

```
CREATE DATABASE Mysql_project;
```

```
use Mysql_project;
```

/\*1) Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

1) Data type of all columns in the "customers" table.

2) Get the time range during which the orders were placed.

3) Count the Cities & States of customers who ordered during the given period.\*/

#a) Data type of all columns in the "customers" table.

```
DESC customers;
```

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

```
10
11 #1)1) Data type of all columns in the "customers" table.
12 • DESC customers;
13
14 #2)Get the time range during which the orders were placed.
15 • SELECT
```

The left sidebar shows the 'SCHEMAS' pane with a tree view of databases. The 'Table: customers' is selected, and its columns are listed:

- customer\_id
- customer\_unique\_id
- customer\_ip\_code\_prefix
- customer\_city
- customer\_state

The 'Result Grid' shows the output of the 'DESC customers;' query:

Field	Type	Null	Key	Default	Extra
customer_id	text	YES			
customer_unique_id	text	YES			
customer_ip_code_prefix	int	YES			
customer_city	text	YES			
customer_state	text	YES			

The bottom status bar shows the following information:

- Object Info: Session
- Time: 37 23:25:20
- Action: CREATE DATABASE Mysql\_project
- Message: Error Code: 1007. Can't create database 'mysql\_project'; database exists
- Duration / Fetch: 0.063 sec
- 5 row(s) returned
- 0.125 sec / 0.000 sec

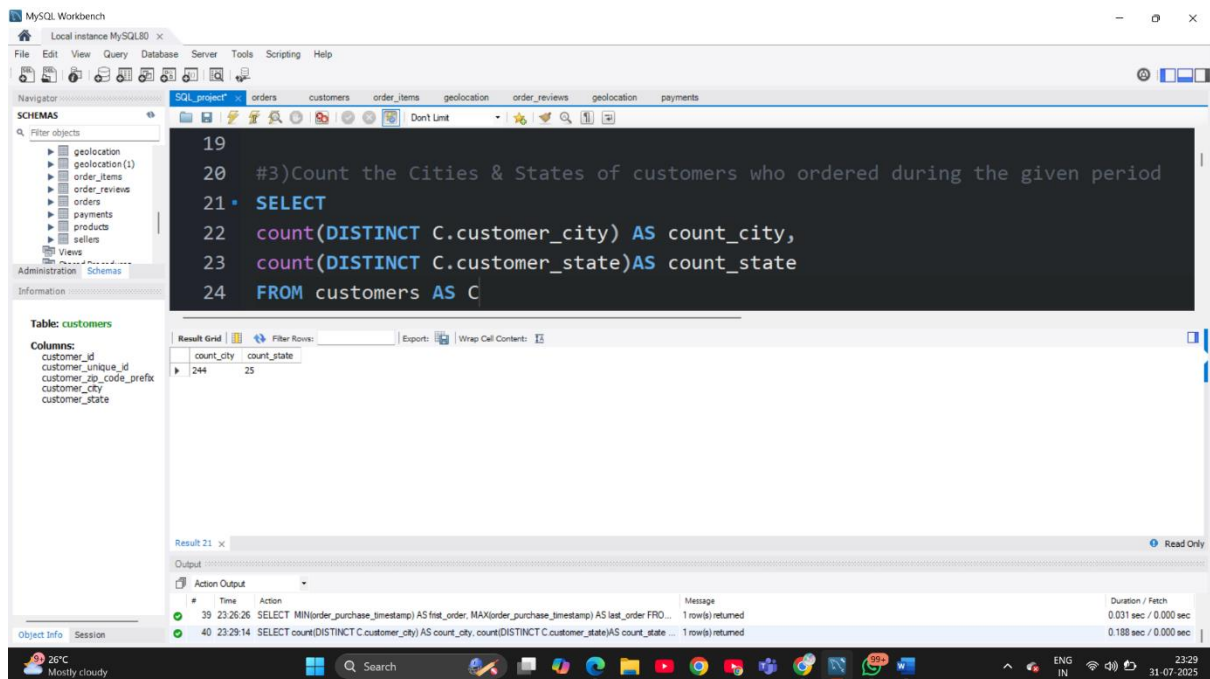
#b)Get the time range during which the orders were placed.

SELECT

MIN(order\_purchase\_timestamp) AS first\_order,

MAX(order\_purchase\_timestamp) AS last\_order

FROM orders;



#c)Count the Cities & States of customers who ordered during the given period

SELECT

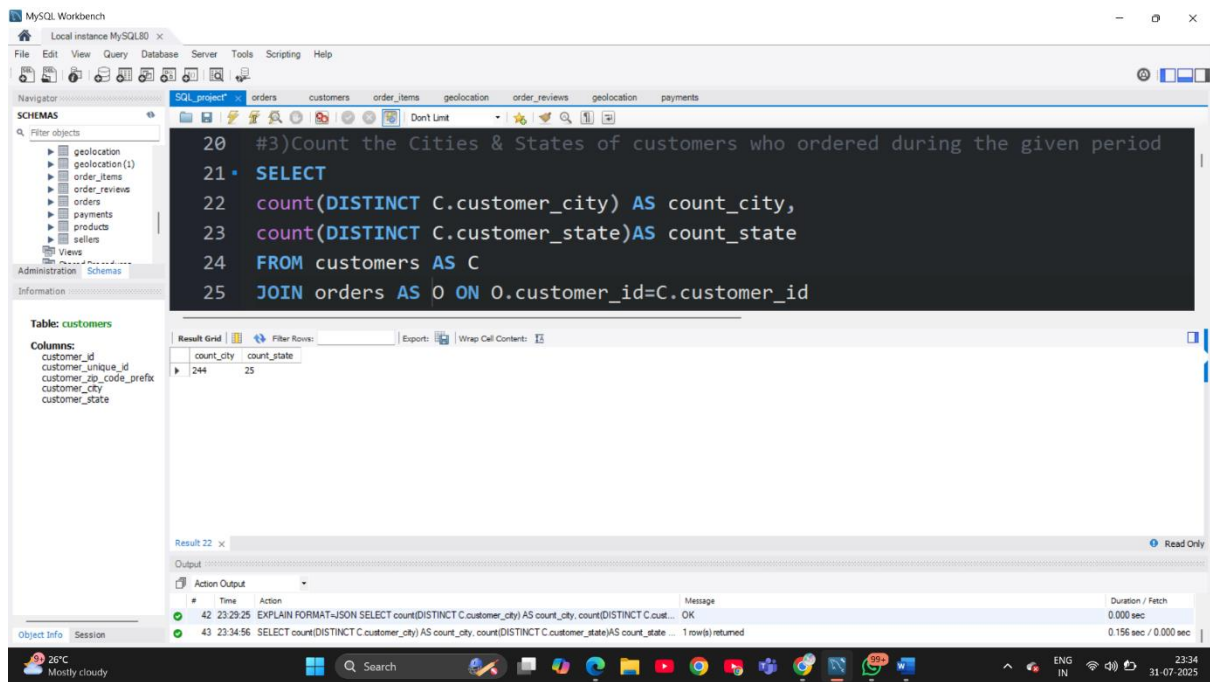
count(DISTINCT C.customer\_city) AS count\_city,

count(DISTINCT C.customer\_state)AS count\_state

FROM customers AS C

JOIN orders AS O ON O.customer\_id=C.customer\_id

WHERE order\_id is not null;



/\*2)In-depth Exploration:

a) Is there a growing trend in the no. of orders placed over the past years?

b)Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

c)During what time of the day do Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

0-6 hrs: Dawn

7-12 hrs: Mornings

13-18 hrs: Afternoon

19-23 hrs: Night\*/

#a)Is there a growing trend in the no. of orders placed over the past years?

SELECT

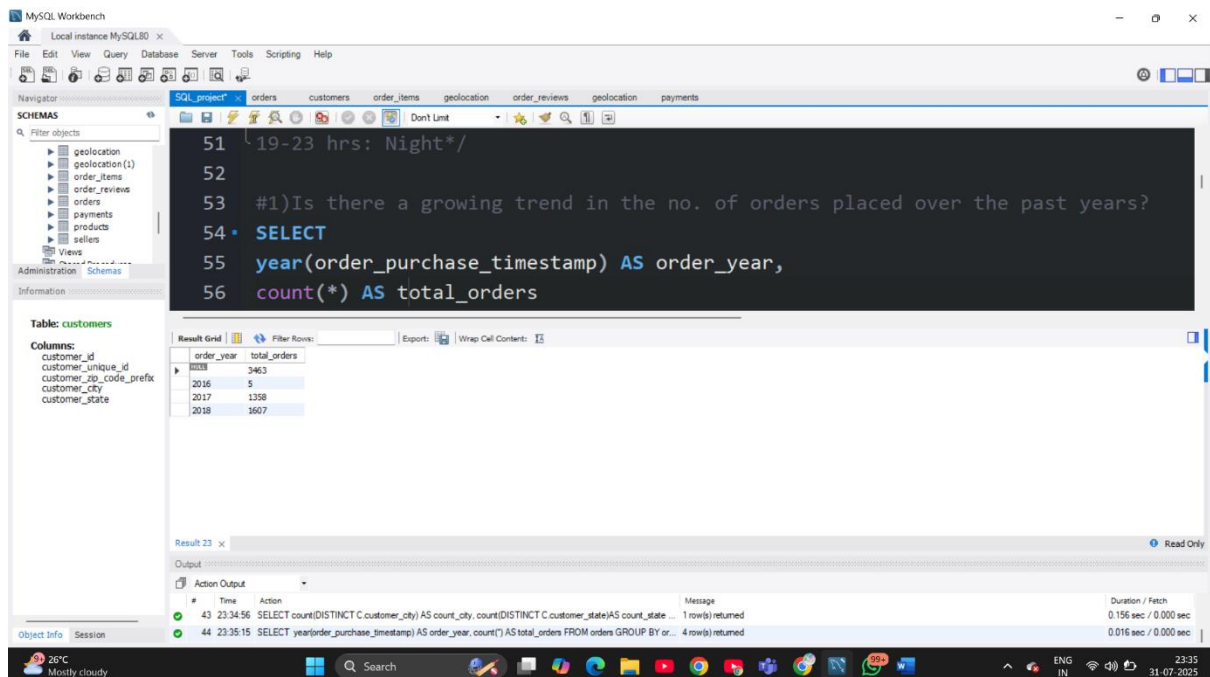
year(order\_purchase\_timestamp) AS order\_year,

count(\*) AS total\_orders

FROM orders

GROUP BY order\_year

ORDER BY order\_year ASC;



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

```
51 19-23 hrs: Night*/
52
53 #1)Is there a growing trend in the no. of orders placed over the past years?
54 • SELECT
55 year(order_purchase_timestamp) AS order_year,
56 count(*) AS total_orders
```

The Results tab shows the following data:

order_year	total_orders
2016	5
2017	1398
2018	1607

The bottom status bar shows the execution of the query: "44 23:35:15 SELECT year(order\_purchase\_timestamp) AS order\_year, count(\*) AS total\_orders FROM orders GROUP BY order\_year ORDER BY order\_year ASC; 4 row(s) returned. 0.016 sec / 0.000 sec".

/\*b)Can we see some kind of monthly seasonality in terms of the no. of orders being placed?\*/

SELECT

YEAR(order\_purchase\_timestamp)AS YEAR,

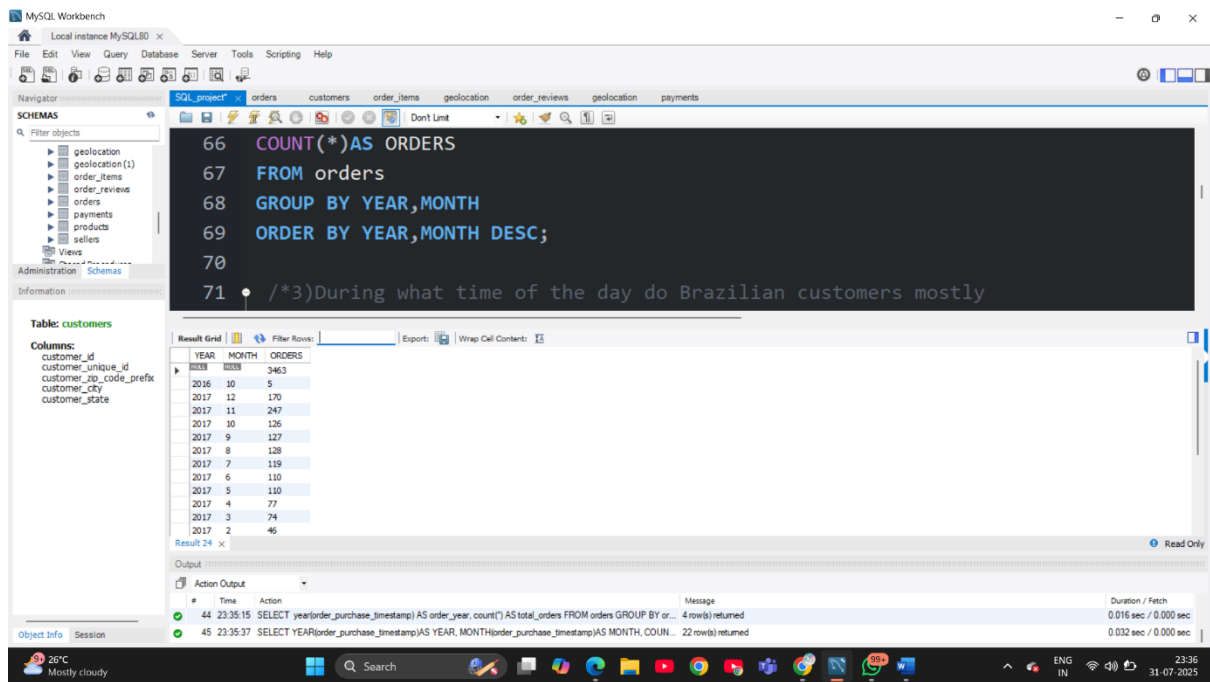
MONTH(order\_purchase\_timestamp)AS MONTH,

COUNT(\*)AS ORDERS

FROM orders

GROUP BY YEAR,MONTH

ORDER BY YEAR,MONTH DESC;



/\*c) During what time of the day do Brazilian customers mostly

place their orders? (Dawn, Morning, Afternoon or Night)

0-6 hrs: Dawn

7-12 hrs: Mornings

13-18 hrs: Afternoon

19-23 hrs: Night\*/

SELECT

CASE

WHEN HOUR(od.order\_purchase\_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'

WHEN HOUR(od.order\_purchase\_timestamp) BETWEEN 7 AND 12 THEN 'Mornings'

WHEN HOUR(od.order\_purchase\_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'

WHEN HOUR(od.order\_purchase\_timestamp) BETWEEN 19 AND 23 THEN 'Night'

END AS DAY\_TIME,

COUNT(\*) AS total\_orders

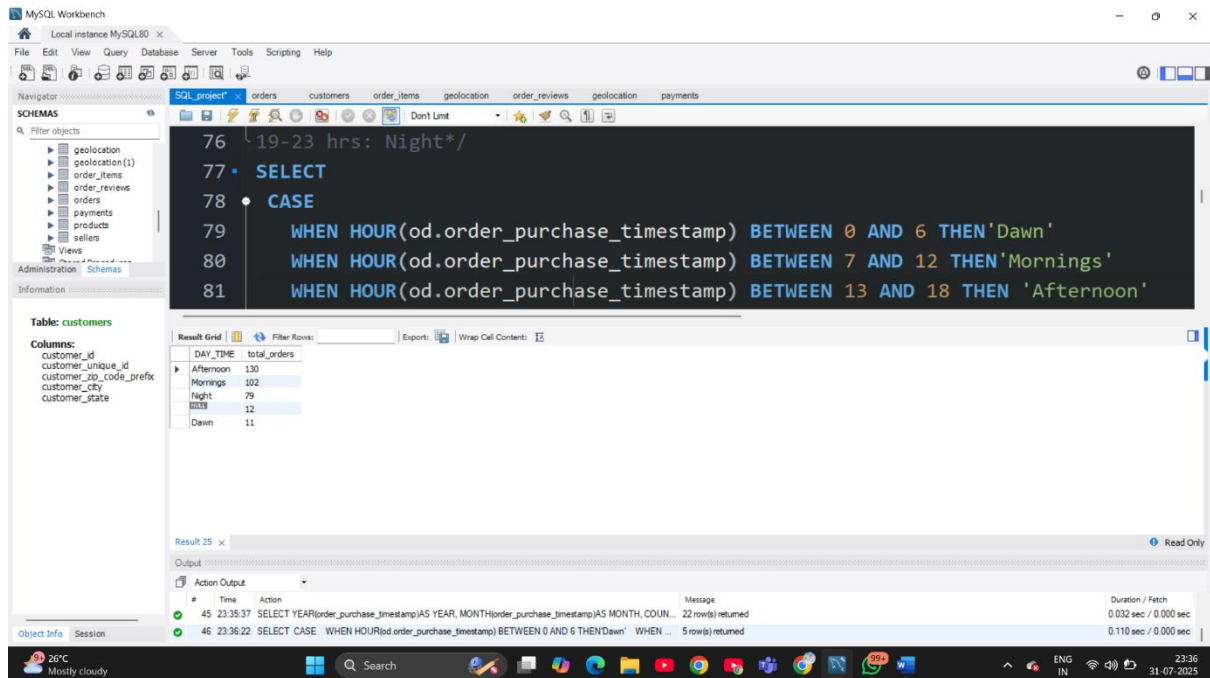
FROM orders AS od

JOIN customers AS c ON c.customer\_id=od.customer\_id

WHERE c.customer\_state like ('%sp%')

GROUP BY DAY\_TIME

ORDER BY total\_orders DESC;



/\*3)Evolution of E-commerce orders in the Brazil region:

a)Get the month-on-month no. of orders placed in each state.

b)How are the customers distributed across all the states?\*/

#a)Get the month-on-month no. of orders placed in each state

SELECT

year(o.order\_purchase\_timestamp) AS year,

MONTH(O.order\_purchase\_timestamp)AS month,

c.customer\_state AS STATE,

COUNT(\*)AS total\_orders

FROM orders AS o

JOIN customers AS C ON o.customer\_id=c.customer\_id

GROUP BY

year(o.order\_purchase\_timestamp),MONTH(O.order\_purchase\_timestamp),c.customer\_state

ORDER BY STATE,year,month DESC;

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

```
95 #1)Get the month-on-month no. of orders placed in each state
96 • SELECT
97 year(o.order_purchase_timestamp) AS year,
98 MONTH(O.order_purchase_timestamp)AS month,
99 c.customer_state AS STATE,
100 COUNT(*)AS total_orders
```

The results are displayed in a table with the following columns: year, month, STATE, total\_orders. The data is sorted by STATE, year, and month in descending order.

year	month	STATE	total_orders
2017	10	AL	1
2018	8	AL	1
2018	6	AL	4
2018	4	AH	1
2017	12	BA	2
2017	11	BA	3
2017	10	BA	7
2017	9	BA	1
2017	5	BA	2
2017	1	BA	2
2018	8	BA	2
2018	7	BA	4
2018	6	BA	1

The bottom of the screenshot shows the Action Output panel with the following message:

```
46 23:36:22 SELECT CASE WHEN HOUR(order_purchase_timestamp) BETWEEN 0 AND 6 THEN'Dawn' WHEN ... 5 row(s) returned
47 23:36:40 SELECT year(o.order_purchase_timestamp) AS year, MONTH(O.order_purchase_timestamp)AS month, c.cust... 181 row(s) returned
```

#b)How are the customers distributed across all the states?

SELECT

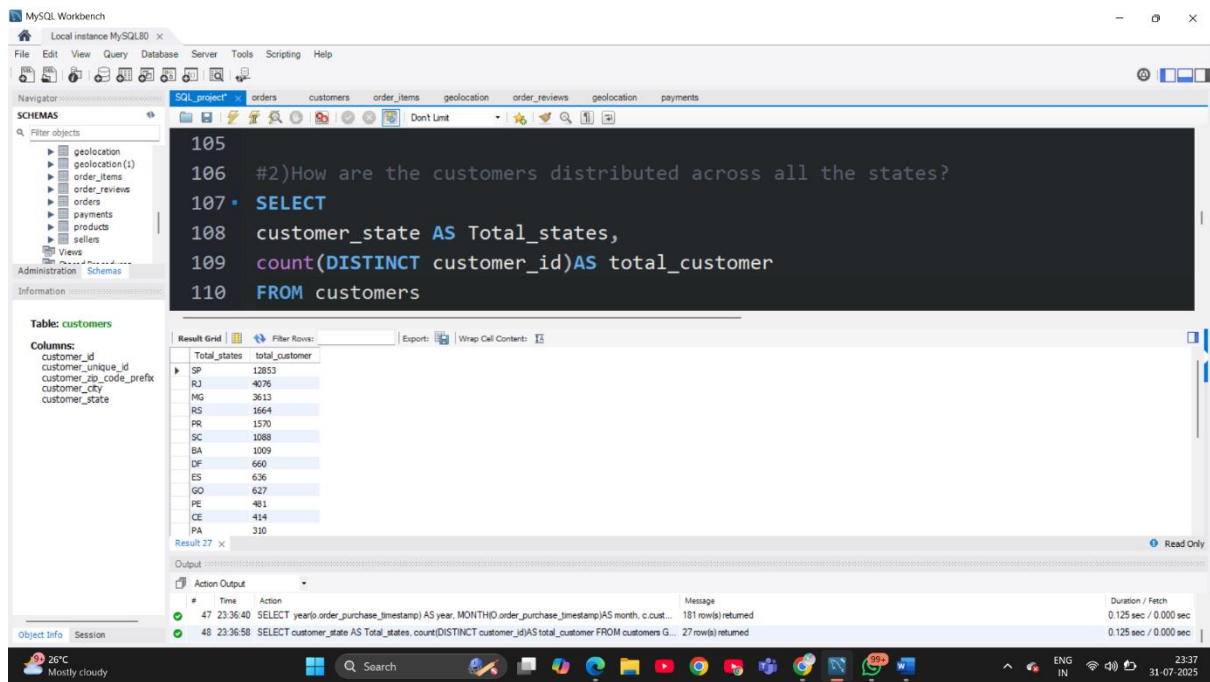
customer\_state AS Total\_states,

count(DISTINCT customer\_id)AS total\_customer

FROM customers

GROUP BY customer\_state

ORDER BY total\_customer DESC;



/\*4)Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.

a)Get the % increase in the cost of orders from year 2017 to 2018 (include months between January to Aug only).

You can use the "payment\_value" column in the payments table to get the cost of orders.

b)Calculate the Total & Average value of order price for each state.

c)Calculate the Total & Average value of order freight for each state.\*/

/\*a)Get the % increase in the cost of orders from year 2017 to 2018 (include months between January to Aug only).

You can use the "payment\_value" column in the payments table to get the cost of orders.\*/

```

SELECT
year(O.order_purchase_timestamp)AS order_year,
sum(P.payment_value)AS total_payment

```



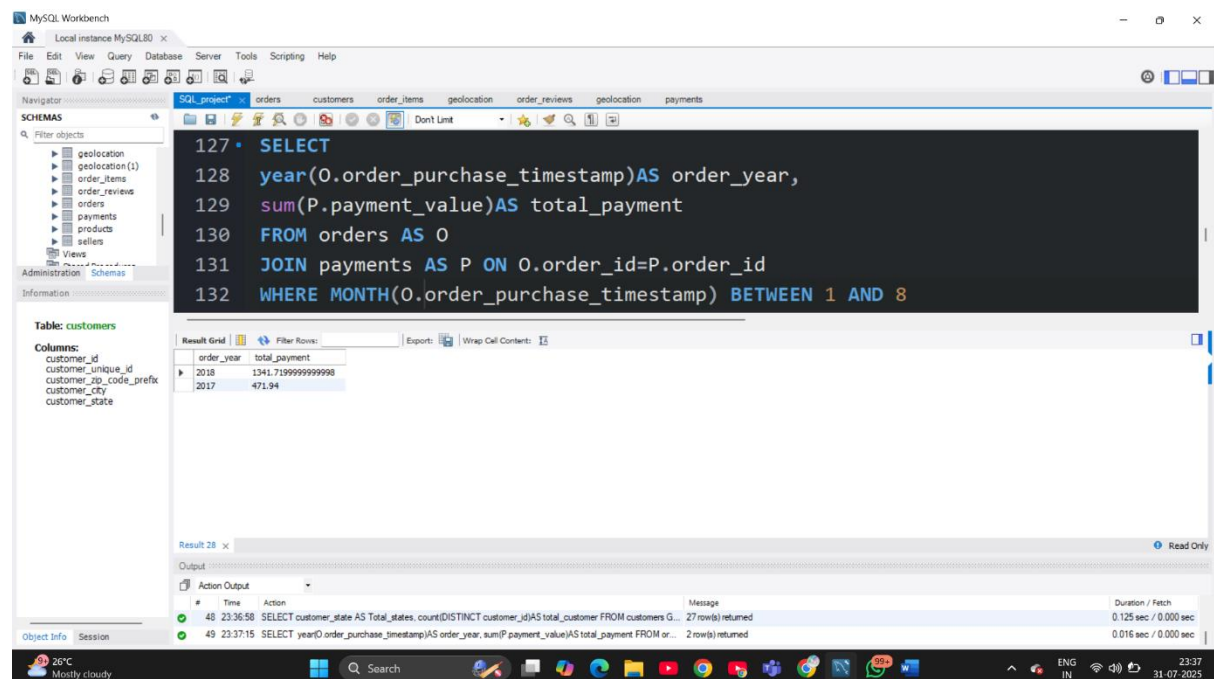
FROM orders AS O

JOIN payments AS P ON O.order\_id=P.order\_id

WHERE MONTH(O.order\_purchase\_timestamp) BETWEEN 1 AND 8

GROUP BY ORDER\_YEAR

ORDER BY ORDER\_YEAR DESC;



#b)Calculate the Total & Average value of order price for each state

SELECT

C.customer\_state AS state,

SUM(oi.price) AS total\_price,

AVG(oi.price)AS total\_avg

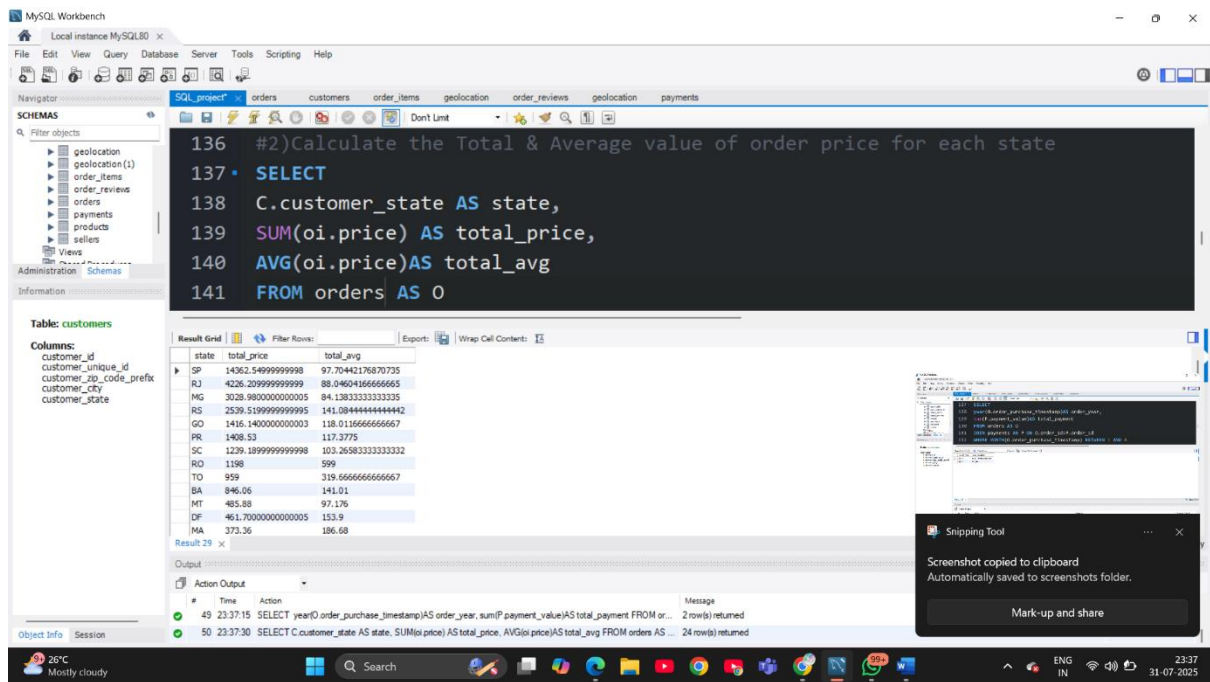
FROM orders AS O

JOIN customers AS C ON O.customer\_id=C.customer\_id

JOIN order\_items AS oi ON O.order\_id=oi.order\_id

GROUP BY state

ORDER BY total\_price DESC;



#c)Calculate the Total & Average value of order freight for each state

SELECT

C.customer\_state AS state,

SUM(oi.freight\_value) AS total\_freight,

AVG(oi.freight\_value) AS avg\_freight

FROM orders AS O

JOIN customers AS C ON O.customer\_id=C.customer\_id

JOIN order\_items AS oi ON O.order\_id=oi.order\_id

GROUP BY state

ORDER BY total\_freight;

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

```

148 • SELECT
149 C.customer_state AS state,
150 SUM(oi.freight_value) AS total_freight,
151 AVG(oi.freight_value) AS avg_freight
152 FROM orders AS O
153 JOIN customers AS C ON O.customer_id=C.customer_id

```

The results are displayed in a table with the following columns: state, total\_freight, and avg\_freight.

state	total_freight	avg_freight
ES	20.02	20.02
AM	22.85	22.85
PA	26.33	26.33
RN	27.07	27.07
SE	44.47	22.235
PI	49.01999999999999	24.509999999999998
PE	54.58	27.29
MS	68.25	34.125
PB	76.14	25.38
RO	77.62	38.81
MA	79.22999999999999	39.614999999999995
CE	105.25999999999999	26.314999999999998
DF	107.24000000000001	35.74666666666667

The bottom of the screenshot shows the Action Output panel with the following message:

```

50 23:37:30 SELECT C.customer_state AS state, SUM(oi.price) AS total_price, AVG(oi.price) AS total_avg FROM orders AS ... 24 row(s) returned
51 23:37:54 SELECT C.customer_state AS state, SUM(oi.freight_value) AS total_freight, AVG(oi.freight_value) AS avg_freigh... 24 row(s) returned

```

/\*5)Analysis based on sales, freight and delivery time.

a)Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

You can calculate the delivery time and the difference

between the estimated & actual delivery date using the given formula:

a)time\_to\_deliver = order\_delivered\_customer\_date - order\_purchase\_timestamp

b)diff\_estimated\_delivery = order\_delivered\_customer\_date - order\_estimated\_delivery\_date

b) Find out the top 5 states with the highest & lowest average freight value.

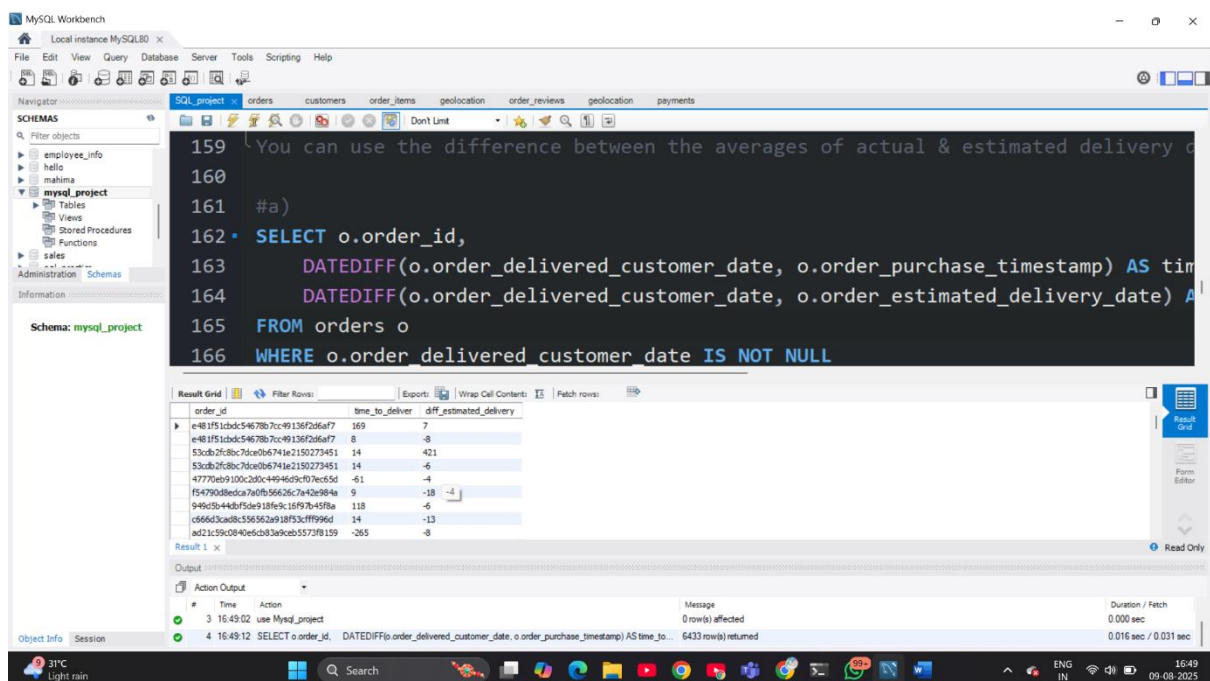
c)Find out the top 5 states with the highest & lowest average delivery time.

d)Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery dates to figure out how fast the delivery was for each state\*/

#a)

```
SELECT o.order_id,  
  
       DATEDIFF(o.order_delivered_customer_date, o.order_purchase_timestamp) AS  
time_to_deliver,  
  
       DATEDIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date) AS  
diff_estimated_delivery  
  
FROM orders o  
  
WHERE o.order_delivered_customer_date IS NOT NULL  
  
AND o.order_estimated_delivery_date IS NOT NULL  
  
AND o.order_purchase_timestamp IS NOT NULL;
```



#b) Find out the top 5 states with the highest & lowest average freight value

SELECT

c.customer\_state AS state,

```

ROUND(AVG(oi.freight_value),2) AS avg_freight

FROM order_items oi

JOIN orders o ON oi.order_id = o.order_id

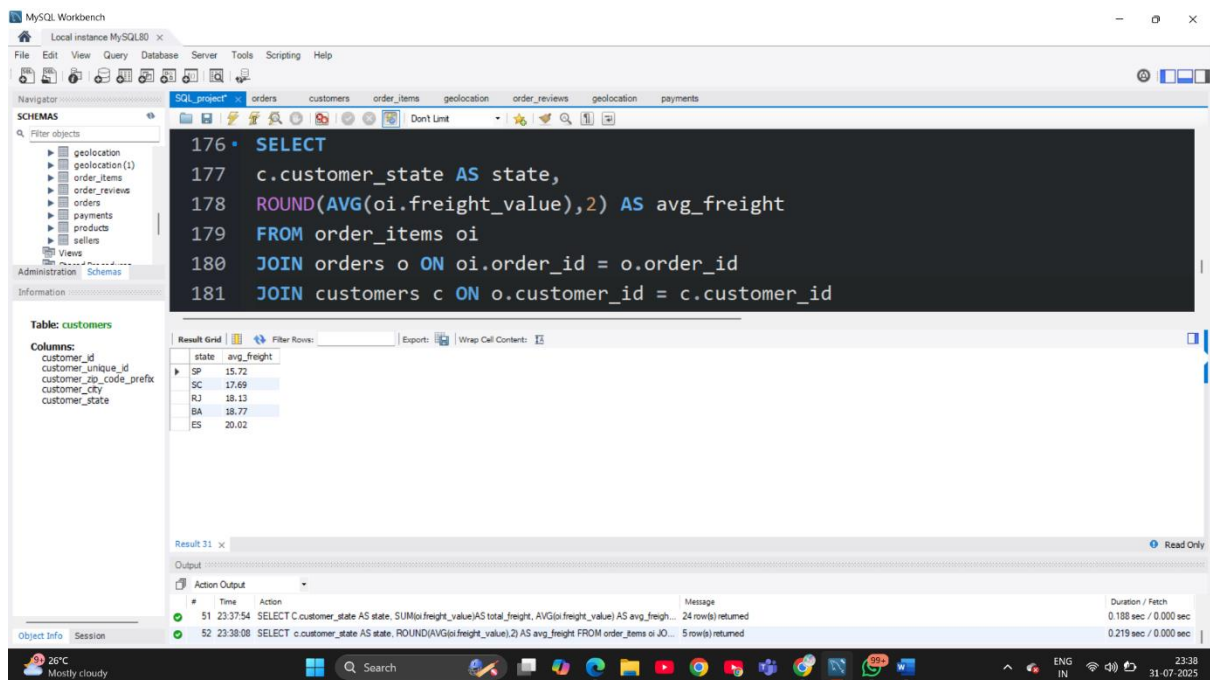
JOIN customers c ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY avg_freight ASC

LIMIT 5;

```



#c) Find out the top 5 states with the highest & lowest average delivery time.

```

SELECT

c.customer_state,

ROUND(AVG(DATEDIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp)),2) AS avg_delivery_days

FROM orders o

JOIN customers c ON o.customer_id = c.customer_id

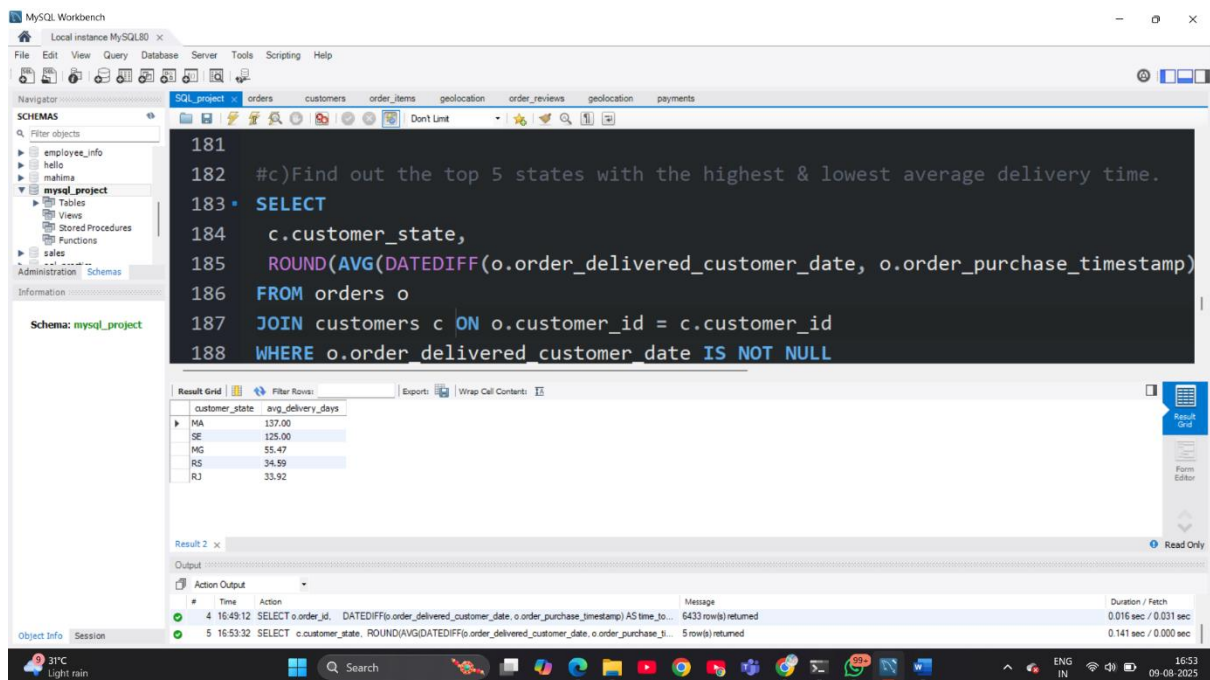
WHERE o.order_delivered_customer_date IS NOT NULL

GROUP BY c.customer_state

ORDER BY avg_delivery_days DESC

```

LIMIT 5;



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

```
181
182 #c)Find out the top 5 states with the highest & lowest average delivery time.
183 • SELECT
184     c.customer_state,
185     ROUND(AVG(DATEDIFF(o.order_delivered_customer_date, o.order_purchase_timestamp))
186 FROM orders o
187 JOIN customers c ON o.customer_id = c.customer_id
188 WHERE o.order_delivered_customer_date IS NOT NULL
```

The Results grid shows the following data:

customer_state	avg_delivery_days
MA	137.00
SE	125.00
MG	55.47
RS	34.99
RJ	33.92

The bottom panel shows the Action Output with the following message:

```
4 16:49:12 SELECT o.order_id, DATEDIFF(o.order_delivered_customer_date, o.order_purchase_timestamp) AS time_to... 6433 row(s) returned
5 16:53:32 SELECT c.customer_state, ROUND(AVG(DATEDIFF(o.order_delivered_customer_date, o.order_purchase_timestamp)) AS avg_delivery_days 5 row(s) returned
```

#d)Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

SELECT

c.customer\_state,

ROUND(AVG(DATEDIFF(o.order\_estimated\_delivery\_date,  
o.order\_delivered\_customer\_date)), 2) AS avg\_days\_early

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

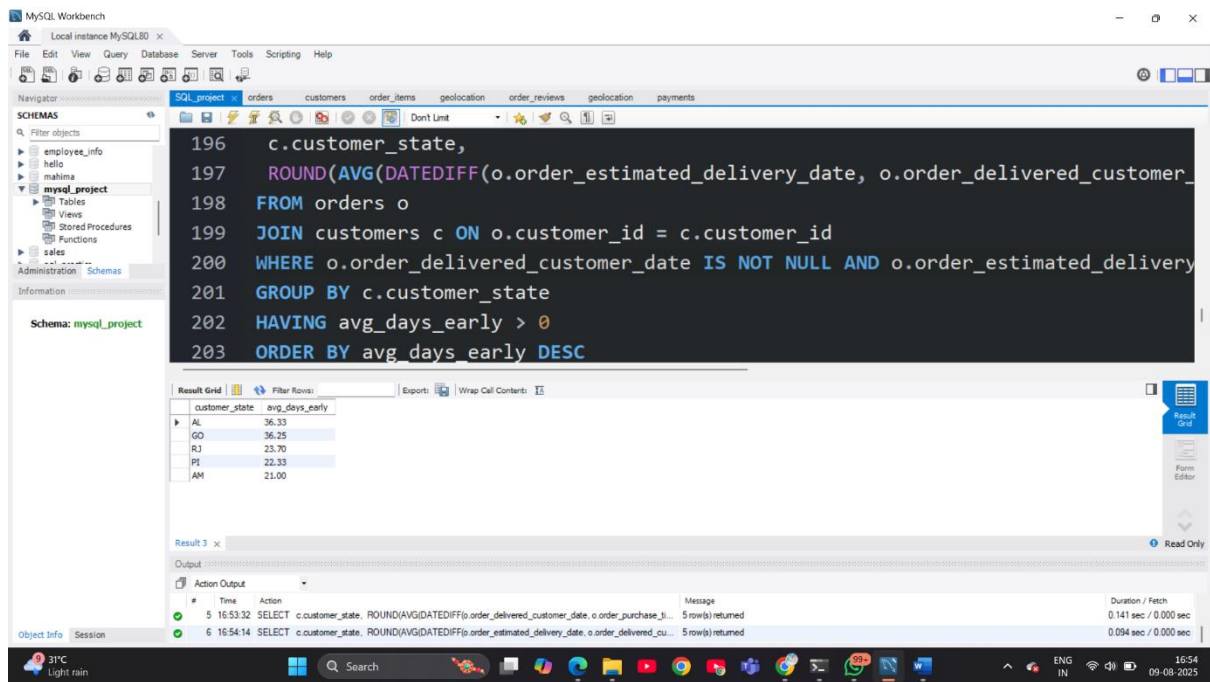
WHERE o.order\_delivered\_customer\_date IS NOT NULL AND  
o.order\_estimated\_delivery\_date IS NOT NULL

GROUP BY c.customer\_state

HAVING avg\_days\_early > 0

ORDER BY avg\_days\_early DESC

LIMIT 5;



/\*6)Analysis based on the payments:

a)Find the month-on-month no. of orders placed using different payment types.

b)Find the no. of orders placed based on the payment instalments

that have been paid.\*/

#a)Find the month-on-month no. of orders placed using different payment types.

SELECT

YEAR(o.order\_purchase\_timestamp) AS order\_year,

MONTH(o.order\_purchase\_timestamp) AS order\_month,

p.payment\_type AS payment,

COUNT(DISTINCT o.order\_id) AS total\_orders

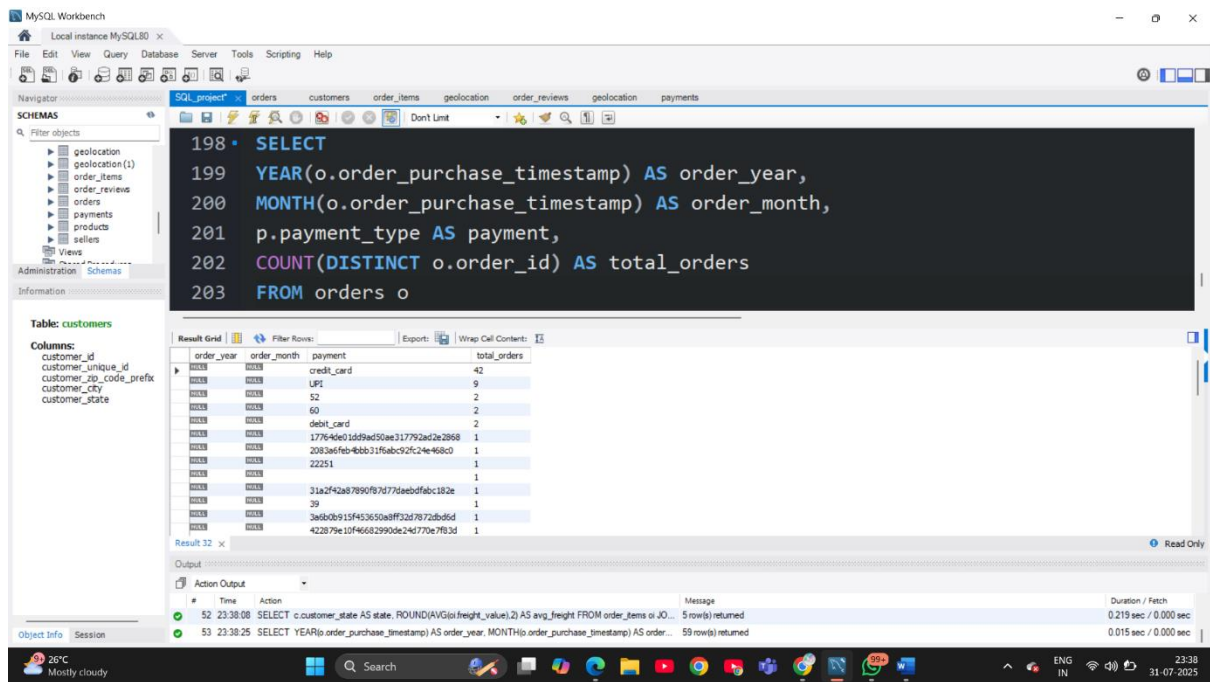
FROM orders o

JOIN payments p ON o.order\_id = p.order\_id

GROUP BY order\_year, order\_month, p.payment\_type

ORDER BY order\_year, order\_month, total\_orders DESC;





#b) Find the no. of orders placed based on the payment instalments that have been paid

SELECT

payment\_installments,

COUNT(DISTINCT order\_id) AS total\_orders

FROM payments

GROUP BY payment\_installments

ORDER BY total\_orders DESC;



MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SQL\_project orders customers order\_items geolocation order\_reviews geolocation payments

SCHEMAS

Filter objects

- geolocation
- geolocation(1)
- order\_items
- order\_reviews
- orders
- payments
- products
- sales
- Views

Administration Schemas

Information

Table: customers

Columns:

- customer\_id
- customer\_unique\_id
- customer\_zip\_code\_prefix
- customer\_city
- customer\_state

```
207
208 #2)Find the no. of orders placed based on the payment instalmentsthat have been
209 • SELECT
210 payment_installments,
211 COUNT(DISTINCT order_id) AS total_orders
212 FROM payments
```

Result Grid

	payment_installments	total_orders
1		261
2		62
3		46
4		31
8		27
5		25
10		22
6		18
7		8
12		3
123		2
202		2
275		2

Result 33 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
53	23:38:25	SELECT YEAR(B order_purchase_timestamp) AS order_year, MONTH(B order_purchase_timestamp) AS order...	59 row(s) returned	0.015 sec / 0.000 sec
54	23:38:40	SELECT payment_installments, COUNT(DISTINCT order_id) AS total_orders FROM payments GROUP BY pay...	119 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

26°C Mostly cloudy

Search

ENG IN

23:38 31-07-2025