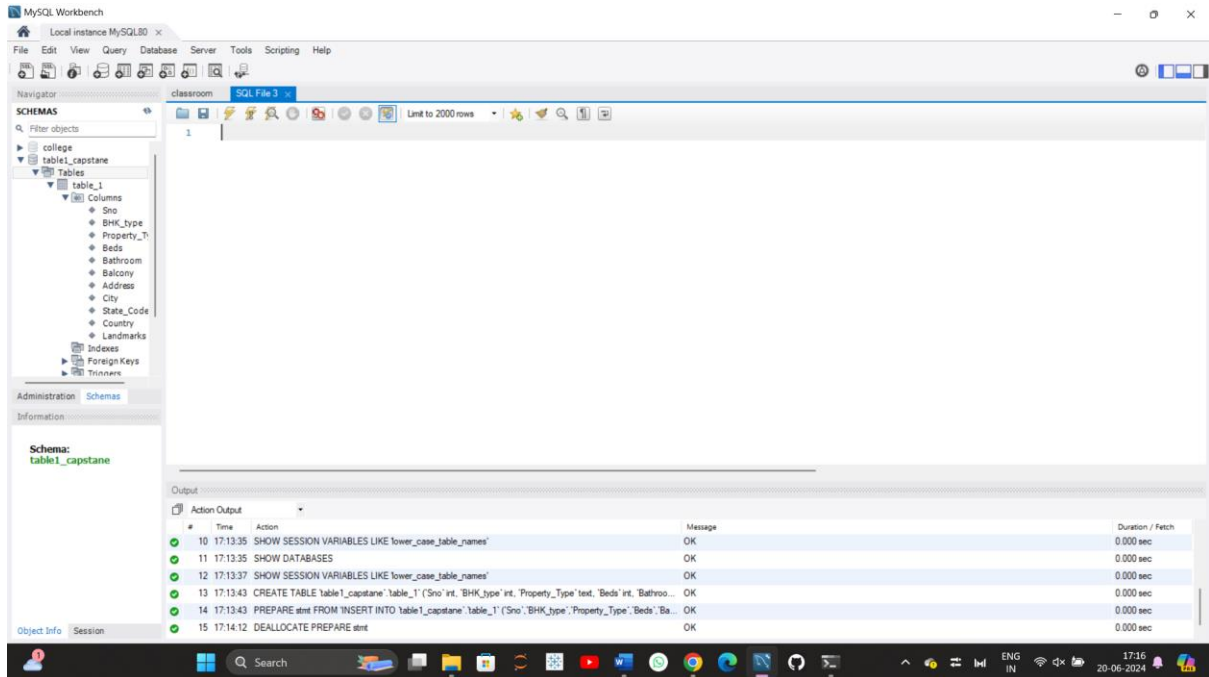


Write the SQL queries

Table1

Successfully imported table_1.csv to my sql server.



- 1- Retrieve properties with balconies, sorted by the number of bedrooms in descending order.

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

```
1 USE table1_capstane;
2
3 -- Retrieve properties with balconies, sorted by the number of bedrooms in descending order:
4 SELECT *
5 FROM table_1
6 WHERE Balcony IS NOT NULL AND Balcony != ''
7 ORDER BY Beds DESC;
```

The Result Grid displays the following data:

Sno	BHK_Type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
226	4	Independent House	4	3	2.0	Independent House, Nagappa street near Alah...	Bengaluru	KA	India	Safina Business Park
234	4	Standalone Building	4	6	1.0	Standalone building, Nagappa St near raamu ar...	Bengaluru	KA	India	Safina Business Park
237	4	Independent House	4	6	3.0	Independent House, Sarkey Main Rd near The ...	Bengaluru	KA	India	Safina Business Park
270	4	Independent House	4	9	2.0	Independent House, Sarkey Rd near Shil pet...	Bengaluru	KA	India	Safina Business Park
271	4	Independent House	4	6	3.0	Independent House, Sarkey Main Rd near The ...	Bengaluru	KA	India	Safina Business Park
281	4	Shree	4	3	3.0	C. V. Raman Avenue, 5th Cross Road, , near R...	Bengaluru	KA	India	Cauvery Theatre
287	4	Independent House	4	6	3.0	Independent House, 2H2G+HQQ, Swimming Po...	Bengaluru	KA	India	Safina Business Park
299	4	Standalone Building	4	6	1.0	Standalone building, Nagappa St near raamu ar...	Bengaluru	KA	India	Safina Business Park
301	4	Standalone Building	4	4	1.0	Standalone Building, Sampangram Nagar near j...	Bengaluru	KA	India	Safina Business Park
316	4	Thimula	4	3	2.0	nagappa street	Bengaluru	KA	India	Safina Business Park
318	4	Independent House	4	3	1.0	Independent House, Yalappa Garden Near Sni...	Bengaluru	KA	India	Safina Business Park

The Output pane shows the following messages:

- 20 17:24:03 USE table1_capstane Error Code: 1049. Unknown database 'table1_capstane' 0.000 sec / 0.000 sec
- 21 17:24:20 USE table1_capstane Error Code: 1049. Unknown database 'table1_capstane' 0.000 sec / 0.000 sec
- 22 17:24:58 USE table1_capstane 0 row(s) affected 0.000 sec / 0.000 sec
- 23 17:25:24 SELECT * FROM table_1 WHERE Balcony IS NOT NULL ORDER BY Beds DESC LIMIT 0, 2000 2000 row(s) returned 0.016 sec / 0.016 sec
- 24 17:31:52 -1. Retrieve properties with balconies, sorted by the number of bedrooms in descending order: SELECT * FROM... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser... 0.000 sec / 0.000 sec
- 25 17:32:07 SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != '' ORDER BY Beds DESC LIMIT 0... 1676 row(s) returned 0.016 sec / 0.000 sec

- 2- Find the top 5 cities with the highest average number of bedrooms per property.

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

```
8
9
10 -- Find the top 5 cities with the highest average number of bedrooms per property:
11 SELECT City, AVG(Beds) AS Average_Beds
12 FROM table_1
13 GROUP BY City
14 ORDER BY Average_Beds DESC
15 LIMIT 5;
```

The Result Grid displays the following data:

City	Average_Beds
Bengaluru	2.7708
Gurgaon	2.7365
Chicago	2.7196
Hyderabad	2.6696
Faridabad	2.6309

The Output pane shows the following messages:

- 22 17:24:58 USE table1_capstane 0 row(s) affected 0.000 sec / 0.000 sec
- 23 17:25:24 SELECT * FROM table_1 WHERE Balcony IS NOT NULL ORDER BY Beds DESC LIMIT 0, 2000 2000 row(s) returned 0.016 sec / 0.016 sec
- 24 17:31:52 -1. Retrieve properties with balconies, sorted by the number of bedrooms in descending order: SELECT * FROM... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser... 0.000 sec / 0.000 sec
- 25 17:32:07 SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != '' ORDER BY Beds DESC LIMIT 0... 1676 row(s) returned 0.016 sec / 0.000 sec
- 26 17:35:31 SELECT City, AVG(Beds) AS Average_Beds FROM table1 GROUP BY City ORDER BY Average_Beds DESC L... Error Code: 1146. Table 'table1_capstane.table1' doesn't exist 0.000 sec / 0.000 sec
- 27 17:36:06 SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC 5 row(s) returned 0.016 sec / 0.000 sec

3- Count the number of properties in each city.

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

```
-- 3. Count the number of properties in each city:--  
  
SELECT City, COUNT(*) AS Property_Count  
FROM table_1  
GROUP BY City;
```

The Result Grid shows the following data:

City	Property_Count
Chicago	214
Bengaluru	336
Pune	196
Mumbai	417
Chennai	255
Delhi	513
Gurgaon	353
Hyderabad	336
Noida	217
Faridabad	680
Greater Noida	166

The Output pane shows the execution log with the following entries:

- 23 17:25:24 SELECT * FROM table_1 WHERE Balcony IS NOT NULL ORDER BY Beds DESC LIMIT 0, 2000 2000 row(s) returned 0.016 sec / 0.016 sec
- 24 17:31:52 --1. Retrieve properties with balconies, sorted by the number of bedrooms in descending order: SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != "" ORDER BY Beds DESC LIMIT 0, 2000 1676 row(s) returned 0.016 sec / 0.000 sec
- 25 17:32:07 SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != "" ORDER BY Beds DESC LIMIT 0, 2000 1676 row(s) returned 0.016 sec / 0.000 sec
- 26 17:35:31 SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC LIMIT 0, 2000 5 row(s) returned 0.000 sec
- 27 17:36:06 SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC LIMIT 0, 2000 5 row(s) returned 0.016 sec / 0.000 sec
- 28 17:38:36 SELECT City, COUNT(*) AS Property_Count FROM table_1 GROUP BY City LIMIT 0, 2000 11 row(s) returned 0.016 sec / 0.000 sec

4- Retrieve all properties with at least 3 bedrooms and 2 bathrooms.

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

```
-- 4. Retrieve all properties with at least 3 bedrooms and 2 bathrooms:--  
  
SELECT *  
FROM table_1  
WHERE Beds >= 3 AND Bathroom >= 2;
```

The Result Grid shows the following data:

Sno	BHK_Type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
1	3	Condo	3	2	Balcony	2243 North Lister Avenue #402	Chicago	IL 60614	United States	CHI - Logan Square
2	4	Condo	4	3	Balcony	3213 North Elston Avenue #1A	Chicago	IL 60618	United States	CHI - Avondale
4	4	Condo	4	4	Balcony	1438 North Park Avenue #1	Chicago	IL 60610	United States	Cook
5	3	Condo	3	2.5	Balcony	443 West Grant Place #C	Chicago	IL 60614	United States	Cook
6	3	Condo	3	3.5	Balcony	352 West Huron Street #8	Chicago	IL 60654	United States	Cook
7	3	Condo	3	2	Balcony	3024 West Armitage Avenue #8	Chicago	IL 60647	United States	CHI - Logan Square
11	5	Condo	5	3	Balcony	2230 North Kenmore Avenue #1	Chicago	IL 60614	United States	CHI - Lincoln Park
12	3	Condo	3	2	Balcony	1951 West Belmont Avenue #2W	Chicago	IL 60618	United States	CHI - North Center
13	4	Condo	4	4	Balcony	1460 North Sandburg Terrace	Chicago	IL 60618	United States	CHI - Near North Side
15	3	Condo	3	2	Balcony	3708 West Belmont Avenue #2	Chicago	IL 60618	United States	CHI - Avondale
16	3	Condo	3	2	Balcony	3708 West Belmont Avenue #3	Chicago	IL 60618	United States	CHI - Avondale

The Output pane shows the execution log with the following entries:

- 24 17:31:52 --1. Retrieve properties with balconies, sorted by the number of bedrooms in descending order: SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != "" ORDER BY Beds DESC LIMIT 0, 2000 1676 row(s) returned 0.016 sec / 0.000 sec
- 25 17:32:07 SELECT * FROM table_1 WHERE Balcony IS NOT NULL AND Balcony != "" ORDER BY Beds DESC LIMIT 0, 2000 1676 row(s) returned 0.016 sec / 0.000 sec
- 26 17:35:31 SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC LIMIT 0, 2000 5 row(s) returned 0.000 sec
- 27 17:36:06 SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC LIMIT 0, 2000 5 row(s) returned 0.016 sec / 0.000 sec
- 28 17:38:36 SELECT City, COUNT(*) AS Property_Count FROM table_1 GROUP BY City LIMIT 0, 2000 11 row(s) returned 0.016 sec / 0.000 sec
- 29 17:41:18 SELECT * FROM table_1 WHERE Beds >= 3 AND Bathroom >= 2 LIMIT 0, 2000 1601 row(s) returned 0.000 sec / 0.000 sec

5- Find properties in a specific state with a certain landmark. (take state and landmark on your own)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

classroom

SQL File 2

Limit to 2000 rows

28

-- 5. Find properties in a specific state with a certain landmark --

30

31 • SELECT *

32 FROM table_1

33 WHERE State_Code = 'HR' AND Landmarks LIKE '%Crown Interiorz Mall%';

34

35

Result Grid

Sno	BHK_Type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
2844	3	Standalone Building	3	2		Mathura Road N, 12/7, NH 44, Near Toll plaza, ...	Faridabad	HR	India	Crown Interiorz Mall
2845	4	Independent House	4	2		Independent House, Pocket C near Post office ...	Faridabad	HR	India	Crown Interiorz Mall
2846	3	Independent House	3	3		Independent House, Main Market Road near Vib...	Faridabad	HR	India	Crown Interiorz Mall
2847	2	Independent House	2	2	2.0	Independent House, Vasundra pocket	Faridabad	HR	India	Crown Interiorz Mall
2848	3	Standalone Building	3	2		NH-2, Mathura Road, Near Badarpur near safro...	Faridabad	HR	India	Crown Interiorz Mall
2849	2	Standalone Building	2	3		Standalone Building, Near Espire Hamilton Tower	Faridabad	HR	India	Crown Interiorz Mall
2850	2	Independent House	2	2	1.0	Independent House, Opposite SIGMA Classes	Faridabad	HR	India	Crown Interiorz Mall
2851	4	Independent House	4	3		Independent House, mathura road, Near sarai ...	Faridabad	HR	India	Crown Interiorz Mall
2853	4	Apartment	4	4		Sarai near Sarai metro station	Faridabad	HR	India	Crown Interiorz Mall
2854	4	Independent House	4	7		Independent House, Khanid near Lingaya's Vid...	Faridabad	HR	India	Crown Interiorz Mall
2855	4	Independent House	4	4	2.0	Independent House, Mohan Cooperative Indust...	Faridabad	HR	India	Crown Interiorz Mall

table_1 7 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
27	17:36:06	SELECT City, AVG(Beds) AS Average_Beds FROM table_1 GROUP BY City ORDER BY Average_Beds DESC	5 row(s) returned	0.016 sec / 0.000 sec
28	17:38:36	SELECT City, COUNT(*) AS Property_Count FROM table_1 GROUP BY City LIMIT 0, 2000	11 row(s) returned	0.016 sec / 0.000 sec
29	17:41:18	SELECT * FROM table_1 WHERE Beds >= 3 AND Bathroom >= 2 LIMIT 0, 2000	1601 row(s) returned	0.000 sec / 0.000 sec
30	17:45:25	SELECT * FROM Table1 WHERE State_Code = 'IL' AND Landmarks LIKE '%Lincoln Park%'; LIMIT 0, 2000	Error Code: 1146. Table 'table1_capstane.table1' doesn't exist	0.000 sec
31	17:46:03	SELECT * FROM table_1 WHERE State_Code = 'IL' AND Landmarks LIKE '%Lincoln Park%'; LIMIT 0, 2000	0 row(s) returned	0.000 sec / 0.000 sec
32	17:48:40	SELECT * FROM table_1 WHERE State_Code = 'HR' AND Landmarks LIKE '%Crown Interiorz Mall%'; LIMIT 0, 2000	186 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

Schema: table1_capstane

Administration Schemas Information

Search Grid Form Editor Field Types

Read Only

Windows Taskbar

Search

20-06-2024 17:49

Table2

Successfully imported table_2.csv to my sql server.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'college' expanded, showing 'table1_capstane' and 'table2'. The main editor window shows a SQL script with the following queries:

```
USE table1_capstane;
select* from table_2;
```

The 'Result Grid' displays the data from 'table_2' with columns: Sno, Carpet_area, Status, Floor, Transaction_Type, Year_Built, Price_per_square_feet. The data includes rows for properties built in 1964, 1995, 2006, and 2000.

The 'Output' pane shows the execution log with the following messages:

- 34 17:53:35 SHOW DATABASES OK 0.000 sec
- 35 17:53:37 SHOW SESSION VARIABLES LIKE 'lower_case_table_names' OK 0.000 sec
- 36 17:53:42 CREATE TABLE table1_capstane.table_2 (Sno int, Carpet_area text, Status text, Floor text, Transaction_Type text, Year_Built text, Price_per_square_feet text) OK 0.000 sec
- 37 17:53:42 PREPARE stmt FROM 'INSERT INTO table1_capstane.table_2 (Sno, Carpet_area, Status, Floor, Transaction_Type, Year_Built, Price_per_square_feet) VALUES (213, 1, 200, Active, Transaction_Type, 1964.0, 333)' OK 0.000 sec
- 38 17:54:08 DEALLOCATE PREPARE stmt OK 0.000 sec
- 39 17:55:56 select* from table_2 LIMIT 0, 2000 2000 row(s) returned 0.000 sec / 0.000 sec

1- Calculate the average price per square foot for properties built before 2010.

The screenshot shows the MySQL Workbench interface. The main editor window shows a SQL script with the following queries:

```
select* from table_2;

-- 1. Calculate the average price per square foot for properties built before 2010
SELECT AVG(Price_per_square_feet) AS Average_Price_Per_Square_Foot
FROM table_2
WHERE Year_Built < 2010;
```

The 'Result Grid' displays the result of the query, showing the average price per square foot for properties built before 2010, which is 10.879662401306833.

The 'Output' pane shows the execution log with the following messages:

- 35 17:53:37 SHOW SESSION VARIABLES LIKE 'lower_case_table_names' OK 0.000 sec
- 36 17:53:42 CREATE TABLE table1_capstane.table_2 (Sno int, Carpet_area text, Status text, Floor text, Transaction_Type text, Year_Built text, Price_per_square_feet text) OK 0.000 sec
- 37 17:53:42 PREPARE stmt FROM 'INSERT INTO table1_capstane.table_2 (Sno, Carpet_area, Status, Floor, Transaction_Type, Year_Built, Price_per_square_feet) VALUES (213, 1, 200, Active, Transaction_Type, 1964.0, 333)' OK 0.000 sec
- 38 17:54:08 DEALLOCATE PREPARE stmt OK 0.000 sec
- 39 17:55:56 select* from table_2 LIMIT 0, 2000 2000 row(s) returned 0.000 sec / 0.000 sec
- 40 17:58:20 SELECT AVG(Price_per_square_feet) AS Average_Price_Per_Square_Foot FROM table_2 WHERE Year_Built < 2010; 1 row(s) returned 0.016 sec / 0.000 sec

2- Find the total number of properties on each floor.

MySQL Workbench interface showing a query to find the total number of properties on each floor. The query is executed against the 'table1_capstane' schema, specifically against 'table_2'.

```
-- 2. Find the total number of properties on each floor--
SELECT Floor, COUNT(*) AS Property_Count
FROM table_2
GROUP BY Floor;
```

Floor	Property_Count
5/5	23
2/4	30
0/0	35
1/3	35
4/4	35
2/2	60
1/5	12
0/3	32
0/4	19
3/3	46
0/1	25

Result 11 x

Output

#	Time	Action	Message	Duration / Fetch
38	17:54:08	DEALLOCATE PREPARE stmt	OK	0.000 sec
39	17:55:56	select* from table_2 LIMIT 0, 2000	2000 row(s) returned	0.000 sec / 0.000 sec
40	17:58:20	SELECT AVG(Price_per_square_feet) AS Average_Price_Per_Square_Foot FROM table_2 WHERE Year_Built...	1 row(s) returned	0.016 sec / 0.000 sec
41	18:00:27	select* from table_2 LIMIT 0, 2000	2000 row(s) returned	0.000 sec / 0.000 sec
42	18:00:45	SELECT Floor, COUNT(*) AS Property_Count FROM Table2 GROUP BY Floor LIMIT 0, 2000	Error Code: 1146. Table 'table1_capstane.table2' doesn't exist	0.000 sec
43	18:01:11	SELECT Floor, COUNT(*) AS Property_Count FROM table_2 GROUP BY Floor LIMIT 0, 2000	144 row(s) returned	0.016 sec / 0.000 sec

3- Retrieve properties with a carpet area greater than 1000 square feet and a status of 'Under Construction'.

MySQL Workbench interface showing a query to retrieve properties with a carpet area greater than 1000 square feet and a status of 'Under Construction'. The query is executed against the 'table1_capstane' schema, specifically against 'table_2'.

```
-- 3. Retrieve properties with a carpet area greater than 1000 square feet and a status of 'Under Construction':
SELECT *
FROM table_2
WHERE Carpet_area > 1000 AND Status = 'Under Construction';
```

Sno	Carpet_area	Status	Floor	Transaction_Type	Year_Built	Price_per_square_feet
1	1000	Under Construction	5/5	1	2020	1000
2	1000	Under Construction	2/4	1	2020	1000
3	1000	Under Construction	0/0	1	2020	1000
4	1000	Under Construction	1/3	1	2020	1000
5	1000	Under Construction	4/4	1	2020	1000
6	1000	Under Construction	2/2	1	2020	1000
7	1000	Under Construction	1/5	1	2020	1000
8	1000	Under Construction	0/3	1	2020	1000
9	1000	Under Construction	0/4	1	2020	1000
10	1000	Under Construction	3/3	1	2020	1000
11	1000	Under Construction	0/1	1	2020	1000
12	1000	Under Construction	5/5	1	2020	1000
13	1000	Under Construction	2/4	1	2020	1000
14	1000	Under Construction	0/0	1	2020	1000

Table_2 14 x

Output

#	Time	Action	Message	Duration / Fetch
44	18:03:57	SELECT * FROM Table2 WHERE CAST(REPLACE(Carpet_area, ',')) AS DECIMAL) > 1000 AND Status = 'U...	Error Code: 1146. Table 'table1_capstane.table2' doesn't exist	0.000 sec
45	18:04:16	SELECT * FROM table_2 WHERE CAST(REPLACE(Carpet_area, ',')) AS DECIMAL) > 1000 AND Status = 'U...	0 row(s) returned	0.031 sec / 0.000 sec
46	18:04:57	SELECT * FROM table_2 WHERE CAST(REPLACE(Carpet_area, ',')) AS DECIMAL) > 1000 AND Status = 'U...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser...	0.000 sec
47	18:06:13	SELECT * FROM table_2 WHERE Carpet_area > 1000 AND Status = 'Under Construction' LIMIT 0, 2000	0 row(s) returned	0.016 sec / 0.000 sec
48	18:07:09	SELECT * FROM Table2 WHERE CAST(REPLACE(Carpet_area, ',')) AS DECIMAL) > 1000 AND Status = 'U...	Error Code: 1146. Table 'table1_capstane.table2' doesn't exist	0.000 sec
49	18:07:26	SELECT * FROM Table_2 WHERE CAST(REPLACE(Carpet_area, ',')) AS DECIMAL) > 1000 AND Status = 'U...	0 row(s) returned	0.016 sec / 0.000 sec

4- Calculate the average price per square foot for each transaction type.

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

```
-- 4. Calculate the average price per square foot for each transaction type
SELECT Transaction_Type, AVG(Price_per_square_foot) AS Average_Price_Per_Square_Foot
FROM table_2
GROUP BY Transaction_Type;
```

The Result Grid shows the following data:

Transaction_Type	Average_Price_Per_Square_Foot
12.435619409053944	

The Action Output pane shows the execution log with the following messages:

- 45 18:04:16 SELECT * FROM table_2 WHERE CAST(REPLACE(Capet_area, ',', '')) AS DECIMAL) > 1000 AND Status = 'U...' 0 row(s) returned
- 46 18:04:57 SELECT * FROM table_2 WHERE Capet_area AS DECIMAL > 1000 AND Status = 'Under Construction' Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ser... 0.000 sec
- 47 18:06:13 SELECT * FROM table_2 WHERE Capet_area > 1000 AND Status = 'Under Construction' LIMIT 0, 2000 0 row(s) returned
- 48 18:07:09 SELECT * FROM table_2 WHERE CAST(REPLACE(Capet_area, ',', '')) AS DECIMAL) > 1000 AND Status = 'U...' Error Code: 1146. Table table1_capstane.table2 doesn't exist 0.000 sec
- 49 18:07:26 SELECT * FROM Table_2 WHERE CAST(REPLACE(Capet_area, ',', '')) AS DECIMAL) > 1000 AND Status = 'U...' 0 row(s) returned
- 50 18:09:33 SELECT Transaction_Type, AVG(Price_per_square_foot) AS Average_Price_Per_Square_Foot FROM table_2... 0.016 sec / 0.000 sec

5- Find the properties with the highest price per square foot, sorted in descending order

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

```
-- 5. Find the properties with the highest price per square foot, sorted in descending order
SELECT *
FROM Table_2
ORDER BY Price_per_square_foot DESC;
```

The Result Grid shows the following data:

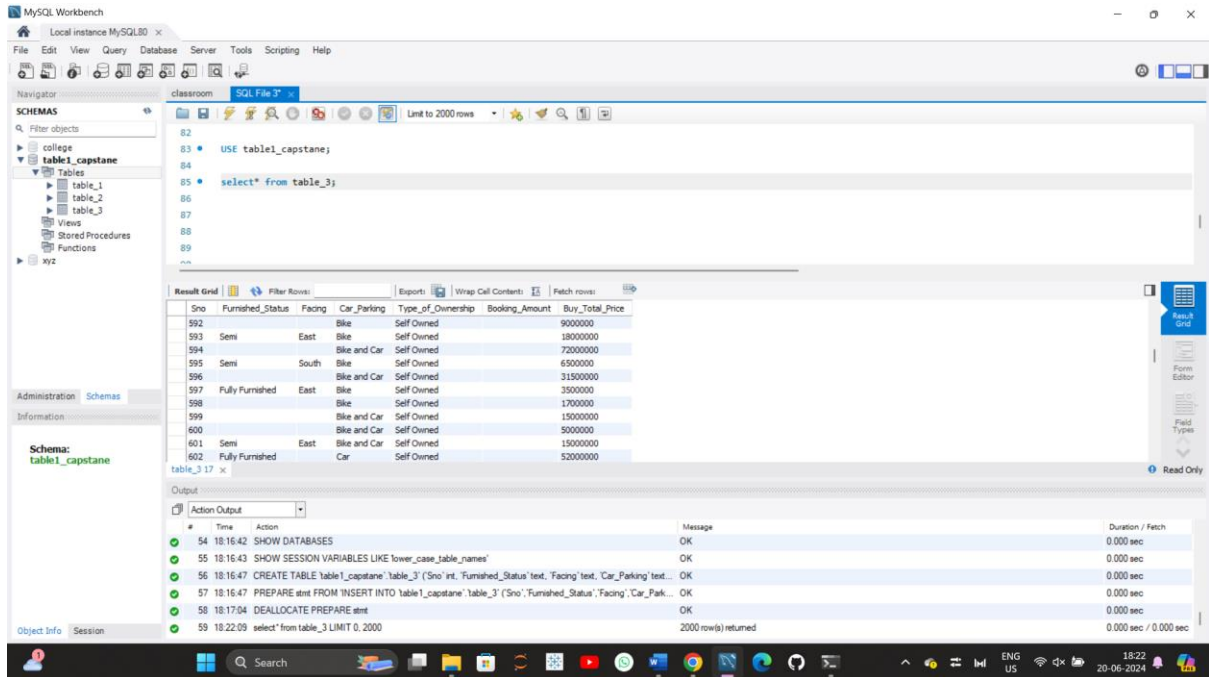
Sno	Capet_area	Status	Floor	Transaction_Type	Year_Built	Price_per_square_foot
92	3,249	Active			2021.0	922
206	3,249	Active			2021.0	922
201	3,434					859
87	3,434					859
73	3,157	Active			2023.0	768
187	3,157	Active			2023.0	768
35	2,580	Active			2013.0	523
149	2,580	Active			2013.0	523
219	3,900					513
106	3,900					513
105	5,705	Active			2000.0	508

The Action Output pane shows the execution log with the following messages:

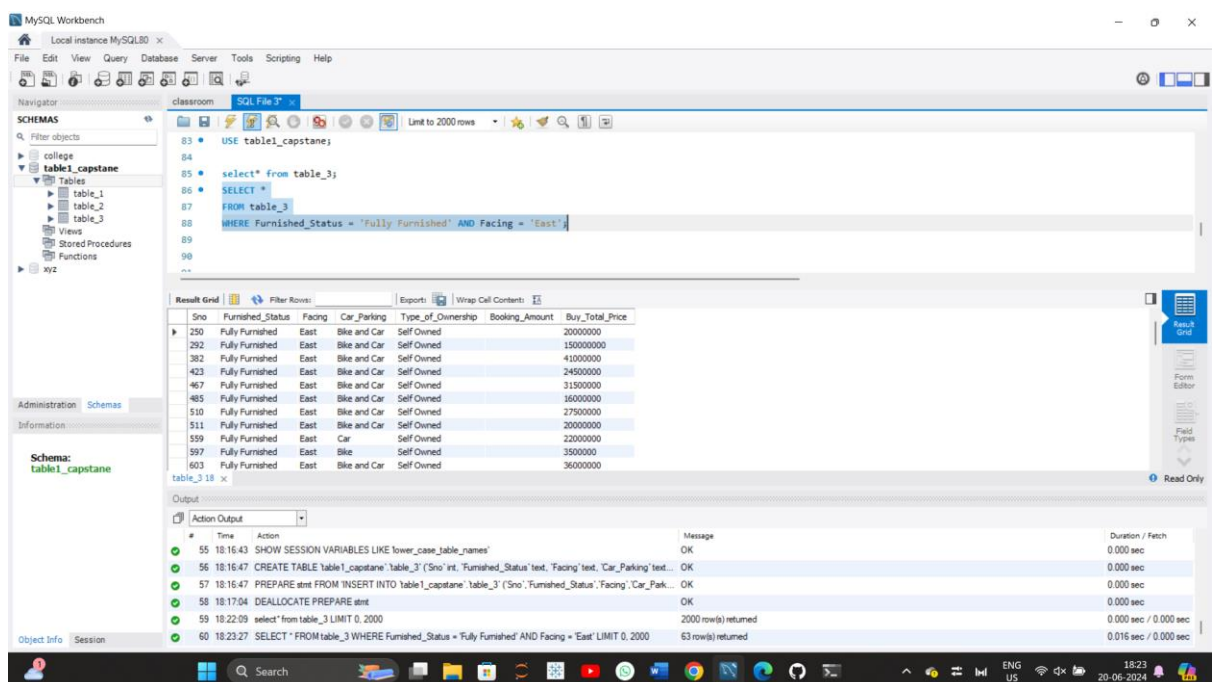
- 47 18:06:13 SELECT * FROM table_2 WHERE Capet_area > 1000 AND Status = 'Under Construction' LIMIT 0, 2000 0 row(s) returned
- 48 18:07:09 SELECT * FROM Table2 WHERE CAST(REPLACE(Capet_area, ',', '')) AS DECIMAL) > 1000 AND Status = 'U...' Error Code: 1146. Table table1_capstane.table2 doesn't exist 0.000 sec
- 49 18:07:26 SELECT * FROM Table_2 WHERE CAST(REPLACE(Capet_area, ',', '')) AS DECIMAL) > 1000 AND Status = 'U...' 0 row(s) returned
- 50 18:09:33 SELECT Transaction_Type, AVG(Price_per_square_foot) AS Average_Price_Per_Square_Foot FROM table_2... 0.016 sec / 0.000 sec
- 51 18:11:53 SELECT * FROM Table2 ORDER BY Price_per_square_foot DESC LIMIT 0, 2000 Error Code: 1146. Table table1_capstane.table2 doesn't exist 0.000 sec
- 52 18:12:43 SELECT * FROM Table_2 ORDER BY Price_per_square_foot DESC LIMIT 0, 2000 2000 row(s) returned 0.016 sec / 0.000 sec

Table3

Successfully imported table_3.csv to my sql server.



- 1- Retrieve all properties with a furnished status of 'Fully Furnished' and a facing direction of 'East'.



2- Calculate the average booking amount for properties with and without car parking:

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

```
-- 2. Calculate the average booking amount for properties with and without car parking.
SELECT Car_Parking,
       AVG(Booking_Amount) AS Average_Booking_Amount
FROM table_3
GROUP BY Car_Parking;
```

The Results grid shows the following data:

Car_Parking	Average_Booking_Amount
0	0
1	0
2	0
3	0

The Output tab shows the execution log with the following messages:

```
57 18:16:47 PREPARE stmt FROM 'INSERT INTO `table1_capstane`.`table_3` (`Sno`,`Furnished_Status`,`Facing`,`Car_Park...` OK
58 18:17:04 DEALLOCATE PREPARE stmt
59 18:22:09 select* from table_3 LIMIT 0. 2000 2000 row(s) returned
60 18:23:27 SELECT* FROM table_3 WHERE Furnished_Status = 'Fully Furnished' AND Facing = 'East' LIMIT 0. 2000 63 row(s) returned
61 18:24:53 SELECT Car_Parking, AVG(Booking_Amount) AS Average_Booking_Amount FROM table_3 GROUP BY Ca... 4 row(s) returned
62 18:25:13 SELECT Car_Parking, AVG(Booking_Amount) AS Average_Booking_Amount FROM table_3 GROUP BY Car_P... 4 row(s) returned
```

3- Find the total price of properties with different types of ownership.

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

```
-- 3. Find the total price of properties with different types of ownership.
SELECT Type_of_Ownership,
       SUM(Buy_Total_Price) AS Total_Price
FROM table_3
GROUP BY Type_of_Ownership;
```

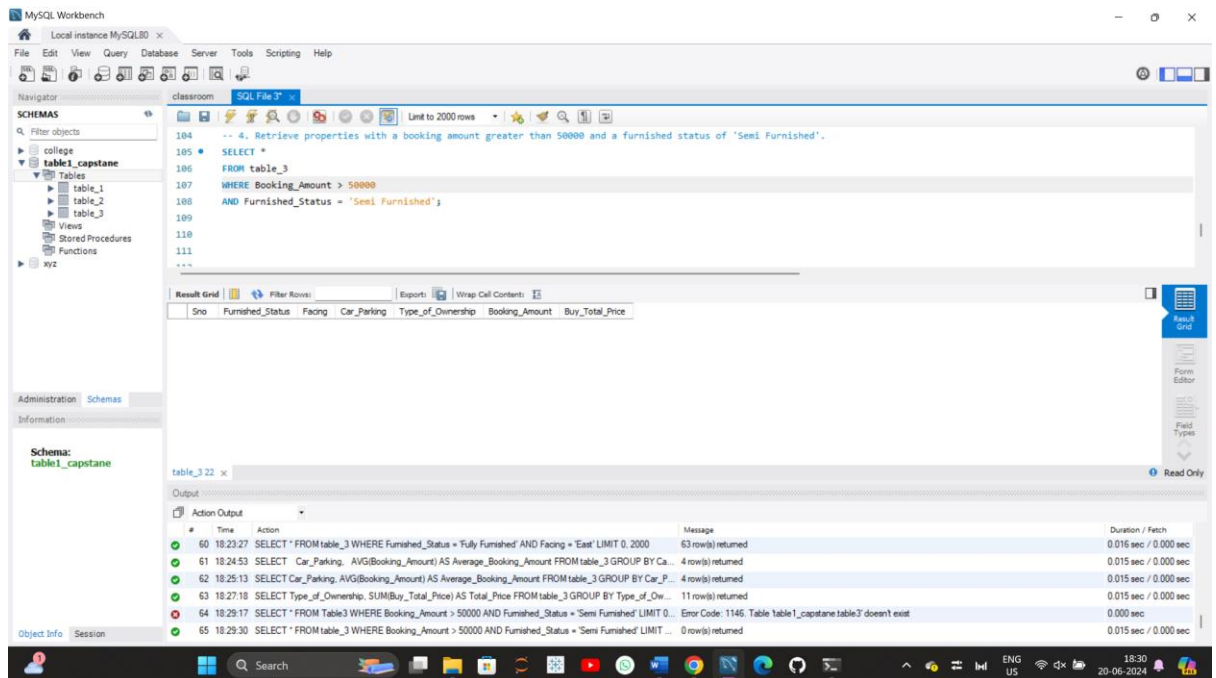
The Results grid shows the following data:

Type_of_Ownership	Total_Price
Self Owned	59524181000
On Lease	46200000
On Lease 10 Years	7000000
On Lease 9 Years	20000000
On Lease 99 Years	30200000
On Lease 999 Years	25000000
On Lease 120 Years	20000000
On Lease 50 Years	5900000
On Lease 3 Years	3800000
On Lease 70 Years	24000000

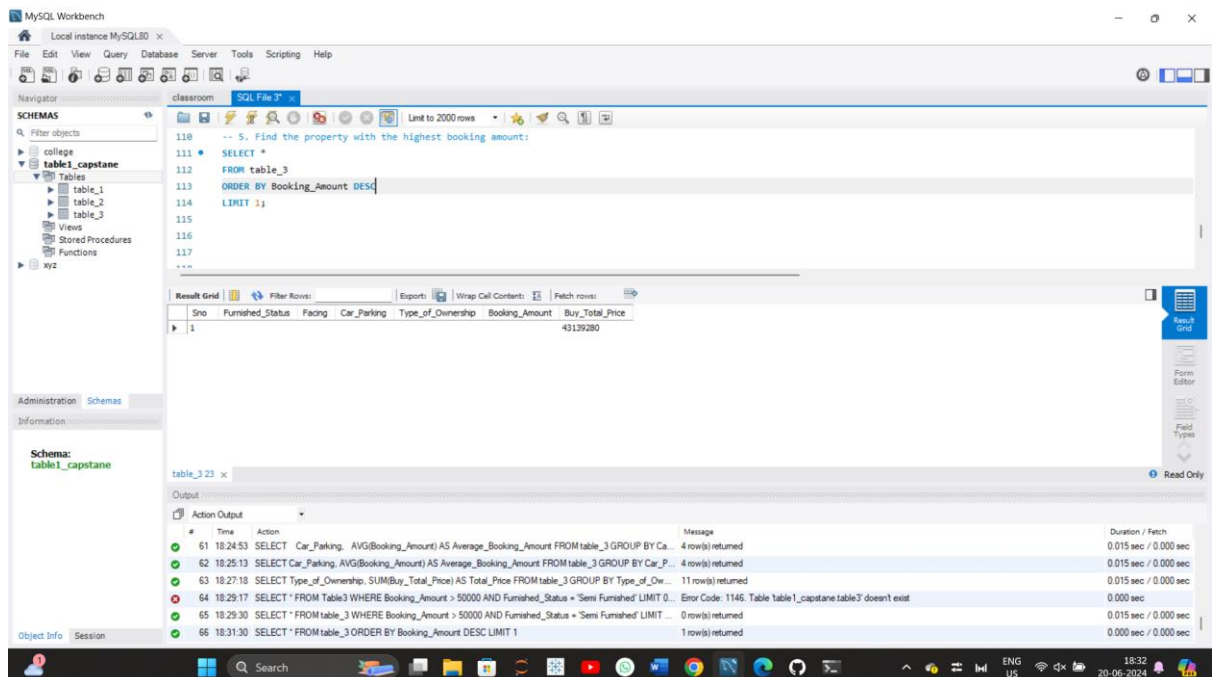
The Output tab shows the execution log with the following messages:

```
58 18:17:04 DEALLOCATE PREPARE stmt
59 18:22:09 select* from table_3 LIMIT 0. 2000 2000 row(s) returned
60 18:23:27 SELECT* FROM table_3 WHERE Furnished_Status = 'Fully Furnished' AND Facing = 'East' LIMIT 0. 2000 63 row(s) returned
61 18:24:53 SELECT Car_Parking, AVG(Booking_Amount) AS Average_Booking_Amount FROM table_3 GROUP BY Ca... 4 row(s) returned
62 18:25:13 SELECT Car_Parking, AVG(Booking_Amount) AS Average_Booking_Amount FROM table_3 GROUP BY Car_P... 4 row(s) returned
63 18:27:18 SELECT Type_of_Ownership, SUM(Buy_Total_Price) AS Total_Price FROM table_3 GROUP BY Type_of_Ow... 11 row(s) returned
```

- 4- Retrieve properties with a booking amount greater than 50000 and a furnished status of 'Semi Furnished'.



- 5- Find the property with the highest booking amount.



7 Join SQL Queries using all 3 tables

- 1- Retrieve properties from table1 that have a higher price per square foot than the average price per square foot in table2.

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

```
SELECT t1.*
FROM table_1 t1
JOIN table_2 t2 ON t1.Sno = t2.Sno
WHERE t2.Price_per_square_foot > (SELECT AVG(Price_per_square_foot) FROM table_2);
```

The query results are displayed in a table with the following columns: Sno, BHK_type, Property_Type, Beds, Bathroom, Balcony, Address, City, State_Code, Country, and Landmarks. The results show 13 rows of property data.

Sno	BHK_type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
1	3	Condo	3	2	Balcony	2243 North Lister Avenue #402	Chicago	IL 60614	United States	CHI - Logan Square
3	2	Condo	2	2		937 West Wrightwood Avenue #A	Chicago	IL 60614	United States	CHI - Lincoln Park
4	4	Condo	4	4		1438 North North Park Avenue #1	Chicago	IL 60610	United States	Cook
6	3	Condo	3	3.5	Balcony	352 West Huron Street #8	Chicago	IL 60654	United States	Cook
9	2	Condo	2	2		33 West Ontario Street #4IES	Chicago	IL 60654	United States	CHI - Near North Side
11	5	Condo	5	3		2230 North Kenmore Avenue #1	Chicago	IL 60614	United States	CHI - Lincoln Park
13	4	Condo	4	4		1460 North Sandburg Terrace #2001-2002	Chicago	IL 60610	United States	CHI - Near North Side
14	2	Condo	2	2		525 West Superior Street #623	Chicago	IL 60654	United States	CHI - Near North Side
15	3	Condo	3	2		3708 West Belmont Avenue #2	Chicago	IL 60618	United States	CHI - Avondale
16	3	Condo	3	2		3708 West Belmont Avenue #3	Chicago	IL 60618	United States	CHI - Avondale
18	3	Condo	3	3		619 North Noble Street #1	Chicago	IL 60642	United States	CHI - West Town

- 2- Find the properties in table1 that are located in cities where the average price per square foot in table2 is higher than the overall average price per square foot.

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

```
GROUP BY t1.City
HAVING AVG(t2.Price_per_square_foot) > (SELECT AVG(Price_per_square_foot) FROM table_2)
SELECT t1.*
FROM table_1 t1
JOIN city_avg_price cap ON t1.City = cap.City;
```

The query results are displayed in a table with the following columns: Sno, BHK_type, Property_Type, Beds, Bathroom, Balcony, Address, City, State_Code, Country, and Landmarks. The results show 10 rows of property data.

Sno	BHK_type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
220	4	Condo	4	3.5		700 North Larrabee Street #1115	Chicago	IL 60654	United States	Cook
219	4	Condo	4	3.5		33 West Ontario Street #4C	Chicago	IL 60654	United States	Cook
218	5	Condo	5	5		445 East North Water Street #E2303-E2305	Chicago	IL 60611	United States	CHI - Near North Side
217	2	Condo	2	1.5		444 West Fullerton Parkway #804	Chicago	IL 60614	United States	Cook
216	3	Condo	3	2	Balcony	828 North California Avenue #2	Chicago	IL 60622	United States	CHI - West Town
215	4	Homes (Single Family)	4	5.5		2036 North MAGNOLIA Avenue	Chicago	IL 60614	United States	Cook
213	2	Condo	2	1		758 North LARRABEE Street #327	Chicago	IL 60654	United States	CHI - Near North Side
212	2	Condo	2	2		937 West Wrightwood Avenue #A	Chicago	IL 60614	United States	CHI - Lincoln Park
211	6	Homes (Single Family)	6	4		3027 North Allen Avenue	Chicago	IL 60618	United States	CHI - Avondale
210	3	Condo	3	2	Balcony	2442 North Clybourn Avenue #9N	Chicago	IL 60614	United States	CHI - Lincoln Park

- 3- Retrieve properties from table1 with a certain landmark that have a lower price per square foot than the average price per square foot for properties with the same landmark in table2. (Choose landmark on our own)

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

```
-- 3. Retrieve properties from Table1 with a certain landmark that have a lower price per square foot than the average price per square foot for
-- properties with the same landmark in Table2
141 SELECT t1.*
142 FROM table_1 t1
143 JOIN table_2 t2 ON t1.Sno = t2.Sno
144 WHERE t1.Landmarks LIKE 'Lincoln Park'
145 AND t2.Price_per_square_feet < (SELECT AVG(Price_per_square_feet) FROM table_2 WHERE Landmarks LIKE 'Lincoln Park')
146
```

The Result Grid shows the following data:

Sno	BHK_type	Property_Type	Beds	Bathroom	Balcony	Address	City	State_Code	Country	Landmarks
50	4	Homes (Single Family)	4	3		2705 North Marshfield Avenue	Chicago	IL 60614	United States	CHI - Lincoln Park
164	4	Homes (Single Family)	4	3		2705 North Marshfield Avenue	Chicago	IL 60614	United States	CHI - Lincoln Park

The Output pane shows the execution log with the following messages:

```
73 18:44:49 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
74 18:47:02 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
75 18:47:44 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
76 18:48:55 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
77 18:53:08 SELECT t1.* FROM table_1 JOIN table_2 ON t1.Sno = t2.Sno WHERE t1.Landmarks LIKE 'Lincoln Park... 2 row(s) returned 0.078 sec / 0.000 sec
78 18:56:11 SELECT t1.* FROM table_1 JOIN table_2 ON t1.Sno = t2.Sno WHERE t1.Landmarks LIKE 'Lincoln Park... 2 row(s) returned 0.094 sec / 0.000 sec
```

- 4- Retrieve properties from table2 with a price per square foot higher than the average booking amount in table3.

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

```
-- 4. Retrieve properties from Table2 with a price per square foot higher than the average booking amount in Table3:
148 SELECT t2.*
149 FROM table_2 t2
150 WHERE t2.Price_per_square_feet > (SELECT AVG(Booking_Amount) FROM table_3);
151
152
153
154
155
```

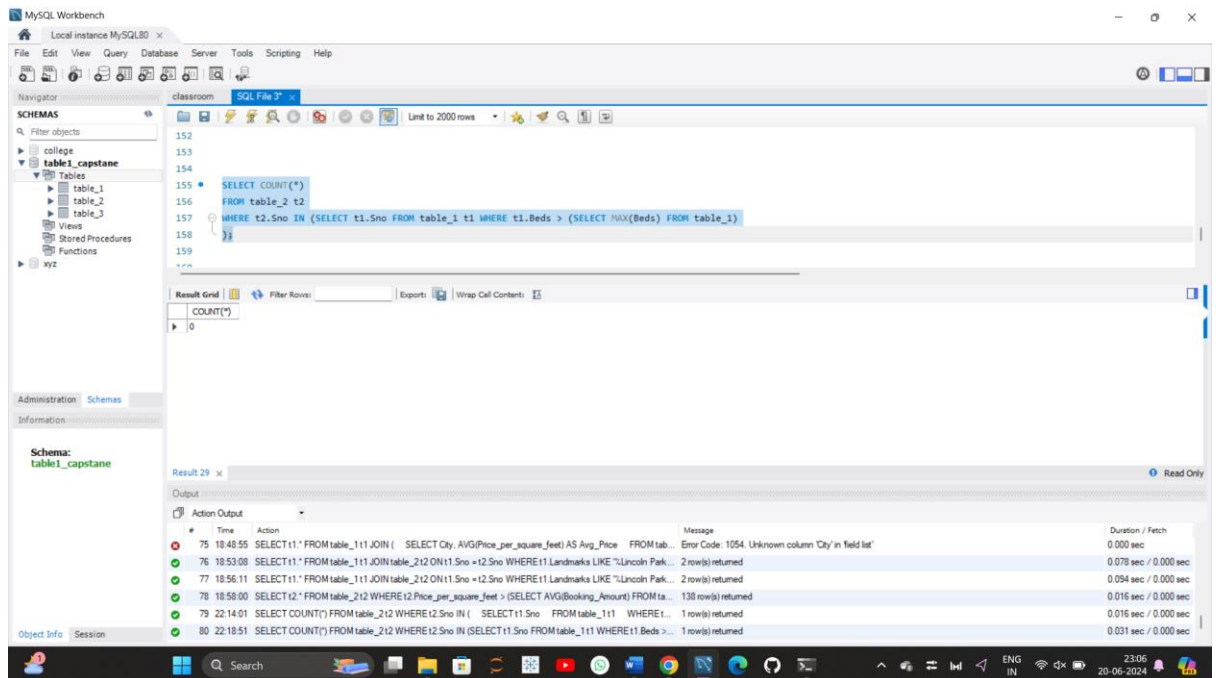
The Result Grid shows the following data:

Sno	Carpet_area	Status	Floor	Transaction_Type	Year_Built	Price_per_square_feet
1	1,500	Active			2004.0	346
3	1,300	Active			1988.0	296
4	4,000					350
6	3,800					303
9	1,350	Active			2003.0	326
11	3,174	Active				246
13	2,400	Active			1962.0	271
14	1,300	Active				377
15	1,575					349
16	1,800					361
18	2,111	Active			1890.0	213

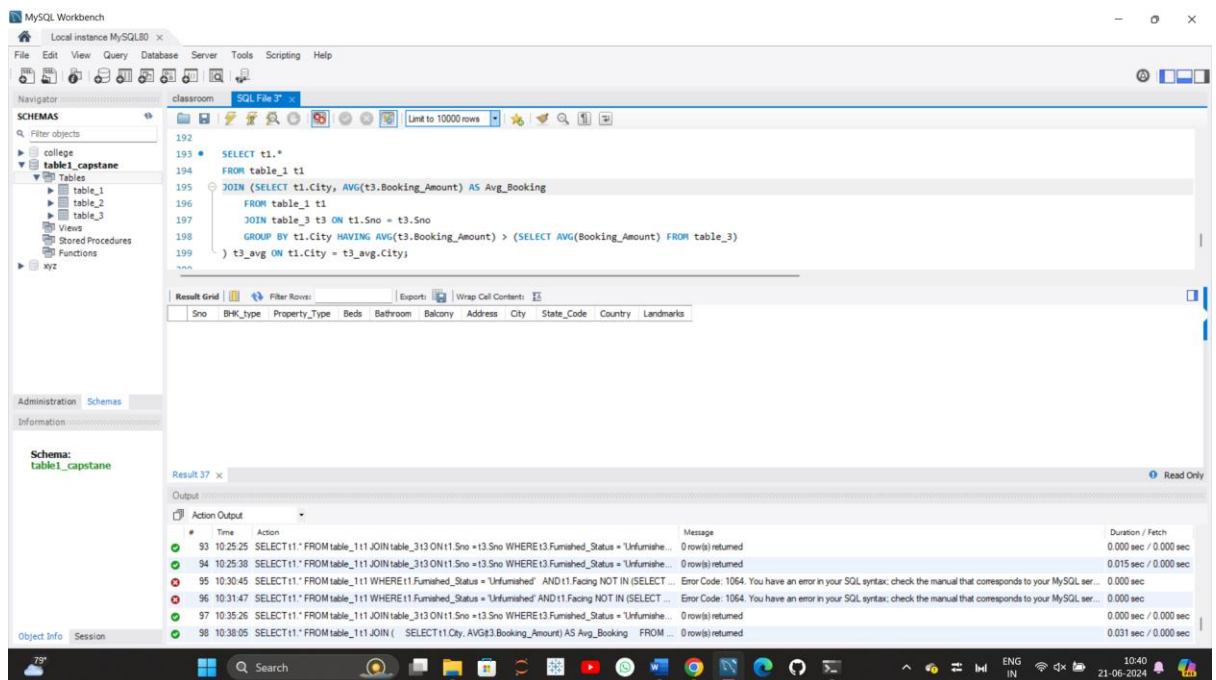
The Output pane shows the execution log with the following messages:

```
73 18:47:02 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
74 18:47:44 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
75 18:48:55 SELECT t1.* FROM table_1 JOIN ( SELECT City, AVG(Price_per_square_feet) AS Avg_Price FROM tab... Error Code: 1054. Unknown column 'City' in field list 0.000 sec
76 18:53:08 SELECT t1.* FROM table_1 JOIN table_2 ON t1.Sno = t2.Sno WHERE t1.Landmarks LIKE 'Lincoln Park... 2 row(s) returned 0.078 sec / 0.000 sec
77 18:56:11 SELECT t1.* FROM table_1 JOIN table_2 ON t1.Sno = t2.Sno WHERE t1.Landmarks LIKE 'Lincoln Park... 2 row(s) returned 0.094 sec / 0.000 sec
78 18:58:00 SELECT t2.* FROM table_2 WHERE t2.Price_per_square_feet > (SELECT AVG(Booking_Amount) FROM ta... 138 row(s) returned 0.016 sec / 0.000 sec
```

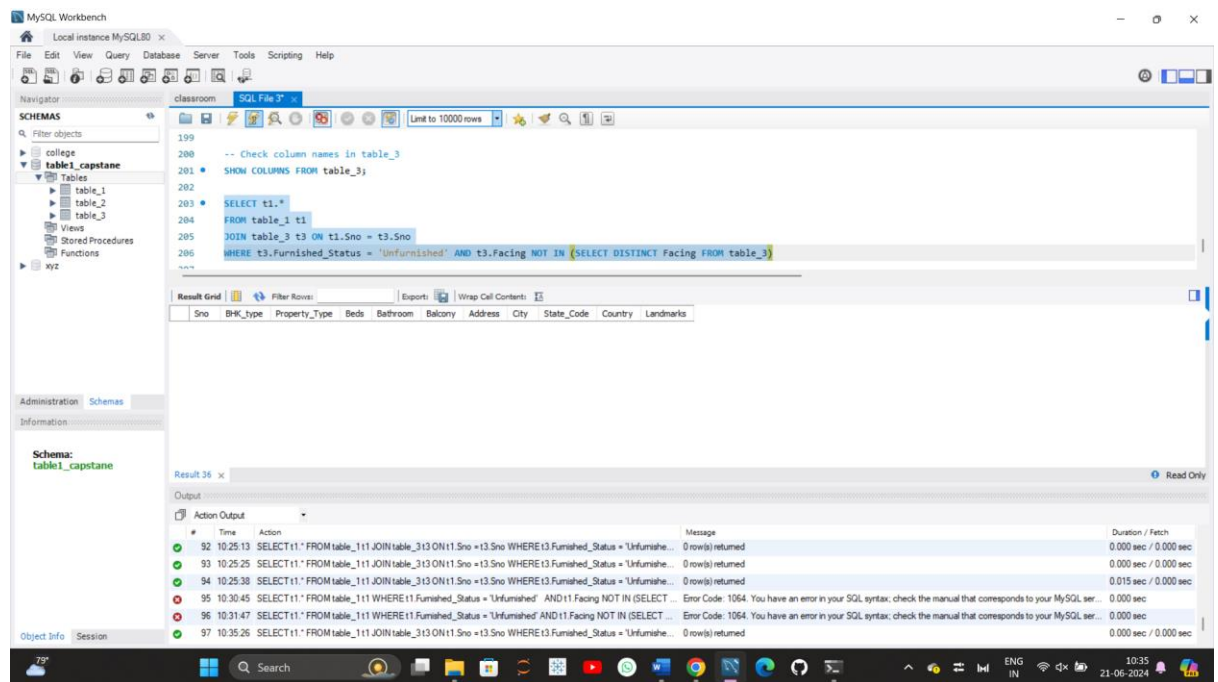

- 5- Count the number of properties in table2 with more bedrooms than the maximum number of bedrooms in table3.



- 6- Find the cities where the average booking amount in table3 is higher than the overall average booking amount, and retrieve properties from table1 located in those cities.



- 7- Retrieve properties from table1 with a furnished status of 'Unfurnished' and a facing direction that does not exist in table3.



8-