

NAME - VEERESH WALI

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

1. Data type of columns in a table
2. Time period for which the data is given
3. Cities and States of customers ordered during the given period

Data type of columns in a table

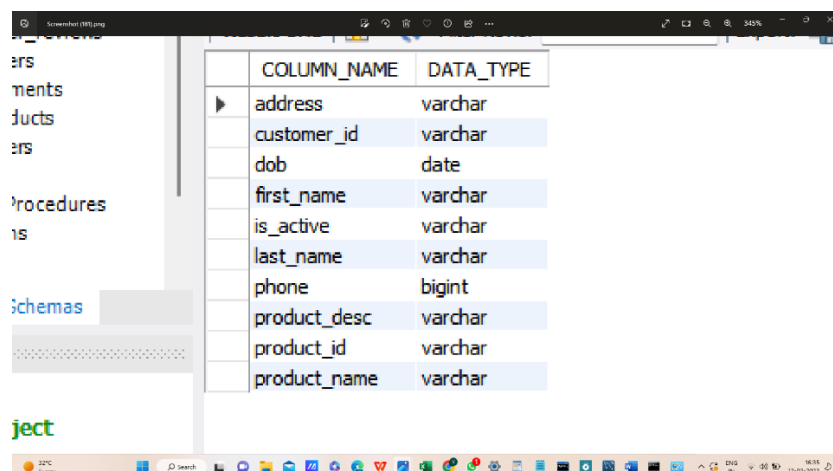
-- CUSTOMERS

```
select * from customers;
```

```
select column_name ,data_type
```

```
from information_schema.columns
```

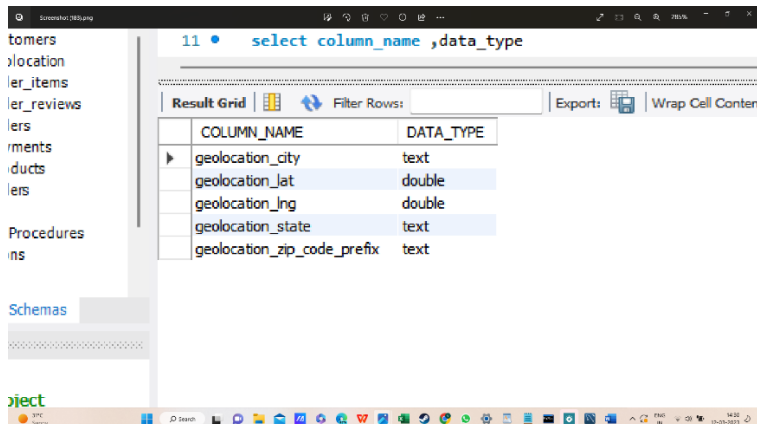
```
where table_name="customers";
```



	COLUMN_NAME	DATA_TYPE
	address	varchar
	customer_id	varchar
	dob	date
	first_name	varchar
	is_active	varchar
	last_name	varchar
	phone	bigint
	product_desc	varchar
	product_id	varchar
	product_name	varchar

-- GEOLOCATION

```
select * from geolocation;  
  
select column_name ,data_type  
from information_schema.columns  
where table_name="geolocation";
```

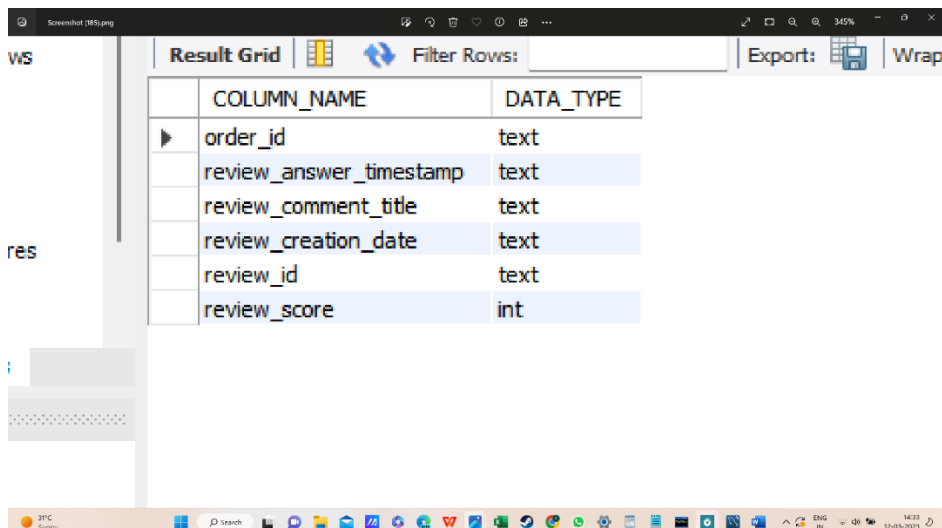


The screenshot shows a database client interface with a sidebar on the left containing a tree view of database objects. The main pane displays the results of a SQL query. The query is: `select column_name ,data_type`. The results are shown in a table with two columns: `COLUMN_NAME` and `DATA_TYPE`. The table contains five rows of data.

COLUMN_NAME	DATA_TYPE
geolocation_city	text
geolocation_lat	double
geolocation_lng	double
geolocation_state	text
geolocation_zip_code_prefix	text

-- ORDER_REVIEWS

```
select column_name ,data_type  
from information_schema.columns  
where table_name="order_reviews";
```

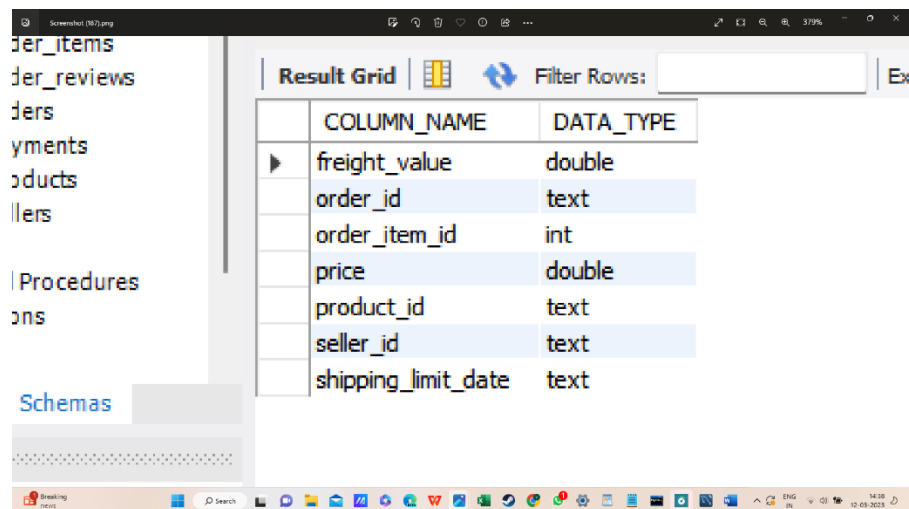


The screenshot shows a database client interface with a sidebar on the left containing a tree view of database objects. The main pane displays the results of a SQL query. The query is: `select column_name ,data_type`. The results are shown in a table with two columns: `COLUMN_NAME` and `DATA_TYPE`. The table contains six rows of data.

COLUMN_NAME	DATA_TYPE
order_id	text
review_answer_timestamp	text
review_comment_title	text
review_creation_date	text
review_id	text
review_score	int

-- ORDER_ITEMS

```
select column_name ,data_type
from information_schema.columns
where table_name="order_items";
```

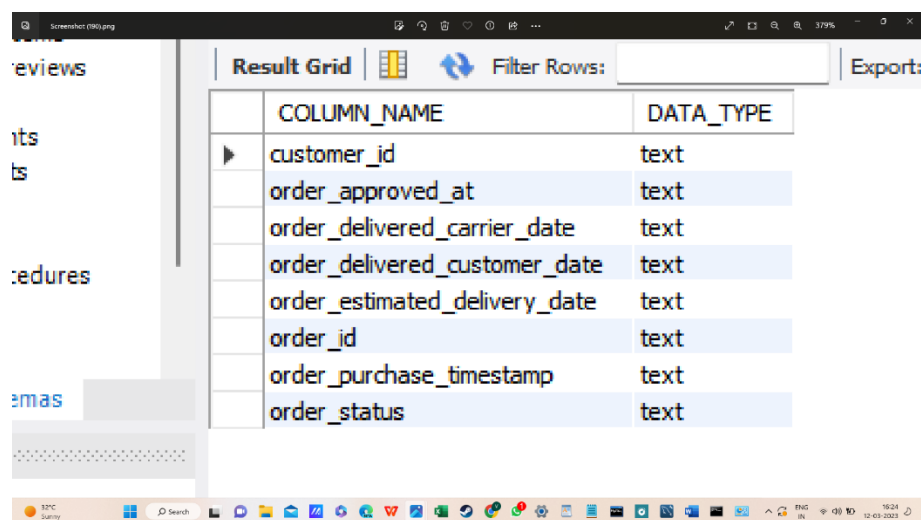


The screenshot shows a database management tool interface. On the left, there is a sidebar with a tree view containing 'Schemas' and a list of tables including 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'suppliers', 'Procedures', and 'views'. The 'order_items' table is selected. The main area displays a 'Result Grid' with the following data:

	COLUMN_NAME	DATA_TYPE
▶	freight_value	double
	order_id	text
	order_item_id	int
	price	double
	product_id	text
	seller_id	text
	shipping_limit_date	text

-- ORDERS

```
select column_name ,data_type
from information_schema.columns
where table_name="orders";
```



The screenshot shows a database management tool interface. On the left, there is a sidebar with a tree view containing 'Schemas' and a list of tables including 'order_reviews', 'orders', 'payments', 'products', 'suppliers', 'Procedures', and 'views'. The 'orders' table is selected. The main area displays a 'Result Grid' with the following data:

	COLUMN_NAME	DATA_TYPE
▶	customer_id	text
	order_approved_at	text
	order_delivered_carrier_date	text
	order_delivered_customer_date	text
	order_estimated_delivery_date	text
	order_id	text
	order_purchase_timestamp	text
	order_status	text

-- PAYMENTS

```
select column_name ,data_type
from information_schema.columns
where table_name="payments";
```

	COLUMN_NAME	DATA_TYPE
▶	order_id	text
	payment_installments	int
	payment_sequential	int
	payment_type	text
	payment_value	double

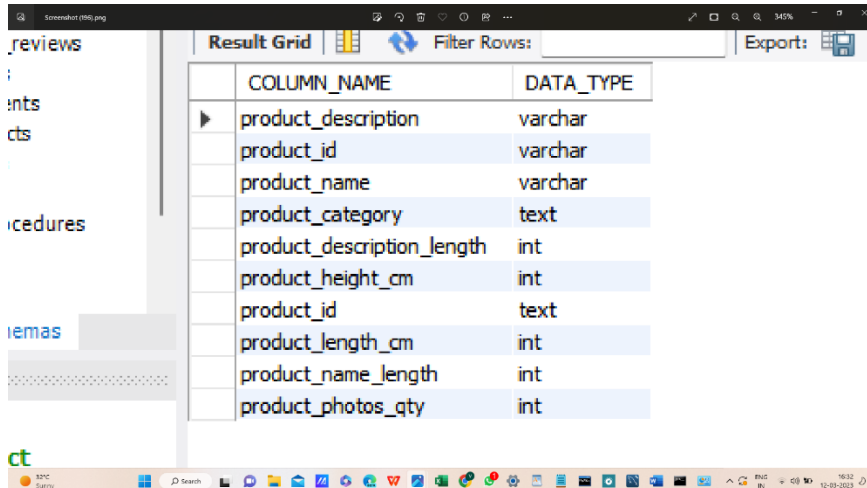
-- PRODUCTS

```
select column_name ,data_type
from information_schema.columns
where table_name="products";
```

	COLUMN_NAME	DATA_TYPE
▶	product_description	varchar
	product_id	varchar
	product_name	varchar
	product_category	text
	product_description_length	int
	product_height_cm	int
	product_id	text
	product_length_cm	int
	product_name_length	int
	product_photos_qty	int

-- SELLERS

```
select column_name ,data_type
from information_schema.columns
where table_name="sellers";
```



COLUMN_NAME	DATA_TYPE
product_description	varchar
product_id	varchar
product_name	varchar
product_category	text
product_description_length	int
product_height_cm	int
product_id	text
product_length_cm	int
product_name_length	int
product_photos_qty	int

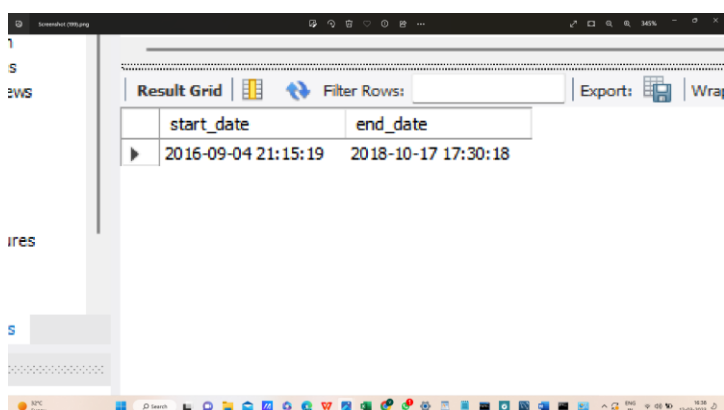
Time period for which the data is given

Select

```
min(order_purchase_timestamp) as start_date,
```

```
max(order_purchase_timestamp) as end_date
```

from orders;



start_date	end_date
2016-09-04 21:15:19	2018-10-17 17:30:18

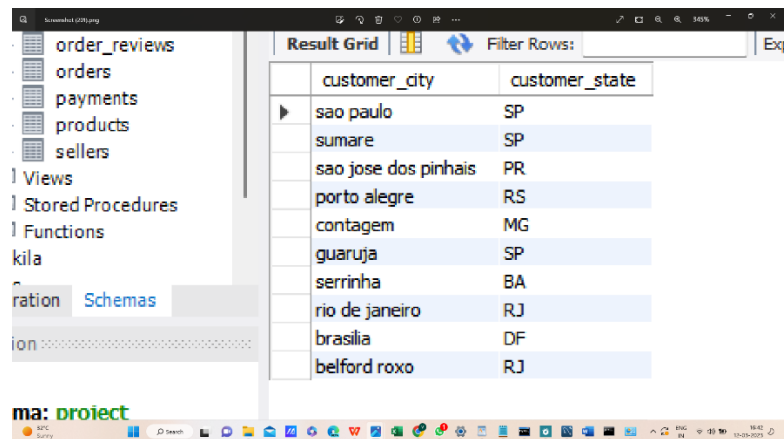
Cities and States of customers ordered during the given period

Select distinct customer_city, customer_state

from customers

Join orders ON customers.customer_id = orders.customer_id

where order_purchase_timestamp between '2016-09-04' and '2018-10-17'



The screenshot shows a database management system interface. On the left is a sidebar with a tree view containing 'order_reviews', 'orders', 'payments', 'products', 'sellers', 'Views', 'Stored Procedures', 'Functions', 'kila', 'ration', and 'ion'. The 'Schemas' tab is selected. The main area is titled 'Result Grid' and displays a table with two columns: 'customer_city' and 'customer_state'. The table contains ten rows of data, each representing a distinct city and state combination. The interface also includes a 'Filter Rows' input field and a 'ma: projeto' logo at the bottom left. The Windows taskbar is visible at the bottom of the screen.

customer_city	customer_state
sao paulo	SP
sumare	SP
sao jose dos pinhais	PR
porto alegre	RS
contagem	MG
guarujá	SP
serrinha	BA
rio de janeiro	RJ
brasília	DF
belford roxo	RJ

2) In-depth Exploration:

1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?
2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)

Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

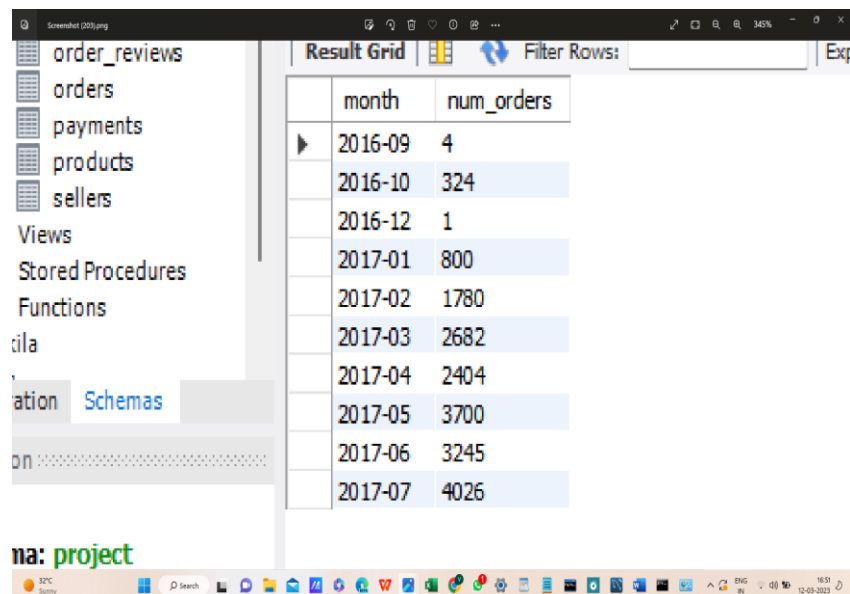
Total number of orders per month

Select date_format (order_purchase_timestamp, '%Y-%m') as month, count(*) as num_orders

from orders

group by month

order by month;

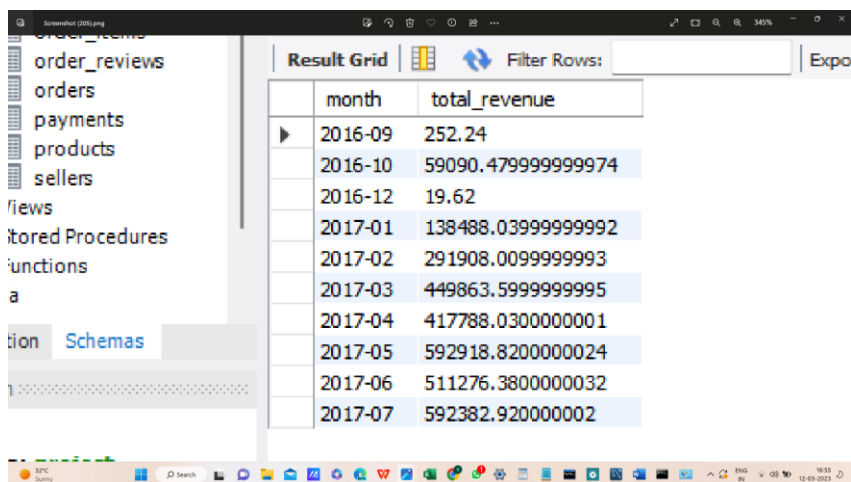


The screenshot shows a database management tool interface. On the left, there is a sidebar with a tree view containing 'order_reviews', 'orders', 'payments', 'products', 'sellers', 'Views', 'Stored Procedures', 'Functions', 'Data', 'Tables', 'Schemas', and 'Database'. The 'Schemas' tab is selected. The main area displays a 'Result Grid' with a table of order data. The table has two columns: 'month' and 'num_orders'. The data shows a clear upward trend from September 2016 to July 2017, with a significant peak in May 2017.

month	num_orders
2016-09	4
2016-10	324
2016-12	1
2017-01	800
2017-02	1780
2017-03	2682
2017-04	2404
2017-05	3700
2017-06	3245
2017-07	4026

Total revenue per month

Select date_format(order_purchase_timestamp, '%Y-%m') as month, Sum(payment_value) as total_revenue
From orders o
Join payments p on o.order_id = p.order_id
Group by month
Order BY month;

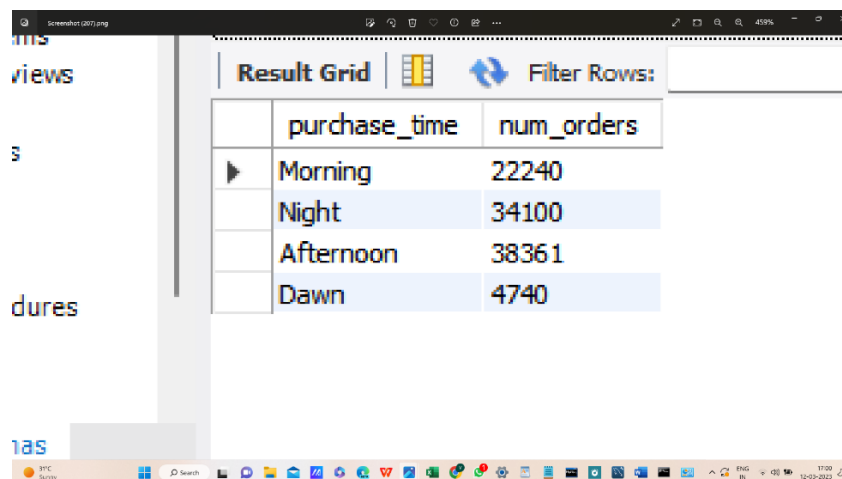


The screenshot shows a database interface with a 'Result Grid' displaying the output of a query. The left sidebar lists database objects: order_reviews, orders, payments, products, sellers, views, stored Procedures, functions, and a table named 'a'. The 'Result Grid' has a 'Filter Rows' button and an 'Export' button. The data is presented in a table with two columns: 'month' and 'total_revenue'. The rows show monthly revenue data from September 2016 to July 2017. The values for 'total_revenue' are formatted with commas as thousands separators.

month	total_revenue
2016-09	252.24
2016-10	59090.479999999974
2016-12	19.62
2017-01	138488.039999999992
2017-02	291908.00999999993
2017-03	449863.59999999995
2017-04	417788.03000000001
2017-05	592918.82000000024
2017-06	511276.38000000032
2017-07	592382.9200000002

What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
select
  case
    when hour (order_purchase_timestamp) >= 0 and hour(order_purchase_timestamp) < 6 then
      'Dawn'
    when hour (order_purchase_timestamp) >= 6 and hour(order_purchase_timestamp) < 12 then
      'Morning'
    when hour (order_purchase_timestamp) >= 12 and hour(order_purchase_timestamp) < 18 then
      'Afternoon'
    else 'Night'
  end as purchase_time,
  count(*) AS num_orders
from
  orders
group by
  purchase_time
limit 0, 10;
```



The screenshot shows a software interface with a 'Result Grid' tab. The grid contains four rows of data representing different times of day and the number of orders. The first row is 'Morning' with 22240 orders, the second is 'Night' with 34100 orders, the third is 'Afternoon' with 38361 orders, and the fourth is 'Dawn' with 4740 orders. The interface also includes a 'Filter Rows' button and a sidebar with various icons.

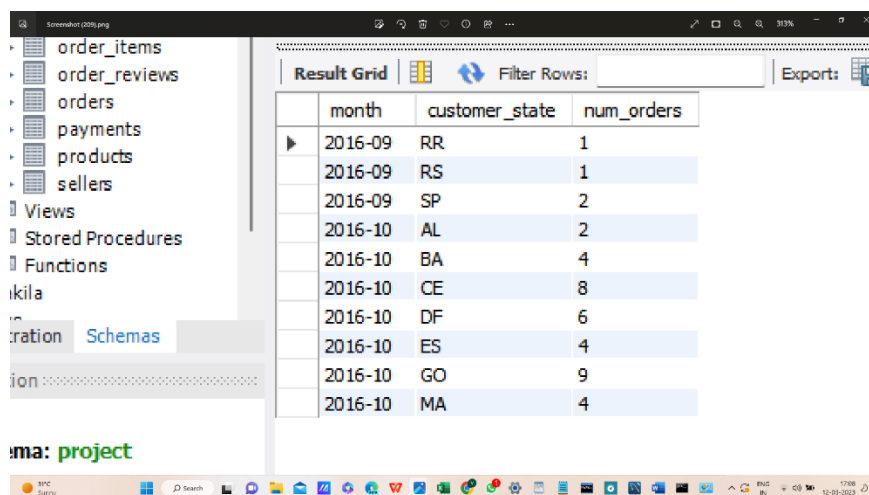
	purchase_time	num_orders
▶	Morning	22240
	Night	34100
	Afternoon	38361
	Dawn	4740

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

1. Get month on month orders by states
2. Distribution of customers across the states in Brazil

Get month on month orders by states

```
select
  date_format (order_purchase_timestamp, '%Y-%m') AS month,
  customer_state,
  count(*) AS num_orders
from
  orders o
join customers c ON o.customer_id = c.customer_id
group by
  month,
  customer_state
order by
  month,
  customer_state;
```



month	customer_state	num_orders
2016-09	RR	1
2016-09	RS	1
2016-09	SP	2
2016-10	AL	2
2016-10	BA	4
2016-10	CE	8
2016-10	DF	6
2016-10	ES	4
2016-10	GO	9
2016-10	MA	4

-- 2) Distribution of customers across the states in Brazil

```
select
  customer_state,
  count(distinct customer_unique_id) AS num_customers
from
  customers
group by
  customer_state
order by
  num_customers desc;
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Schemas' folder is expanded, showing the 'sakila' database. The 'Result Grid' tab is active, displaying a query result with two columns: 'customer_state' and 'num_customers'. The data is as follows:

customer_state	num_customers
SP	40302
RJ	12384
MG	11259
RS	5277
PR	4882
SC	3534
BA	3277
DF	2075
ES	1964
GO	1952

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

- Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table
- Mean & Sum of price and freight value by customer state

b) Mean & Sum of price and freight value by customer state

```

select
c.customer_state,
round(avg(op.price), 2) AS avg_price,
round(sum(op.price), 2) AS sum_price,
round(avg(op.freight_value), 2) AS avg_freight,
round(sum(op.freight_value), 2) AS sum_freight
from order_items op
join orders o on op.order_id = o.order_id
join customers c on o.customer_id = c.customer_id
where o.order_status in ('delivered', 'approved')
group by c.customer_state;

```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Schemas' folder is expanded, showing the 'sakila' database. The 'Result Grid' tab is active, displaying a query result with five columns: 'customer_state', 'avg_price', 'sum_price', 'avg_freight', and 'sum_freight'. The data is as follows:

customer_state	avg_price	sum_price	avg_freight	sum_freight
SP	109.1	5067803.06	15.12	702079.55
RS	118.83	728897.47	21.61	132575.32
SC	123.75	507012.13	21.51	88115.65
BA	134.02	493584.14	26.49	97553.67
MS	142.33	115429.97	23.35	18937.58
RJ	124.42	1759651.13	20.91	295750.44
PI	161.99	84721	39.12	20457.19
MG	120.18	1552521.53	20.62	266431.76
ES	120.74	268643.45	22.03	49014.48
RO	167.34	45682.76	41.33	11283.24

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery
2. Find `time_to_delivery` & `diff_estimated_delivery`. Formula for the same given below:
 - `time_to_delivery = order_purchase_timestamp - order_delivered_customer_date`
 - `diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date`
3. Group data by state, take mean of `freight_value`, `time_to_delivery`, `diff_estimated_delivery`
4. Sort the data to get the following:
5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
6. Top 5 states with highest/lowest average time to delivery
7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

2(1)time to delivery = order purchase timestamp - order delivered customer date

2(2)diff estimated delivery = order estimated delivery date - order delivered customer date

select

o.order_id,

timestampdiff(day, o.order_purchase_timestamp, o.order_delivered_customer_date) as

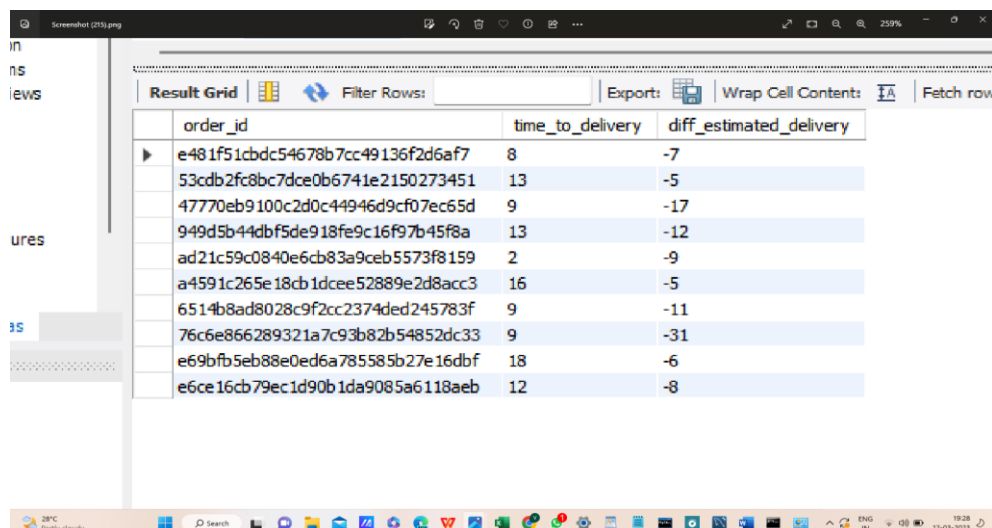
time_to_delivery,

timestampdiff(day, o.order_estimated_delivery_date, o.order_delivered_customer_date) as

diff_estimated_delivery

from orders o

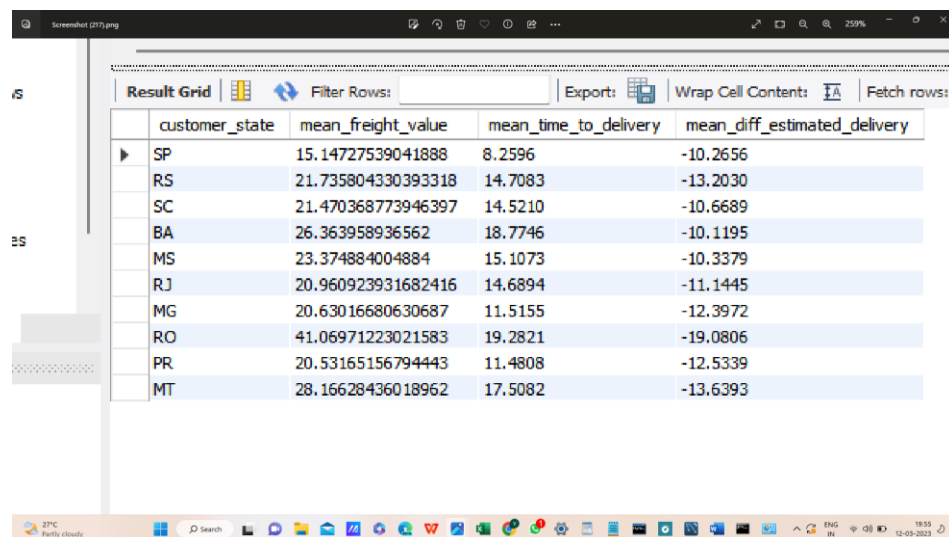
where o.order_status = 'delivered';



order_id	time_to_delivery	diff_estimated_delivery
e481f51cbdc54678b7cc49136f2d6af7	8	-7
53cdb2fc8bc7dce0b6741e2150273451	13	-5
47770eb9100c2d0c44946d9cf07ec65d	9	-17
949d5b44dbf5de918fe9c16f97b45f8a	13	-12
ad21c59c0840e6cb83a9ceb5573f8159	2	-9
a4591c265e18cb1dcee52889e2d8acc3	16	-5
6514b8ad8028c9f2cc2374ded245783f	9	-11
76c6e866289321a7c93b82b54852dc33	9	-31
e69bfb5eb88e0ed6a785585b27e16dbf	18	-6
e6ce16cb79ec1d90b1da9085a6118aeb	12	-8

Group data by state, take mean of freight value, time to delivery, diff estimated delivery

```
select c.customer_state,  
       avg(oi.freight_value) as mean_freight_value,  
       avg(timestampdiff(DAY, o.order_purchase_timestamp,  
o.order_delivered_customer_date)) as mean_time_to_delivery,  
       avg(timestampdiff(DAY, o.order_estimated_delivery_date,  
o.order_delivered_customer_date)) as mean_diff_estimated_delivery  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state;
```



The screenshot shows a database query result grid with the following data:

customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
SP	15.14727539041888	8.2596	-10.2656
RS	21.735804330393318	14.7083	-13.2030
SC	21.470368773946397	14.5210	-10.6689
BA	26.363958936562	18.7746	-10.1195
MS	23.374884004884	15.1073	-10.3379
RJ	20.960923931682416	14.6894	-11.1445
MG	20.63016680630687	11.5155	-12.3972
RO	41.06971223021583	19.2821	-19.0806
PR	20.53165156794443	11.4808	-12.5339
MT	28.16628436018962	17.5082	-13.6393

4)Sort the data to get the following:

```
select  
c.customer_state,  
       avg(oi.freight_value) as mean_freight_value,  
       avg(timestampdiff(day, o.order_purchase_timestamp, o.order_delivered_customer_date)) as  
mean_time_to_delivery,  
       AVG(timestampdiff(day, o.order_estimated_delivery_date, o.order_delivered_customer_date))  
as mean_diff_estimated_delivery  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state  
order by mean_freight_value desc, mean_time_to_delivery asc;
```

customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
RR	42.984423076923086	27.8261	-17.4348
PB	42.723803986711	20.1195	-12.1502
RO	41.06971223021583	19.2821	-19.0806
AC	40.0733695652174	20.3297	-20.0110
PI	39.147970479704824	18.9312	-10.6826
MA	38.25700242718449	21.2038	-9.1100
TO	37.24660317460317	17.0032	-11.4613
SE	36.653168831168806	20.9787	-9.1653
AL	35.84367117117114	23.9930	-7.9766
PA	35.83268518518515	23.3017	-13.3748

5) Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

```

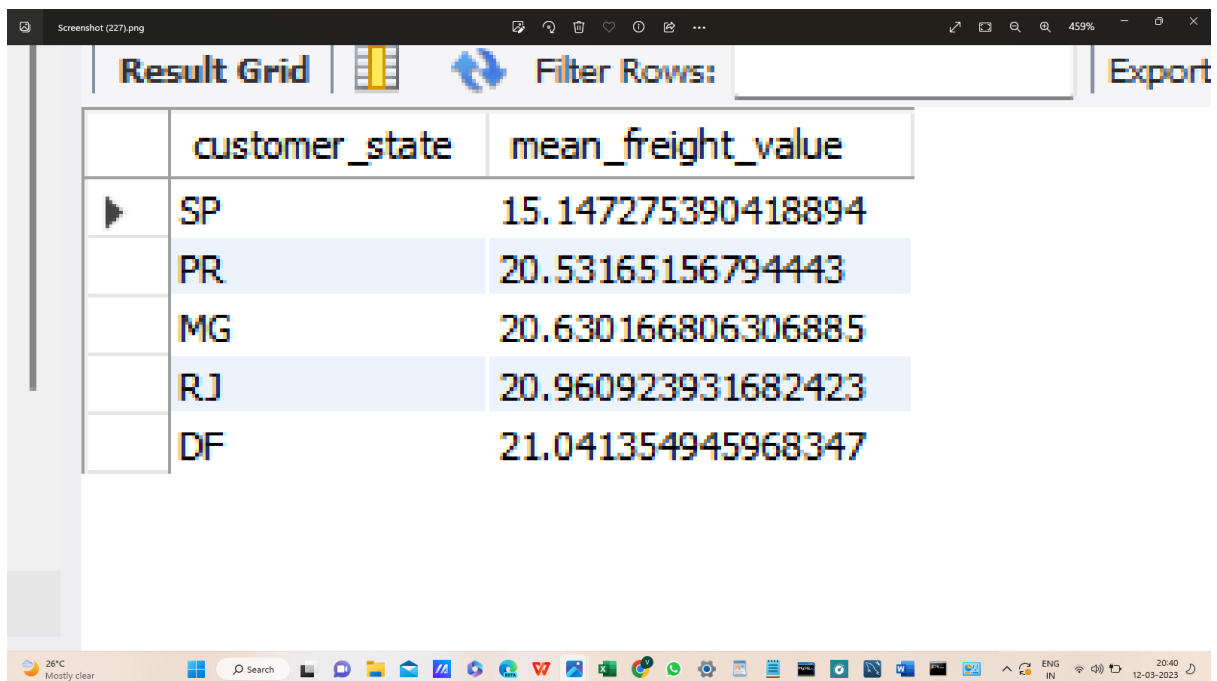
select c.customer_state,
       avg(oi.freight_value) as mean_freight_value
from orders o
join customers c on o.customer_id = c.customer_id
join order_items oi on o.order_id = oi.order_id
group by c.customer_state
order by mean_freight_value DESC
limit 5;

```

customer_state	mean_freight_value
RR	42.98442307692309
PB	42.723803986711
RO	41.06971223021582
AC	40.0733695652174
PI	39.147970479704824

lowest average freight value

```
select c.customer_state,  
       avg(oi.freight_value) as mean_freight_value  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state  
order by mean_freight_value asc  
limit 5;
```

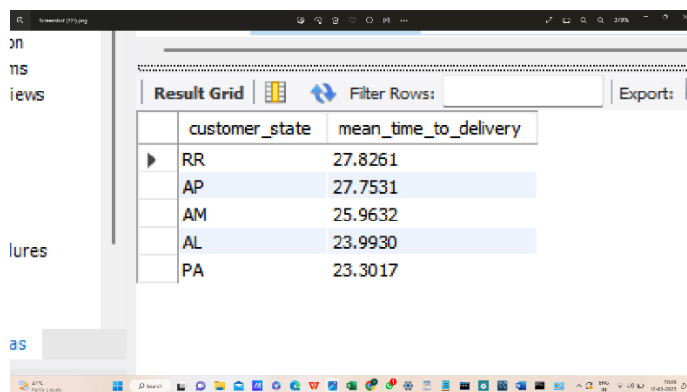


The screenshot shows a database query result grid with two columns: 'customer_state' and 'mean_freight_value'. The results are ordered by the mean freight value in ascending order. The states shown are SP, PR, MG, RJ, and DF. The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button. The Windows taskbar at the bottom shows the date as 12-03-2023 and the time as 20:40.

	customer_state	mean_freight_value
▶	SP	15.147275390418894
	PR	20.53165156794443
	MG	20.630166806306885
	RJ	20.960923931682423
	DF	21.041354945968347

6)Top 5 states with highest/lowest average time to delivery

```
select c.customer_state,  
       avg(timestampdiff(day, o.order_purchase_timestamp, o.order_delivered_customer_date)) AS  
mean_time_to_delivery  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state  
order by mean_time_to_delivery desc  
limit 5;
```

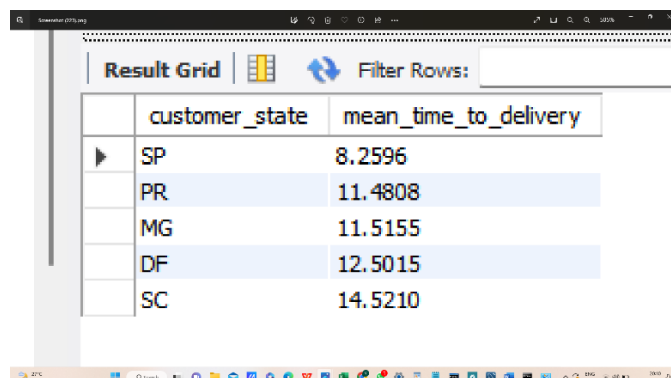


The screenshot shows a database query result in a 'Result Grid' format. The table has two columns: 'customer_state' and 'mean_time_to_delivery'. The data is sorted in descending order of the mean time to delivery. The top 5 states are RR, AP, AM, AL, and PA.

customer_state	mean_time_to_delivery
RR	27.8261
AP	27.7531
AM	25.9632
AL	23.9930
PA	23.3017

-- lowest average time to delivery

```
select c.customer_state,  
       avg(timestampdiff(day, o.order_purchase_timestamp, o.order_delivered_customer_date)) AS  
mean_time_to_delivery  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state  
order by mean_time_to_delivery asc  
limit 5;
```

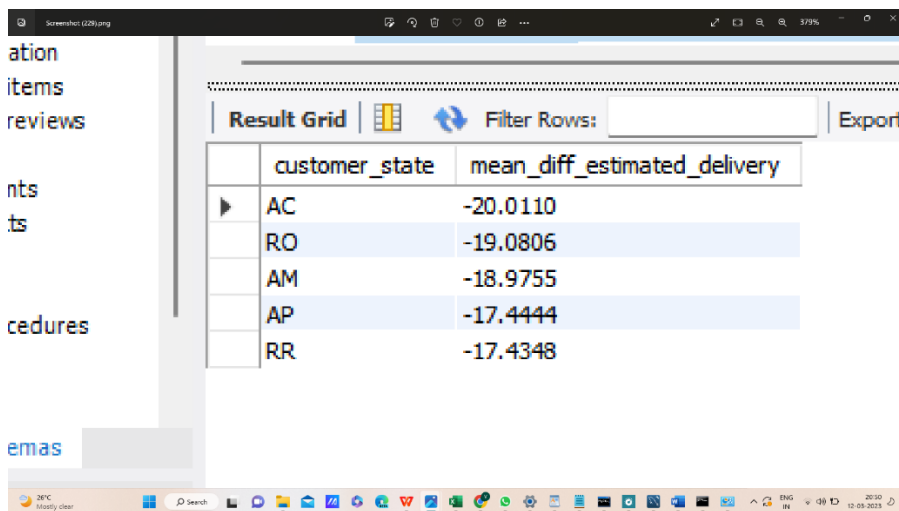


The screenshot shows a database query result in a 'Result Grid' format. The table has two columns: 'customer_state' and 'mean_time_to_delivery'. The data is sorted in ascending order of the mean time to delivery. The top 5 states are SP, PR, MG, DF, and SC.

customer_state	mean_time_to_delivery
SP	8.2596
PR	11.4808
MG	11.5155
DF	12.5015
SC	14.5210

7)Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
Select c.customer_state,  
       avg(timestampdiff(day, o.order_estimated_delivery_date, o.order_delivered_customer_date))  
as mean_diff_estimated_delivery  
from orders o  
join customers c on o.customer_id = c.customer_id  
join order_items oi on o.order_id = oi.order_id  
group by c.customer_state  
having mean_diff_estimated_delivery < 0  
order BY mean_diff_estimated_delivery ASC  
limit 5;
```



The screenshot shows a software interface with a sidebar on the left containing menu items like 'ation', 'items', 'reviews', 'nts', 'ts', 'cedures', and 'emas'. The main area displays a 'Result Grid' with a table of data. The table has two columns: 'customer_state' and 'mean_diff_estimated_delivery'. The data is sorted in ascending order of the mean difference, showing the top 5 states where delivery is faster than estimated (negative values).

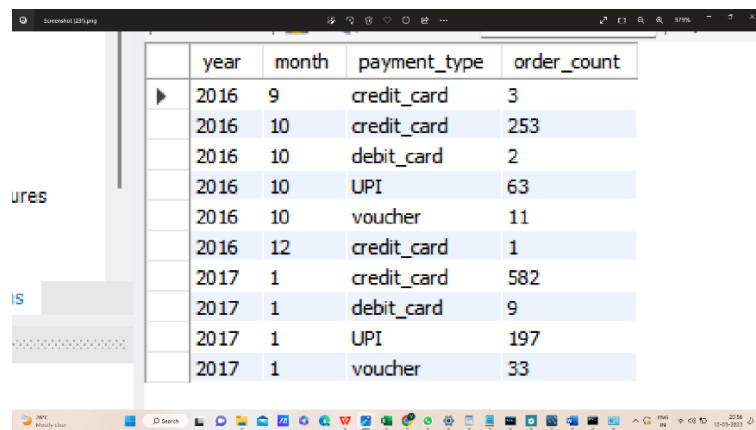
customer_state	mean_diff_estimated_delivery
AC	-20.0110
RO	-19.0806
AM	-18.9755
AP	-17.4444
RR	-17.4348

6. Payment type analysis:

1. Month over Month count of orders for different payment types
2. Count of orders based on the no. of payment instalments

6(1) Month over Month count of orders for different payment types

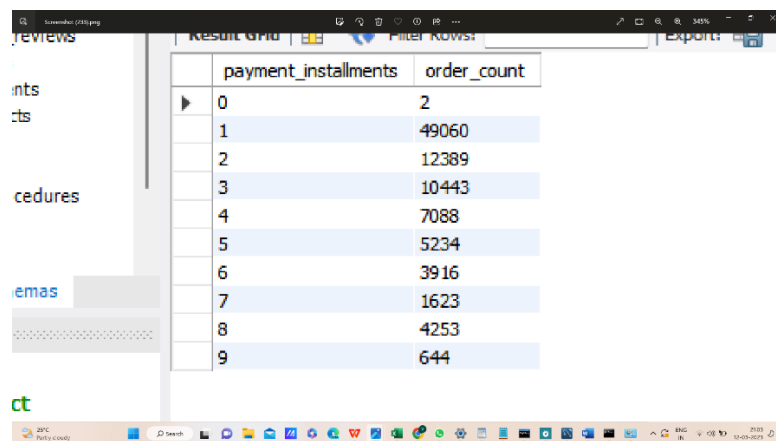
```
select
Year(o.order_purchase_timestamp) as year,
month(o.order_purchase_timestamp) as month,
p.payment_type,
count(distinct o.order_id) as order_count
from
payments p
join orders o on p.order_id = o.order_id
group by
year(o.order_purchase_timestamp),
month(o.order_purchase_timestamp),
p.payment_type;
```



	year	month	payment_type	order_count
	2016	9	credit_card	3
	2016	10	credit_card	253
	2016	10	debit_card	2
	2016	10	UPI	63
	2016	10	voucher	11
	2016	12	credit_card	1
	2017	1	credit_card	582
	2017	1	debit_card	9
	2017	1	UPI	197
	2017	1	voucher	33

6(2)Count of orders based on the no. of payment instalments

```
select
p.payment_installments,
count(distinct o.order_id) as order_count
from
payments p
join orders o on p.order_id = o.order_id
group by
p.payment_installments
```



payment_installments	order_count
0	2
1	49060
2	12389
3	10443
4	7088
5	5234
6	3916
7	1623
8	4253
9	644