

Buy Homes/Properties - REAL ESTATE CAPSTONE PROJECT

Capstone Project

Phase – 2:-

➤ Write the SQL queries :-

➤ **Table1**

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

```
1 • select * from table_1
2   where Number_Of_Balcony = True
3   order by Number_Of_Beds desc ;
4
```

The screenshot shows a database interface with a SQL query editor at the top and a 'Result Grid' below it. The query is: `select * from table_1 where Number_Of_Balcony = True order by Number_Of_Beds desc ;`. The 'Result Grid' displays a table with 10 columns: Sr_No, BHK_Type, Property_Type, Number_Of_Beds, Number_of_Bathroom, Number_Of_Balcony, Address, City, State Code, and Country. The results are filtered to show only properties with balconies, sorted by the number of bedrooms in descending order. The first few rows are visible, showing properties in Chicago and Dearborn, MI.

Sr_No	BHK_Type	Property_Type	Number_Of_Beds	Number_of_Bathroom	Number_Of_Balcony	Address	City	State Code	Country
8	1	Condo	1	2	1	653 North Kingsbury Street #2402, Chicago, IL ...	Chicago		USA
10	1	Condo	1	1	1	1528 South Wabash Avenue #303, Chicago, IL ...	Chicago		USA
23	1	Condo	1	1	1	The Regatta 420 E Waterside Dr Unit 714 Chica...	Chicago		USA
24	1	Condo	1	1	1	The Regatta 420 E Waterside Dr Unit 714 Chica...	Chicago		USA
25	1	Condo	1	1	1	The Regatta 420 E Waterside Dr Unit 714 Chica...	Chicago		USA
26	1	Condo	1	1	1	The Regatta 420 E Waterside Dr Unit 714 Chica...	Chicago		USA
31	1	Condo	1	1	1	Dearborn Park High Rise 1899 S Plymouth Ct U...	Chicago		USA
32	1	Condo	1	1	1	Dearborn Park High Rise 1899 S Plymouth Ct U...	Chicago		USA
33	1	Condo	1	1	1	Dearborn Park High Rise 1899 S Plymouth Ct U...	Chicago		USA

Below the 'Result Grid', there is an 'Output' section showing the execution details of the query. It indicates that the query was executed at 18:32:39 and returned 876 rows.

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

```
1 • select City, avg(Number_Of_Beds) as avg_bedroom
2   from table_1
3   group by City
4   order by avg_bedroom desc limit 5 ;
5
```

The screenshot shows a database interface with a SQL query editor at the top and a 'Result Grid' below it. The query is: `select City, avg(Number_Of_Beds) as avg_bedroom from table_1 group by City order by avg_bedroom desc limit 5 ;`. The 'Result Grid' displays a table with 2 columns: City and avg_bedroom. The results show the top 5 cities with the highest average number of bedrooms per property. The cities are Chicago, Bengaluru, Pune, Ahmedabad, and Delhi.

City	avg_bedroom
Chicago	3.5932
Bengaluru	2.7987
Pune	2.6250
Ahmedabad	2.6038
Delhi	2.5750

Below the 'Result Grid', there is an 'Output' section showing the execution details of the query. It indicates that the query was executed at 18:28:56 and returned 5 rows.

3. Count the number of properties in each city.

The screenshot shows a database query interface with a SQL editor at the top containing the following query:

```
1 • select City, count(*) as property_Count
2   from table_1
3  group by City
4  order by property_Count desc;
```

Below the editor is a 'Result Grid' showing the results of the query:

City	property_Count
New York	697
Los Angeles	683
Chicago	472
Chennai	381
Mumbai	238
Delhi	160
Ahmedabad	159
Bengaluru	159
Pune	120
Thane	120

On the right side, there is a sidebar with options: 'Automatically disabled manual current toggle', 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, an 'Output' section shows the execution log:

```
1 10:12:18 select City, count(*) as property_Count from table_1 group by City order by property_Count desc LIMIT 0, 50000 10 row(s) returned
```

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

The screenshot shows a database query interface with a SQL editor at the top containing the following query:

```
1 • select * from table_1
2   where Number_Of_Beds >= 3
3   And Number_Of_Bathroom >=2 ;
4
```

Below the editor is a 'Result Grid' showing the results of the query:

Sr_No	BHK_Type	Property_Type	Number_Of_Beds	Number_of_Bathroom	Number Of Balcony	Address	City	State Code	Country
1	4	Condo	4	4	4	305 South Racine Avenue #PHD, Chicago, IL 6...	Chicago		USA
2	4	Condo	4	4	4	305 South Racine Avenue #PHD, Chicago, IL 6...	Chicago		USA
4	3	Condo	3	3	3	340 East Randolph Street #1705, Chicago, IL 6...	Chicago		USA
5	3	Condo	3	3	3	1211 South Prairie Avenue #706, Chicago, IL 6...	Chicago		USA
6	3	Condo	3	4	3	1201 South Prairie Avenue #1101, Chicago, IL ...	Chicago		USA
7	4	Condo	4	2	4	339 West Barry Avenue #38C, Chicago, IL 60657	Chicago		USA
9	3	Condo	3	4	3	21 East Huron Street #4204, Chicago, IL 60611	Chicago		USA
11	12	Condo	12	5	12	5330 S Wabash Ave Chicago, IL 60615 Washin...	Chicago		USA
12	12	Condo	12	5	12	5330 S Wabash Ave Chicago, IL 60615 Washin...	Chicago		USA

On the right side, there is a sidebar with options: 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, an 'Output' section shows the execution log:

```
1 18:25:18 select * from table_1 where Number_Of_Beds >= 3 And Number_Of_Bathroom >=2 1014 row(s) returned
```

5. Find properties in a specific City with a certain landmark. (Take City and landmark on your own).

SQL File 2" x table_1

```

1 • SELECT *FROM capstone_project.table_1
2 WHERE City = 'Delhi'
3 AND Landmark = '';

```

Result Grid

Sr.No	BHK Type	Property Type	Number Of Beds	Number Of Bathroom	Number Of Balcony	Address	City	State Code	Country	Landmark
3067	4	Apartments	4	2	4	Paschim Vihar, Delhi, Delhi	Delhi		INDIA	
3068	2	Apartments	2	1	2	Rohini Sector 4, Delhi, Delhi	Delhi		INDIA	
3069	3	Builder Floors	3	2	3	Ramesh Nagar, Delhi, Delhi	Delhi		INDIA	
3070	4	Apartments	4	4	4	Asha Park, Delhi, Delhi	Delhi		INDIA	
3071	3	Houses & Villas	3	3	3	Sultanpuri, Delhi, Delhi	Delhi		INDIA	
3072	2	Apartments	2	1	2	Pitampura, Delhi, Delhi	Delhi		INDIA	
3073	2	Builder Floors	2	2	2	Ramesh Nagar, Delhi, Delhi	Delhi		INDIA	
3074	2	Builder Floors	2	2	2	Ramesh Nagar, Delhi, Delhi	Delhi		INDIA	
3075	1	Apartments	1	1	1	Hastnoli, Delhi, Delhi	Delhi		INDIA	
3076	3	Builder Floors	3	2	3	Paschim Vihar, Delhi, Delhi	Delhi		INDIA	

table_1.44 x

Output

Action Output

#	Time	Action	Message
1	10:48:03	SELECT *FROM capstone_project.table_1 WHERE City = 'Delhi' AND Landmark = ''	160 row(s) returned

➤ Table2

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

1 • select avg(Price_Per_Sqaure_Feet) as avg_price_per_sq_ft

2 from table_2

3 where Year_Of_Built < 2010 ;

4

Result Grid

avg_price_per_sq_ft
0

Result 21 x

Output

Action Output

#	Time	Action	Message
1	18:20:57	select avg(Price_Per_Sqaure_Feet) as avg_price_per_sq_ft from table_2 where Year_Of_Built < 2010	1 row(s) returned

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

SQL File 2: table_1

```
1 SELECT Floor, count(*) as total_properties
2 FROM table_2
3 group by Floor
4 order by Floor;
```

Result Grid

Floor	total_properties
0	108
1	239
10	25
11	24
12	26
13	3
14	6
15	18
16	6
18	6

Output

Action Output

#	Time	Action	Message
1	11:22:24	SELECT Floor, count(*) as total_properties FROM table_2 group by Floor order by Floor	24 row(s) returned

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

SQL File 2: table_1

```
1 SELECT * FROM table_2
2 WHERE Carpet_Area >= 1000
3 AND Status = 'Under Construction';
```

Result Grid

Sr_No	Carpet_Area	Status	Floor	Transaction_Type	Year_Of_Built	Price_Per_Sqaure_Feet
2545	1000	UNDER CONSTRUCTION				
2549	1000	UNDER CONSTRUCTION				
2625	1000	UNDER CONSTRUCTION				
2629	1000	UNDER CONSTRUCTION				
2705	1000	UNDER CONSTRUCTION				
2709	1000	UNDER CONSTRUCTION				
2785	1000	UNDER CONSTRUCTION				
2789	1000	UNDER CONSTRUCTION				
2865	1000	UNDER CONSTRUCTION				
2869	1000	UNDER CONSTRUCTION				
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Output

Action Output

#	Time	Action	Message
1	11:49:51	SELECT * FROM table_2 WHERE Carpet_Area >= 1000 AND Status = 'Under Construction'	10 row(s) returned

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

```
1 • select Transaction_Type, avg('Price_Per_Sqaure_Feet') as average_price_per_sq_foot
2   from table_2
3  group by Transaction_Type;
```

Result Grid

Transaction_Type	average_price_per_sq_foot
0	

Result 77 x

Output

Action Output

#	Time	Action	Message
1	11:49:51	SELECT * FROM table_2 WHERE Carpet_Area >= 1000 AND Status ='Under Construction'	10 row(s) returned

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

```
1 • select *, Price_Per_Sqaure_Feet as price_per_sq_ft
2   from table_2
3  order by Price_Per_Sqaure_Feet desc;
4
```

Result Grid

Sr_No	Carpet_Area	Status	Floor	Transaction_Type	Year_Of_Built	Price_Per_Sqaure_Feet	price_per_sq_ft
756	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
914	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
757	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
755	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
546	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
545	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
544	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
543	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
754	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
917	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft
1074	1,340Sq	UNDER CONSTRUCTION			2011	\$962Price per Sq Ft	\$962Price per Sq Ft

table_2 79 x

Output

Action Output

#	Time	Action	Message
1	11:59:36	select *, Price_Per_Sqaure_Feet as price_per_sq_ft from table_2 order by Price_Per_Sqaure_Feet desc	3189 row(s) returned

➤ Table3

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

The screenshot shows a database query interface. The SQL query is as follows:

```
1 • select * from table_3
2   where Furnished_Status = 'Fully Furnished'
3   And Facing = 'East';
4
```

The results are displayed in a table with the following columns: Sr_No, Furnished_Status, Facing, Car_Parking, Type_Of_Ownership, Booking_Amount, and Buy_Total_Price. The table contains 14 rows of data.

Sr_No	Furnished_Status	Facing	Car_Parking	Type_Of_Ownership	Booking_Amount	Buy_Total_Price
1872	Furnished	East		BUILDER	530000	26500000
1875	Furnished	East	1	OWNER	440000	22000000
1877	Furnished	East	1	OWNER	280000	14000000
1952	Furnished	East		BUILDER	530000	26500000
1955	Furnished	East	1	OWNER	440000	22000000
1957	Furnished	East	1	OWNER	280000	14000000
2032	Furnished	East		BUILDER	530000	26500000
2035	Furnished	East	1	OWNER	440000	22000000
2037	Furnished	East	1	OWNER	280000	14000000
2141	Furnished	East		BUILDER	111960	5598000
2181	Furnished	East		BUILDER	111960	5598000

The output section shows the action output and message:

```
# Time Action Message
1 18:09:50 select * from table_3 where Furnished_Status = 'Fully Furnished' And Facing = 'East' 46 row(s) returned
```

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

The screenshot shows a database query interface. The SQL query is as follows:

```
1 • select Car_Parking, avg(Booking_Amount) as avg_booking_amount
2   from table_3
3   group by Car_Parking;
4
```

The results are displayed in a table with the following columns: Car_Parking and avg_booking_amount. The table contains 14 rows of data.

Car_Parking	avg_booking_amount
	1649100.0034
1 Car Attached Garage	697349.1667
1 Car Detached Garage	536290.3333
2 Car Attached Garage	871929.0000
1 Car Garage Electric Vehicle...	897530.0000
1 Parking Space Automatic G...	1304984.0000
2 Car Garage Tandem Parkin...	1328258.0000
1 Car Garage Gated Parking ...	1494498.0000
2 Car Garage Tandem Parkin...	1388104.0000
2 Car Garage Driveway Auto...	1328258.0000
1	229576.4517

The output section shows the action output and message:

```
# Time Action Message
1 18:13:08 select Car_Parking, avg(Booking_Amount) as avg_booking_amount from table_3 group by Car_Parking 14 row(s) returned
```

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

```
1 • select Type_Of_Ownership, sum(Buy_Total_Price) as total_price
2   from table_3
3  group by Type_Of_Ownership;
4
```

Type_Of_Ownership	total_price
BUILDER	17095921284
OWNER	16689044153
AGENT	162484729372

Result 8 x

Output

Action Output

#	Time	Action	Message
1	17:57:54	select Type_Of_Ownership, sum(Buy_Total_Price) as total_price from table_3 group by Type_Of_Ownership	3 row(s) returned

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

```
1 • select * from table_3
2   where Booking_Amount > 50000
3   and Furnished_Status = 'SemiFurnished' ;
4
```

Sr_No	Furnished_Status	Facing	Car_Parking	Type_Of_Ownership	Booking_Amount	Buy_Total_Price
1868	SemiFurnished	West	1	OWNER	260000	13000000
1898	SemiFurnished	East	1	OWNER	398000	19900000
1899	SemiFurnished	East	0	OWNER	190000	9500000
1900	SemiFurnished	East	1	OWNER	530000	26500000
1904	SemiFurnished	NorthEast	1	OWNER	380000	19000000
1909	SemiFurnished	East		BUILDER	62000	3100000
1915	SemiFurnished	SouthEast	3	OWNER	250000	12500000
1917	SemiFurnished	East	0	OWNER	198000	9900000
1932	SemiFurnished	East	0	OWNER	760000	38000000
1933	SemiFurnished	East		BUILDER	340000	17000000
1934	SemiFurnished	West	1	OWNER	250000	12500000

table_3 11 x

Output

Action Output

#	Time	Action	Message
1	18:01:51	select * from table_3 where Booking_Amount > 50000 and Furnished_Status = 'SemiFurnished'	369 row(s) returned

5- Find the property with the highest booking amount.

```
1 • select * from table_3
2 Order by Booking_Amount Desc limit 1;
3
```

The screenshot shows a database query tool interface. The SQL query is entered in the top editor. Below the editor, the 'Result Grid' is displayed, showing a single row of data for the property with the highest booking amount. The 'Output' pane at the bottom shows the execution details.

Sr_No	Furnished_Status	Facing	Car_Parking	Type_Of_Ownership	Booking_Amount	Buy_Total_Price
745	HALL	HALL	HALL	AGENT	7065200	353260000

table_3 13 x

Output

Action Output

#	Time	Action	Message
1	18:04:05	select * from table_3 Order by Booking_Amount Desc limit 1	1 row(s) returned

➤ 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

```
1 • SELECT t1.*
2 FROM table_1 t1
3 WHERE t1.Sr_No IN (
4     SELECT t1_inner.Sr_No
5     FROM table_1 t1_inner
6     INNER JOIN table_2 t2 ON t1_inner.Sr_No = t2.Sr_No
7     GROUP BY t1_inner.Sr_No
8     HAVING avg(t2.Price_Per_Sqaure_Feet) < (
9         SELECT AVG(Price_Per_Sqaure_Feet)
10        FROM table_2
11    )
12 );
```

The screenshot shows a database query tool interface. The SQL query is entered in the top editor. Below the editor, the 'Result Grid' is displayed, showing a single row of data for the property with the highest booking amount. The 'Output' pane at the bottom shows the execution details.

Sr_No	BHK_Type	Property_Type	Number_Of_Beds	Number_of_Bathroom	Number_Of_Balcony	Address	City	State Code	Country	Landmark
HALL	HALL	HALL	HALL	HALL	HALL	HALL	HALL	HALL	HALL	HALL

table_135 x

Output

Action Output

#	Time	Action	Message
1	15:22:49	SELECT t1.* FROM table_1 t1 WHERE t1.Sr_No IN (SELECT t1_inner.Sr_No FROM table_1 t1_inner ...	0 row(s) returned

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.

```

1 • SELECT t1.*
2   FROM table_1 t1
3  WHERE t1.City IN (
4      SELECT t2.Price_Per_Sqaure_Feet
5      FROM table_2 t2
6      GROUP BY t2.Sr_No
7      HAVING AVG(t2.Price_Per_Sqaure_Feet) > (
8          SELECT AVG(Price_Per_Sqaure_Feet)
9          FROM table_2
10     )
11 );

```

Result Grid

Sr_No	BHK_Type	Property_Type	Number_Of_Beds	Number_of_Bathroom	Number_Of_Balcony	Address	City	State Code	Country	Landmark
1	1BHK	Apartment	1	1	1	123 Main St	New York	NY	USA	Central Park

table_1 139 x

Output

Action Output

#	Time	Action	Message
1	15:28:44	SELECT t1.* FROM table_1 t1 WHERE t1.City IN (SELECT t2.Price_Per_Sqaure_Feet FROM table_2 t2...	0 row(s) returned

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)

```

1 • SELECT t1.Landmark, AVG(t2.Price_Per_Sqaure_Feet) AS avg_price_per_sq_ft
2   FROM table_1 t1
3  INNER JOIN table_2 t2 ON t1.Sr_No = t2.Sr_No
4  WHERE t1.Landmark = ''
5  AND t2.Price_Per_Sqaure_Feet > (
6      SELECT AVG(t2_inner.Price_Per_Sqaure_Feet)
7      FROM table_2 t2_inner
8      WHERE t2_inner.Sr_No = t1.Sr_No
9  )
10 GROUP BY t1.Landmark
11 ORDER BY t1.Landmark;

```

Result Grid

Landmark	avg_price_per_sq_ft
----------	---------------------

Result 3 x

Output

Action Output

#	Time	Action	Message
1	14:41:34	SELECT t1.Landmark, AVG(t2.Price_Per_Sqaure_Feet) AS avg_price_per_sq_ft FROM table_1 t1 INNER JOI...	0 row(s) returned

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

```
1 • SELECT *
2 FROM table_2
3 WHERE table_2.Price_Per_Sqaure_Feet > (
4     SELECT AVG(table_3.Booking_Amount)
5     FROM table_3
6 )
```

Result Grid

Sr_No	Carpet_Area	Status	Floor	Transaction_Type	Year_Of_Built	Price_Per_Sqaure_Feet
*	NULL	NULL	NULL	NULL	NULL	NULL

table_2 19 x

Output

Action Output

#	Time	Action	Message
1	14:45:38	SELECT * FROM table_2 WHERE table_2.Price_Per_Sqaure_Feet > (SELECT AVG(table_3.Booking_Amount) FROM table_3)	0 row(s) returned

5- Count the number of properties in table1 with more bedrooms than the maximum number of bedrooms in table1

```
1 • SELECT COUNT(*) AS Num_Property
2 FROM table_1 t1
3 WHERE t1.Number_Of_Beds > (
4     SELECT MAX(t1.Number_Of_Beds)
5     FROM table_1 t1
6 )
```

Result Grid

Num_Property
0

Result 24 x

Output

Action Output

#	Time	Action	Message
1	14:53:02	SELECT COUNT(*) AS Num_Property FROM table_1 t1 WHERE t1.Number_Of_Beds > (SELECT MAX(t1.Number_Of_Beds) FROM table_1 t1)	1 row(s) returned

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

The screenshot shows a database query editor with the following SQL query:

```

1 SELECT t1.*
2 FROM table_1 t1
3 WHERE t1.City IN (
4     SELECT t1.City
5     FROM table_3 t3
6     GROUP BY t1.City
7     HAVING AVG(t3.Booking_Amount) > (
8         SELECT AVG(Booking_Amount)
9         FROM table_3
10    )
11 )

```

The results grid is empty, showing all fields as NULL. The output pane shows the query execution details:

#	Time	Action	Message
1	15:07:58	SELECT t1.* FROM table_1 t1 WHERE t1.City IN (SELECT t1.City FROM table_3 t3 GROUP BY t1.C...	0 row(s) returned

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

The screenshot shows a database query editor with the following SQL query:

```

1 SELECT t3.*
2 FROM table_3 t3
3 WHERE t3.Furnished_Status = 'Unfurnished'
4 AND t3.Facing = 'NOT IN (
5     SELECT DISTINCT t3.Facing
6     FROM table_3 t3
7 )'

```

The results grid shows the following data:

Sr_No	Furnished_Status	Facing	Car_Parking	Type_Of_Ownership	Booking_Amount	Buy_Total_Price
1869	Unfurnished	East	0	AGENT	50000	2500000
1870	Unfurnished	East	1	AGENT	200000	10000000
1873	Unfurnished	SouthWest		BUILDER	113600	5680000
1874	Unfurnished	West	0	AGENT	262000	13100000
1876	Unfurnished	East	1	AGENT	188000	9400000

The output pane shows the query execution details:

#	Time	Action	Message
1	15:17:29	SELECT t3.* FROM table_3 t3 WHERE t3.Furnished_Status = 'Unfurnished' AND t3.Facing = 'NOT IN (S...	615 row(s) returned

----- Phase 2 End -----