY_NO IN (SELECT PROPERTY...	0 row(s) returned	0.000 sec / 0.000 sec
88	12:24:43	select branch.branch_no,branch.city,count(*) as ad_cost from advertisement inn...	5 row(s) returned	0.000 sec / 0.000 sec
89	12:25:03	SELECT STAFF * from STAFF inner join BRANCH on STAFF.city=BRANCH.city and...	2 row(s) returned	0.000 sec / 0.000 sec
90	12:25:19	SELECT * from staff where salary in (select MAX(SALARY) from staff where salary no...	1 row(s) returned	0.000 sec / 0.000 sec
91	12:25:48	SELECT * FROM STAFF WHERE NAME LIKE '____h' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

14) Write a query to print details of the staff whose SALARY lies between 100000 and 500000.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

12 INSERT INTO STAFF VALUES(50022,'robert','99999','mumbai','maharashtra','6863209733','robert@gmail.com');
13 INSERT INTO STAFF VALUES(50022,'mujahed','95000','pune','maharashtra','952002474','mujahed@gmail.com');
14 INSERT INTO STAFF VALUES(50023,'rakesh','100000','pune','maharashtra','6863209733','rakesh@gmail.com');
15 SELECT * FROM STAFF;
16 -- 14
17 SELECT * FROM STAFF WHERE SALARY BETWEEN 100000 AND 500000;
18 -- 13

```

The Result Grid shows the following data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50023	rakesh	100000	pune	maharashtra	6863209733	rakesh@gmail.com

The Action Output pane shows the execution of the query:

#	Time	Action	Message	Duration / Fetch
88	12:24:43	select branch.branch_no,branch.city,count(*) as ad_cost from advertisement inn...	5 row(s) returned	0.000 sec / 0.000 sec
89	12:25:03	SELECT STAFF * from STAFF inner join BRANCH on STAFF.city=BRANCH.city and...	2 row(s) returned	0.000 sec / 0.000 sec
90	12:25:19	SELECT * from staff where salary in (select MAX(SALARY) from staff where salary no...	1 row(s) returned	0.000 sec / 0.000 sec
91	12:25:48	SELECT * FROM STAFF WHERE NAME LIKE '____h' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
92	12:25:57	SELECT * FROM STAFF WHERE SALARY BETWEEN 100000 AND 500000 LIMIT...	1 row(s) returned	0.000 sec / 0.000 sec

15) Write a query to display the list of employees who draw same salary

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'practica5' selected. The main editor window shows a SQL query in 'SQL File 3'. The query is as follows:

```
27 -- 12
28 * SELECT * from staff where salary in (select MAX(SALARY) from staff where salary not in (select MAX(salary) from staff));
29
30 -- 15:
31 * INSERT INTO STAFF VALUES(50024,'RK',45000,'Hyderabad','Telangana',6074331446,'rk@gmail.com');
32 * select s.* from staff s,staff s1 where s.SALARY=s1.SALARY and s.NAME!=s1.NAME;
```

The 'Result Grid' shows the results of the last query (Query 32), displaying two rows of staff data:

STAFF_NO	NAME	SALARY	CITY	STATE	PHONE	EMAIL
50024	RK	45000	Hyderabad	Telangana	6074331446	rk@gmail.com
50012	Surya	45000	Hyderabad	Telangana	6074331464	sj@gmail.com

The bottom panel shows the 'Action Output' tab, which lists the execution of several SQL statements and their results:

#	Time	Action	Message	Duration / Fetch
89	12:25:03	SELECT STAFF 'from STAFF inner join BRANCH on STAFF.city=BRANCH.city and ...	2 row(s) returned	0.000 sec / 0.000 sec
90	12:25:19	SELECT * from staff where salary in (select MAX(SALARY) from staff where salary no...	1 row(s) returned	0.000 sec / 0.000 sec
91	12:25:48	SELECT * FROM STAFF WHERE NAME LIKE '_____' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
92	12:25:57	SELECT * FROM STAFF WHERE SALARY BETWEEN 100000 AND 500000 LIMIT...	1 row(s) returned	0.000 sec / 0.000 sec
93	12:26:11	select s.* from staff s,staff s1 where s.SALARY=s1.SALARY and s.NAME!=s1.NAME...	2 row(s) returned	0.000 sec / 0.000 sec

The bottom status bar shows the system clock as 12:26 IN 22-09-2020.