Target Case Study

Target is a globally renowned brand and a prominent retailer in the United States.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions, including order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Some approaches we can draw from this dataset are:

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

- Data type of all columns in the "customers" table.
- Get the time range in which the orders were placed.

2. In-depth Exploration:

- Is there a growing trend in the number of orders placed over the past few years?
- Can we see some kind of monthly seasonality in terms of the number of orders being placed?
- During what time of the day do the Brazilian customers mostly place their orders?

3. Evolution of E-commerce Orders in the Brazil Region:

- Get the month-on-month number of orders placed in each state.
- How are the customers distributed across all the states?

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

- Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
- You can use the "payment_value" column in the payments table to get the cost of orders.

5. Analysis based on sales, freight and delivery time.

6. Analysis based on the payments.

Approach 1

Let's check the **structure & characteristics of the dataset.**

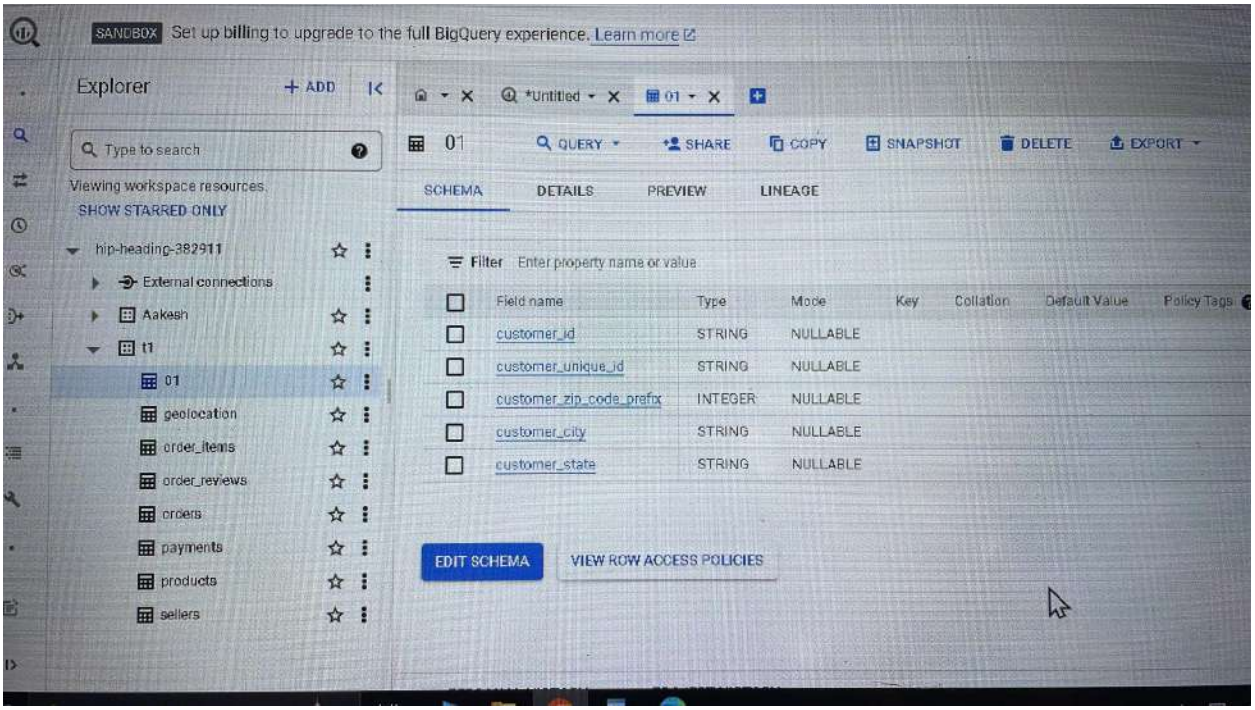
- **Data type** of all columns in the "**customers**" table.

1. The dataset has been uploaded in the BigQuery.

2. Now by simply double clicking on the "customers" table it will show the data type of all the columns in the "customers" table in the SCHEMA section .

3. There are 5 columns in the "customers" table i.e.:

1- customer_id (STRING), 2- customer_unique_id (STRING), 3- customer_zip_code_prefix (INTEGER), 4- customer_city (STRING), 5- customer_state (STRING)



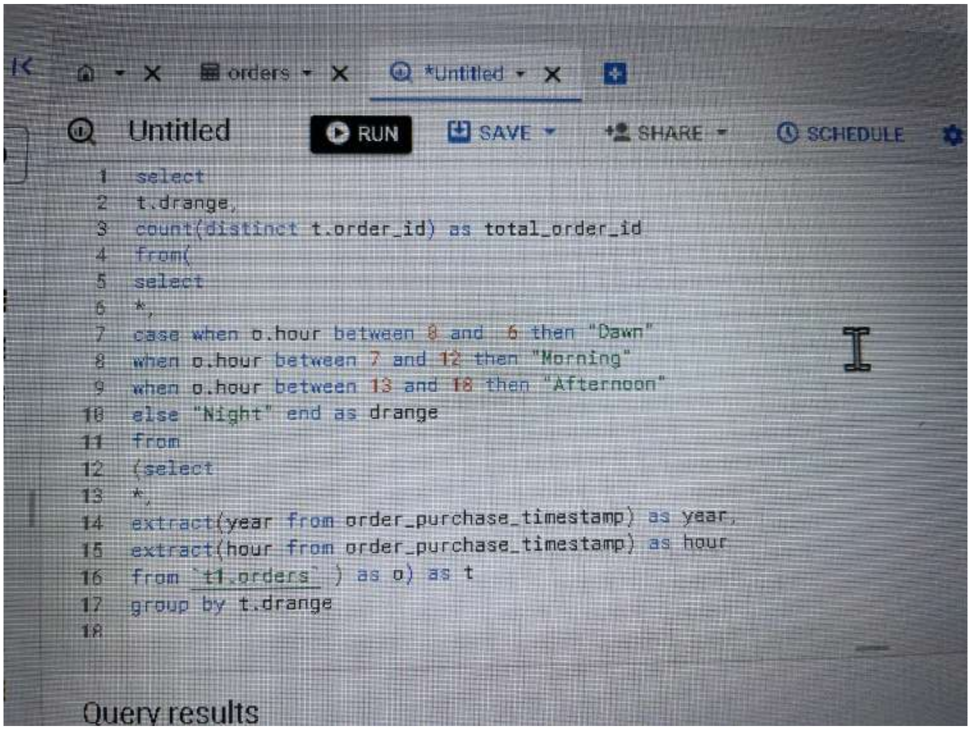
	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
□	customer_id	STRING	NULLABLE				
□	customer_unique_id	STRING	NULLABLE				
□	customer_zip_code_prefix	INTEGER	NULLABLE				
□	customer_city	STRING	NULLABLE				
□	customer_state	STRING	NULLABLE				

Note: Zoom the image to get a more clear picture.

Get the **time range** in which the **orders were placed**.

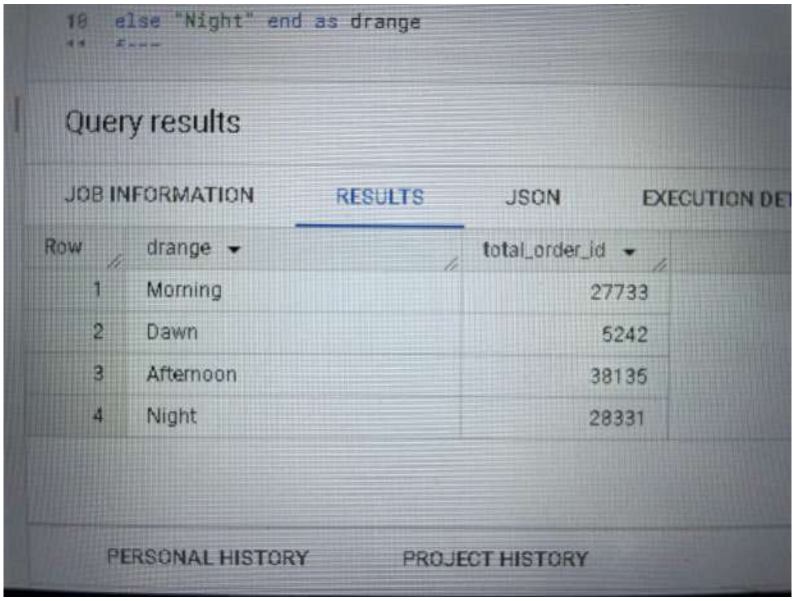
1. From the "order" table, we first extract hours from order_purchase_timestamp and then categorize those hours into 4 ranges: "dawn, morning, afternoon," and "Night".

2. Then we group by these ranges to get the count of Distinct "order_id" to get the total number of unique orders according to the range.



```sql
select
t.drange,
count(distinct t.order_id) as total_order_id
from(
select
*,
case when o.hour between 0 and  6 then "Dawn"
when o.hour between 7 and 12 then "Morning"
when o.hour between 13 and 18 then "Afternoon"
else "Night" end as drange
from
(select
*,
extract(year from order_purchase_timestamp) as year,
extract(hour from order_purchase_timestamp) as hour
from `t1.orders` ) as o) as t
group by t.drange
```

result :



```sql
else "Night" end as drange
```

Row	drange	total_order_id
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

As we can see, the Query output is:

1. Morning : 27733

2. Dawn: 5242

3. Afternoon: 38135

4. Night: 28331

Observation:

- **People are less likely to order while Dawn**
- **Whereas in afternoon, Brazilian people do place orders more as compared to other ranges**
- **Morning and night have almost same orders**

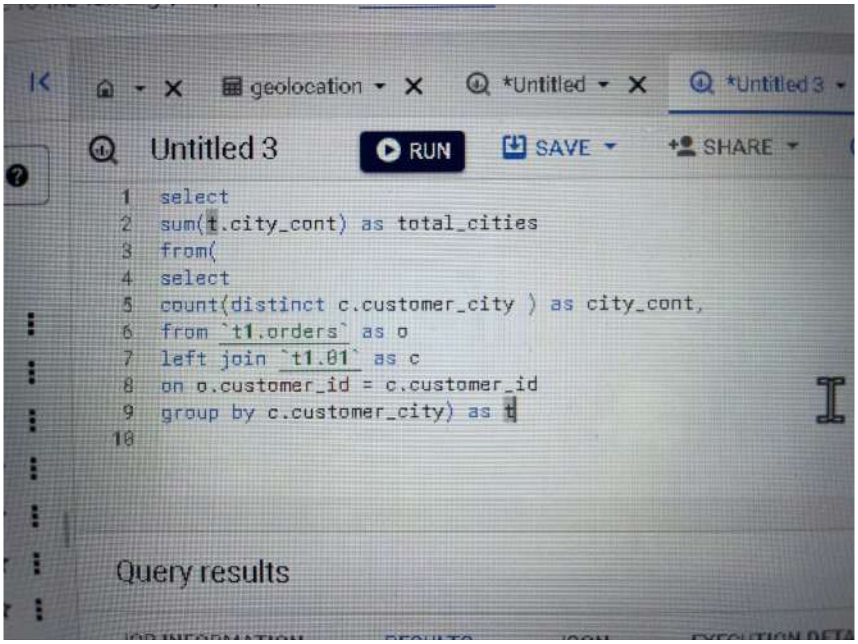
Count the **Cities and States of customers** who ordered during the given period.

1. First, we joined the "customer" table with the "order" table so that only the customers who have ordered can be displayed.

2. Then we'll group by "customer_city" and count the distinct cities in the table, and then apply a sum over it to get the total number of DISTINCT cities.

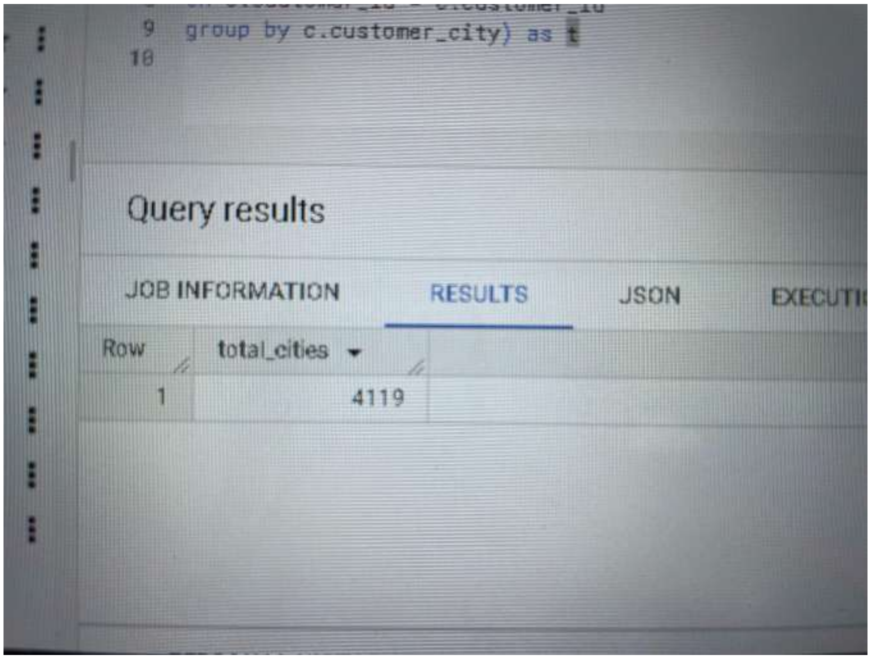
3. The same method will be repeated for getting the STATES

For City:



```
select
sum(t.city_cont) as total_cities
from(
select
count(distinct c.customer_city ) as city_cont,
from `t1.orders` as o
left join `t1.01` as c
on o.customer_id = c.customer_id
group by c.customer_city) as t
```

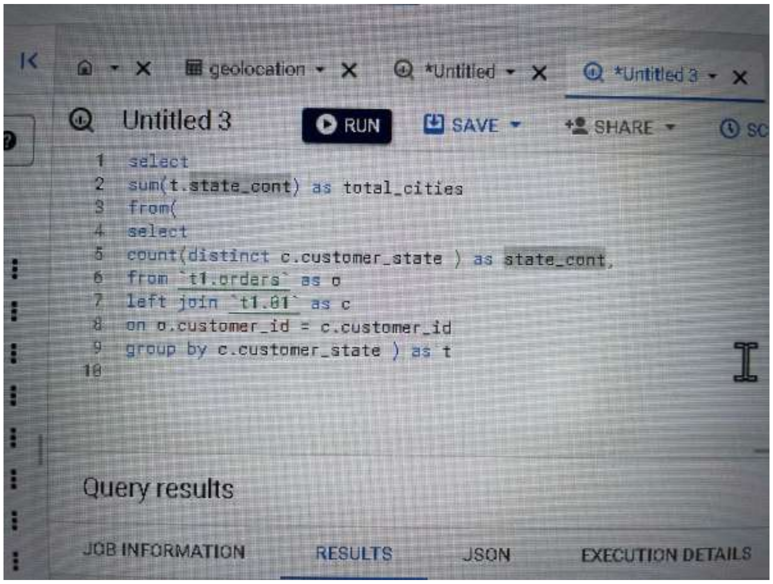
output:



```
group by c.customer_city) as t
```

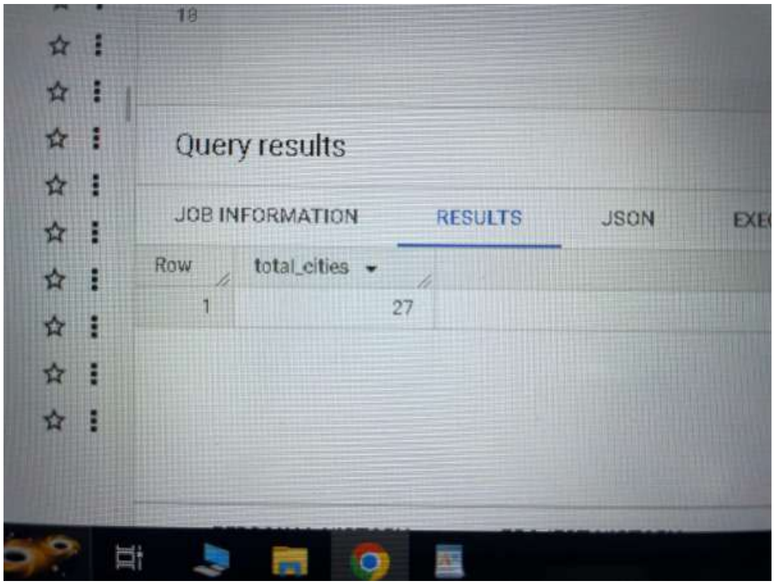
Row	total_cities
1	4119

For States:



```
select
sum(t.state_cont) as total_cities
from(
select
count(distinct c.customer_state ) as state_cont,
from `t1.orders` as o
left join `t1.01` as c
on o.customer_id = c.customer_id
group by c.customer_state ) as t
```

output:



Row	total_cities
1	27

Observation:

- Customers who order belong to a total of 4119 different cities.
- And totaling 27 different states from Brazil.

Approach 2

In-depth Exploration:

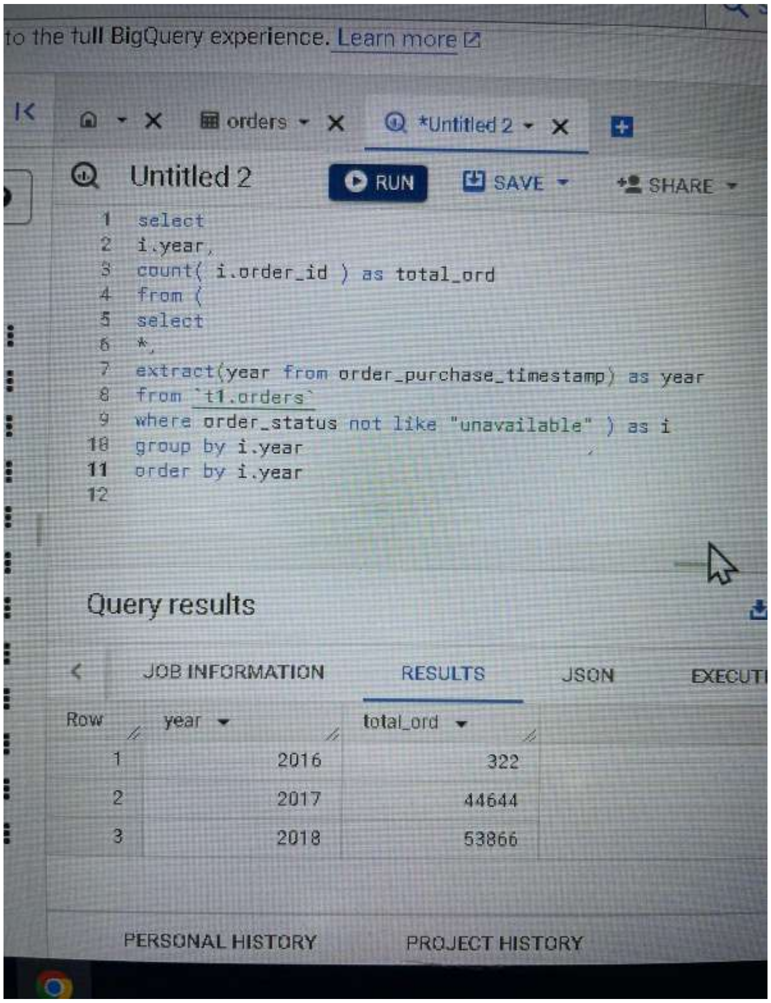
<u>Is there a **growing trend** in the number of **orders placed** over the **past few years?**</u>

1. First, we will filter the data based on the order status

2. Then we will create the years column from the order_timestamp

3. Now we will group the data on year

4. Then we will get the count of orders



```
select
i.year,
count( i.order_id ) as total_ord
from (
select
*,
extract(year from order_purchase_timestamp) as year
from `t1.orders`
where order_status not like "unavailable" ) as i
group by i.year
order by i.year
```

Row	year	total_ord
1	2016	322
2	2017	44644
3	2018	53866

Output:

1. 2016: 322
2. 2017: 44644
3. 2018: 53866

Observation:

1. **Yes, there is a growing trend compared to previous years**
2. **As we can see, customers orders placed in 2018 were higher than the previous year**
3. **But we only have two months of records for year 2016, we cannot compare 2017's record with 2016's**
4. **But for 2018, yes, we have positive signs for the business**

Can we see some kind of **monthly seasonality** in terms of the number of **orders being placed?**

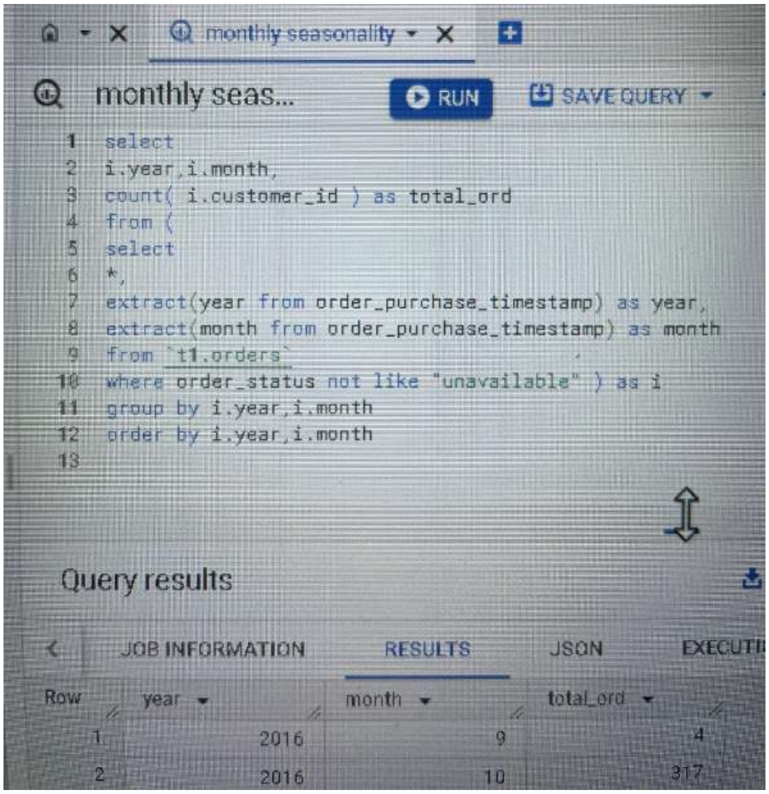
1. First, we will filter the data based on the order status

2. Then we will create a year column from the order_timestamp.

3. Then we will create the month column from the order_timestamp.

4. Now we will group the data by year and month.

5. Then count the customers orders month-wise

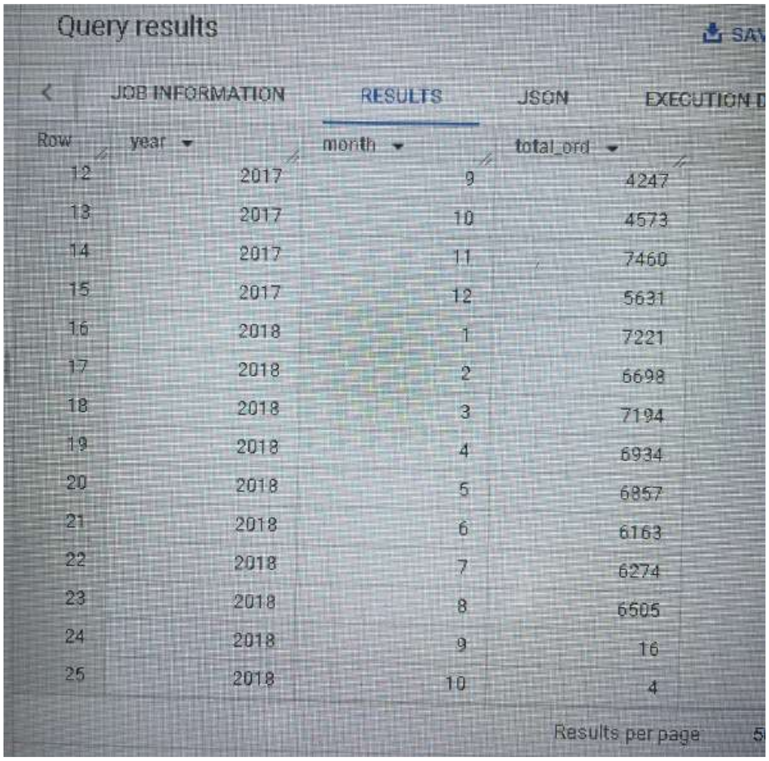


```
select
i.year,i.month,
count( i.customer_id ) as total_ord
from (
select
*,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month
from `t1.orders`
where order_status not like "unavailable" ) as i
group by i.year,i.month
order by i.year,i.month
```

Query results

Row	year	month	total_ord
1	2016	9	4
2	2016	10	317

output:


Query results

Row	year	month	total_ord
12	2017	9	4247
13	2017	10	4573
14	2017	11	7460
15	2017	12	5631
16	2018	1	7221
17	2018	2	6698
18	2018	3	7194
19	2018	4	6934
20	2018	5	6857
21	2018	6	6163
22	2018	7	6274
23	2018	8	6505
24	2018	9	16
25	2018	10	4

Observation:

1. **Here we can see that in 2017, as we move forward month-wise, the total number of orders placed is increasing.**

2. **But for 2018, the case is not the same; through-out the year, the total number of orders placed were in the range of 6,500–7,500 for almost all months**

3. **Hence,we cannot say there is some kind of monthly seasonality in terms of orders placed**

During what **time of the day** do Brazilian customers mostly **place their orders?**

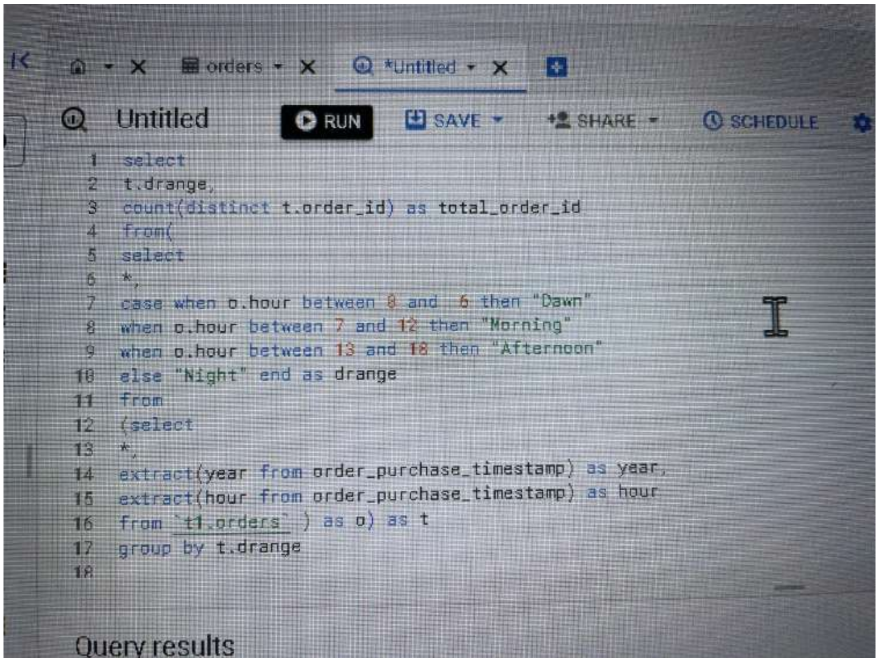
1. First, we create an "hour" column from order_timestamp."

2. Then we will categorise this "hour" column into four sections

i.e., 1. Morning, 2. Dawn, 3. Afternoon, 4. Night

3. Then we will group by the data on these four segments

4. Count the Total order

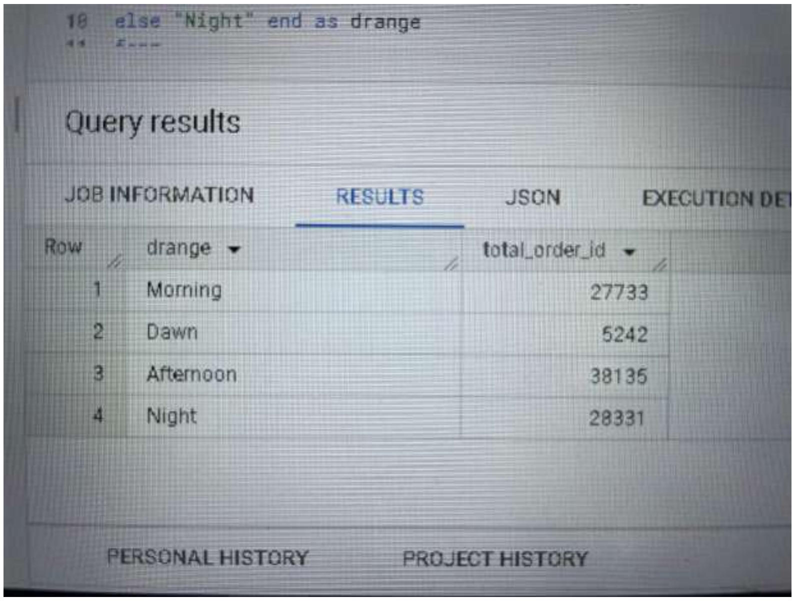


```sql
select
t.drange,
count(distinct t.order_id) as total_order_id
from(
select
*,
case when o.hour between 0 and  6 then "Dawn"
when o.hour between 7 and 12 then "Morning"
when o.hour between 13 and 18 then "Afternoon"
else "Night" end as drange
from
(select
*,
extract(year from order_purchase_timestamp) as year,
extract(hour from order_purchase_timestamp) as hour
from `t1.orders` ) as o) as t
group by t.drange

```

Query results

output.



```sql
else "Night" end as drange
```

Query results

Row	drange	total_order_id
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

As we can see, the Query output is:

1. Morning : 27733
2. Dawn: 5242
3. Afternoon: 38135

4. Night: 28331

Observation:

- **Brazilian people are less likely to order while Dawn is on, as people might be sleeping or less active on phone.**
- **Whereas in afternoon, Brazilian people do place orders more as compared to other ranges**
- **Morning and night have almost same orders**
- Hence, **Afternoon time** period is a favourite time for Brazilians.

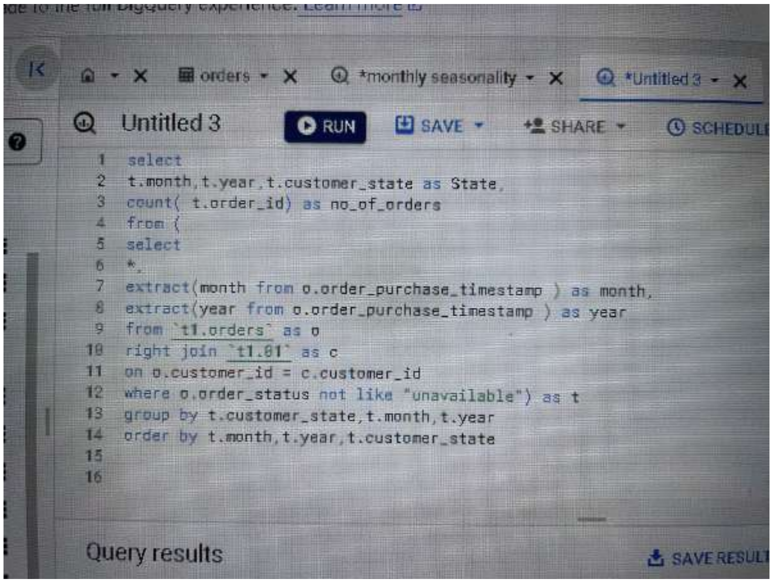
Approach 3

Evolution of E-commerce orders in the Brazil region

Get the month-on-month number of orders placed in each state

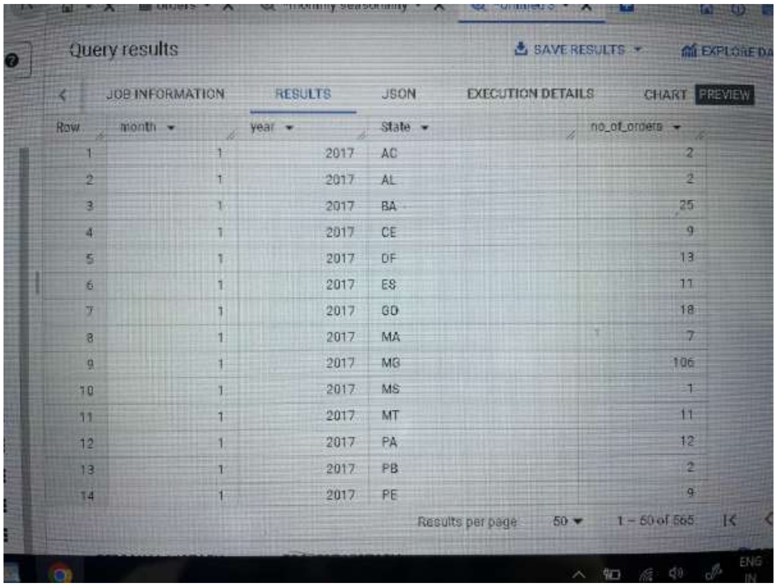
1. First, we will create a "Month" and "Year" column from order_purchase_timestamp.
2. Then we will join two tables, i.e. "order" table and the "customer" table, as "order" table has order details and "customer" table has state information of each customer
3. After that, we will group by month, year and state

4. Get the Count of order IDs within the group by



```
select
t.month,t.year,t.customer_state as State,
count( t.order_id) as no_of_orders
from (
select
*,
extract(month from o.order_purchase_timestamp ) as month,
extract(year from o.order_purchase_timestamp ) as year
from `t1.orders` as o
right join `t1.01` as c
on o.customer_id = c.customer_id
where o.order_status not like "unavailable") as t
group by t.customer_state,t.month,t.year
order by t.month,t.year,t.customer_state
```

RESULT:



Row	month	year	State	no_of_orders
1	1	2017	AC	2
2	1	2017	AL	2
3	1	2017	BA	25
4	1	2017	CE	9
5	1	2017	DF	13
6	1	2017	ES	11
7	1	2017	GO	18
8	1	2017	MA	7
9	1	2017	MG	106
10	1	2017	MS	1
11	1	2017	MT	11
12	1	2017	PA	12
13	1	2017	PB	2
14	1	2017	PE	9

Results per page 50 1 – 50 of 565

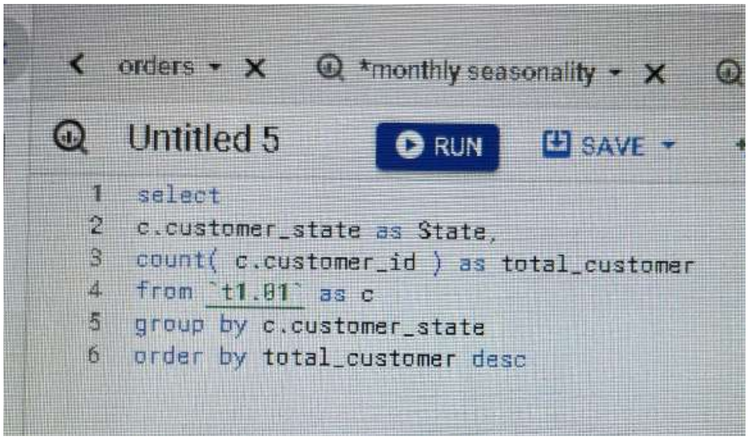
Observation:

1. **For each month and year-wise for each state, we got the Total order placed**
2. **A total of 565 raws were obtained.**

How are the customers distributed across all the states?

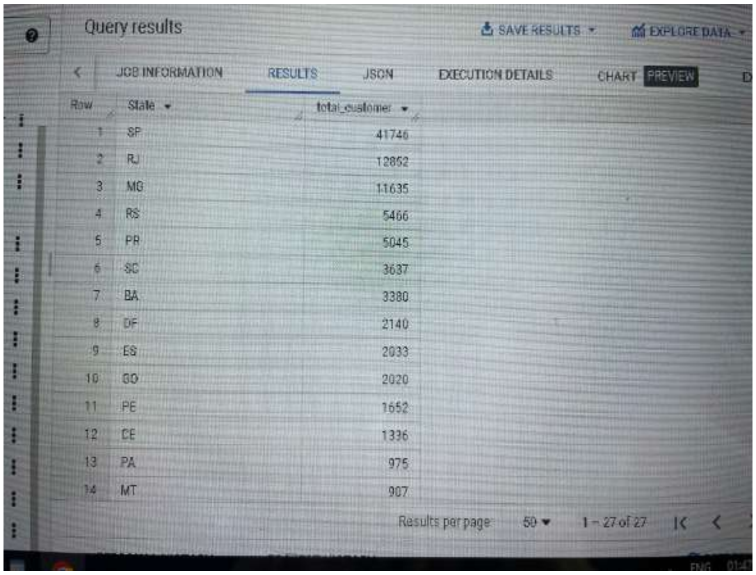
1. From the "customer" table, we will group by State

2. Get the total count of customer IDs.



```
select
c.customer_state as State,
count( c.customer_id ) as total_customer
from `t1.01` as c
group by c.customer_state
order by total_customer desc
```

RESULT:



Row	State	total_customer
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336
13	PA	975
14	MT	907

Results per page: 50 1 – 27 of 27

Observation:

1. **There are total 27 states where our customers are distributed**
2. **The state with the most customers is SP, with total 41746 unique customers**

3. **The state with the lowest customers is RR, with total 46 customers**

Suggestion:

1. **Our marketing campaigns can be more frequent in states with fewer customers, so that more customers can add**
2. **States with higher customer numbers tend to have more mouth-to-mouth publicity so we can keep marketing efforts constant instead of increasing**

Approach 4

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

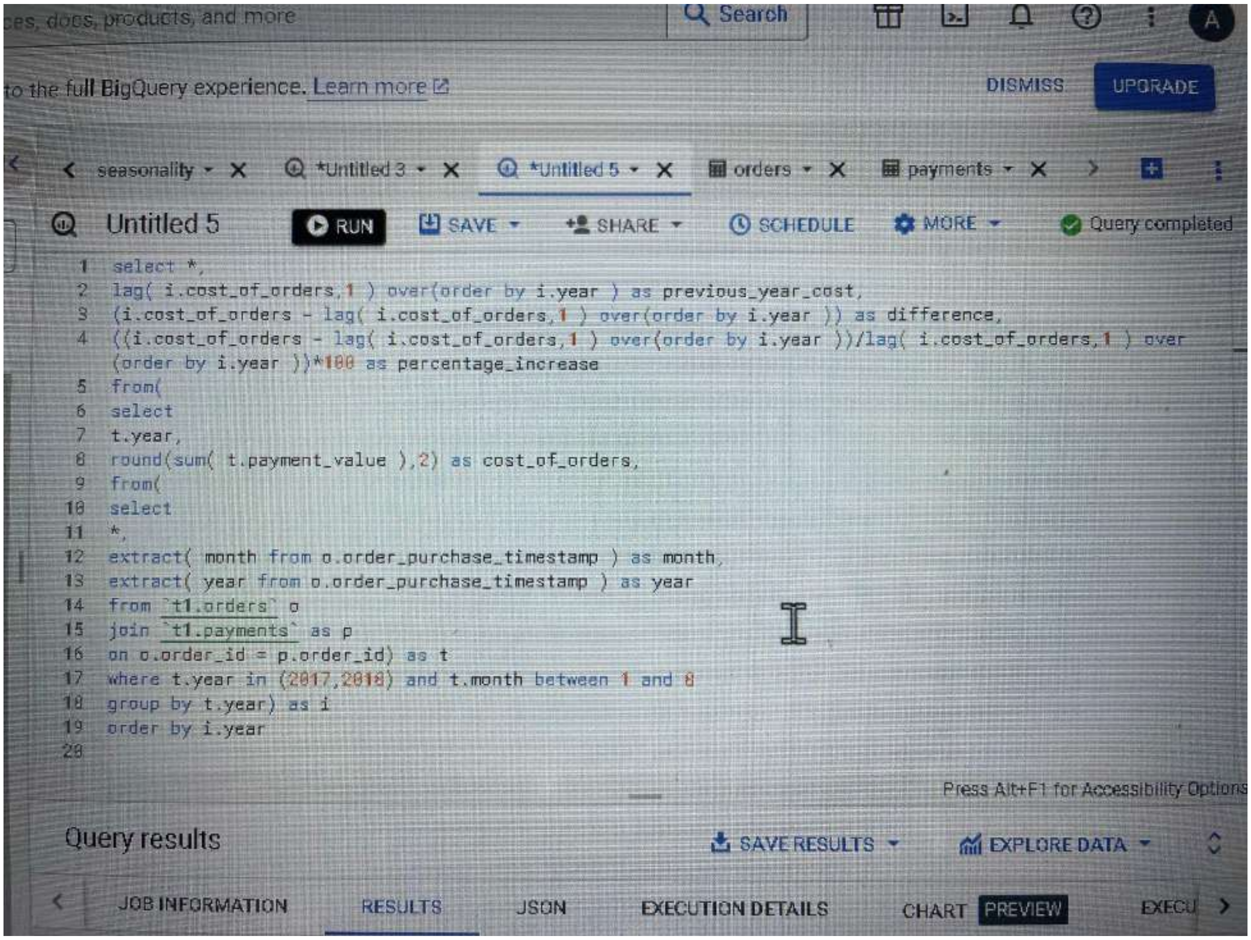
<u>Get the percentage increase in the cost of orders from 2017 to 2018 (include months between January and August only).</u>

1. We will join "order" table and "payment" table

2. Then we'll filter based on months, i.e. between "jan-aug" and for year 2017-18

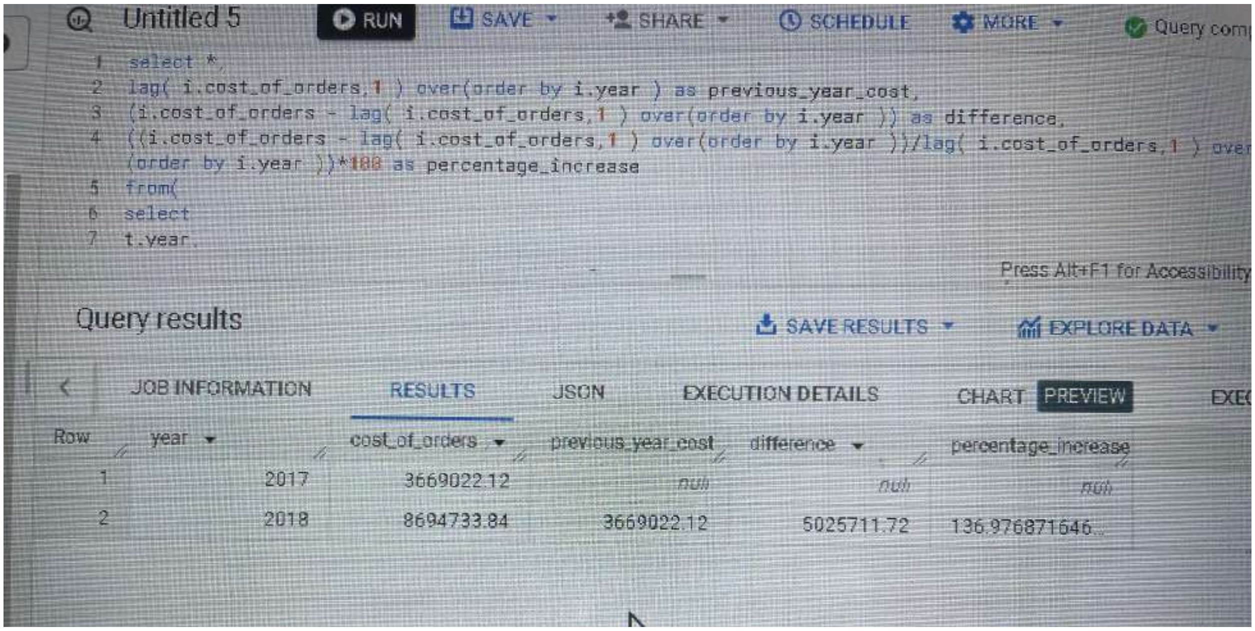
3. After that, we'll goup by on year to get the sum of payment_value as cost of order for a year

4. Applying the window function in the table will get the desired value, i.e., a % increase in the cost of orders from 2017 to 2018.



```
select *,
lag( i.cost_of_orders,1 ) over(order by i.year ) as previous_year_cost,
(i.cost_of_orders - lag( i.cost_of_orders,1 ) over(order by i.year )) as difference,
((i.cost_of_orders - lag( i.cost_of_orders,1 ) over(order by i.year ))/lag( i.cost_of_orders,1 ) over
(order by i.year ))*100 as percentage_increase
from(
select
t.year,
round(sum( t.payment_value ),2) as cost_of_orders,
from(
select
*,
extract( month from o.order_purchase_timestamp ) as month,
extract( year from o.order_purchase_timestamp ) as year
from `t1.orders` o
join `t1.payments` as p
on o.order_id = p.order_id) as t
where t.year in (2017,2018) and t.month between 1 and 8
group by t.year) as i
order by i.year
```

RESULT:



```
select *,
lag( i.cost_of_orders,1 ) over(order by i.year ) as previous_year_cost,
(i.cost_of_orders - lag( i.cost_of_orders,1 ) over(order by i.year )) as difference,
((i.cost_of_orders - lag( i.cost_of_orders,1 ) over(order by i.year ))/lag( i.cost_of_orders,1 ) ove
(order by i.year ))*100 as percentage_increase
from(
select
t.year,
```

Row	year	cost_of_orders	previous_year_cost	difference	percentage_increase
1	2017	3669022.12	null	null	null
2	2018	8694733.84	3669022.12	5025711.72	136.976871646...

OUTPUT:

Year	cost_of_orders	Previous_year_cost	difference	percentage_increased
2017:	3669022.12	null	null	null
2018:	8694733.84	3669022.12	5025711.72	136.976

Observation:

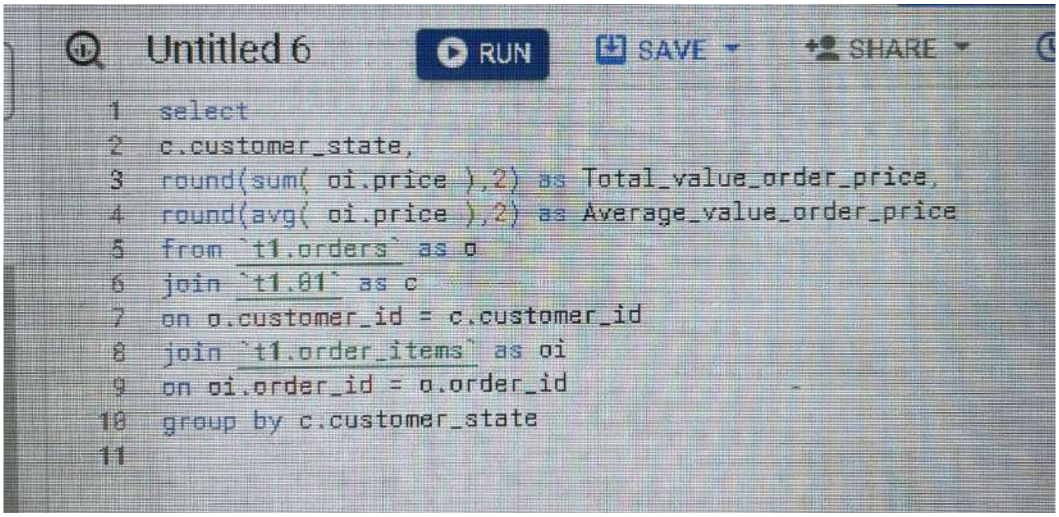
1. **The percent increase in the cost of orders from 2017 to 2018 is "136.97%."**

Calculate the Total and Average value of the order price for each state.

1. We'll merge the three tables, i.e. the "orders" table, "customer" table, and the "order_item" table

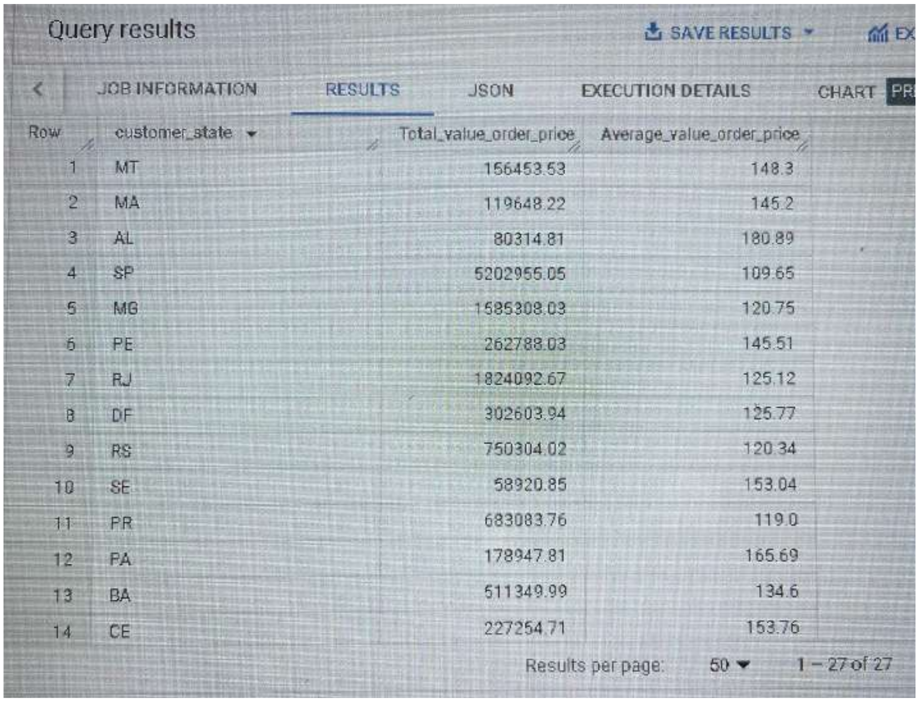
2. Then we will group by on States

3. We'll get the average price and Total price over states.



```
select
c.customer_state,
round(sum( oi.price ),2) as Total_value_order_price,
round(avg( oi.price ),2) as Average_value_order_price
from `t1.orders` as o
join `t1.01` as c
on o.customer_id = c.customer_id
join `t1.order_items` as oi
on oi.order_id = o.order_id
group by c.customer_state
```

RESULT:



Row	customer_state	Total_value_order_price	Average_value_order_price
1	MT	156453.53	148.3
2	MA	119648.22	145.2
3	AL	80314.81	180.89
4	SP	5202955.05	109.65
5	MG	1585308.03	120.75
6	PE	262788.03	145.51
7	RJ	1824092.67	125.12
8	DF	302603.94	125.77
9	RS	750304.02	120.34
10	SE	58920.85	153.04
11	PR	683083.76	119.0
12	PA	178947.81	165.69
13	BA	511349.99	134.6
14	CE	227254.71	153.76

Observation:

1. We got total 27 states

2. Where "SP" state has the highest total value of order price for each state

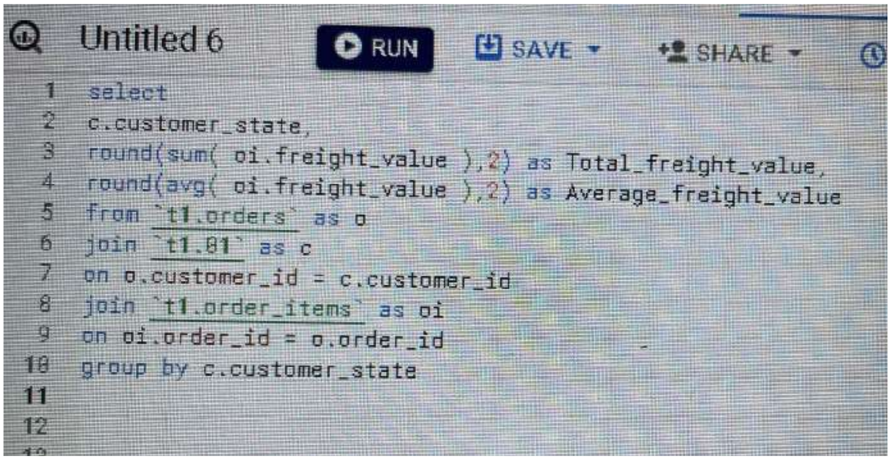
3. Where "PB" state has the highest Average value of order price for each state,

Calculate the Total and Average value of order freight for each state

1. We'll merge the three tables, i.e. the "orders" table, "customer" table, and the "order_item" table

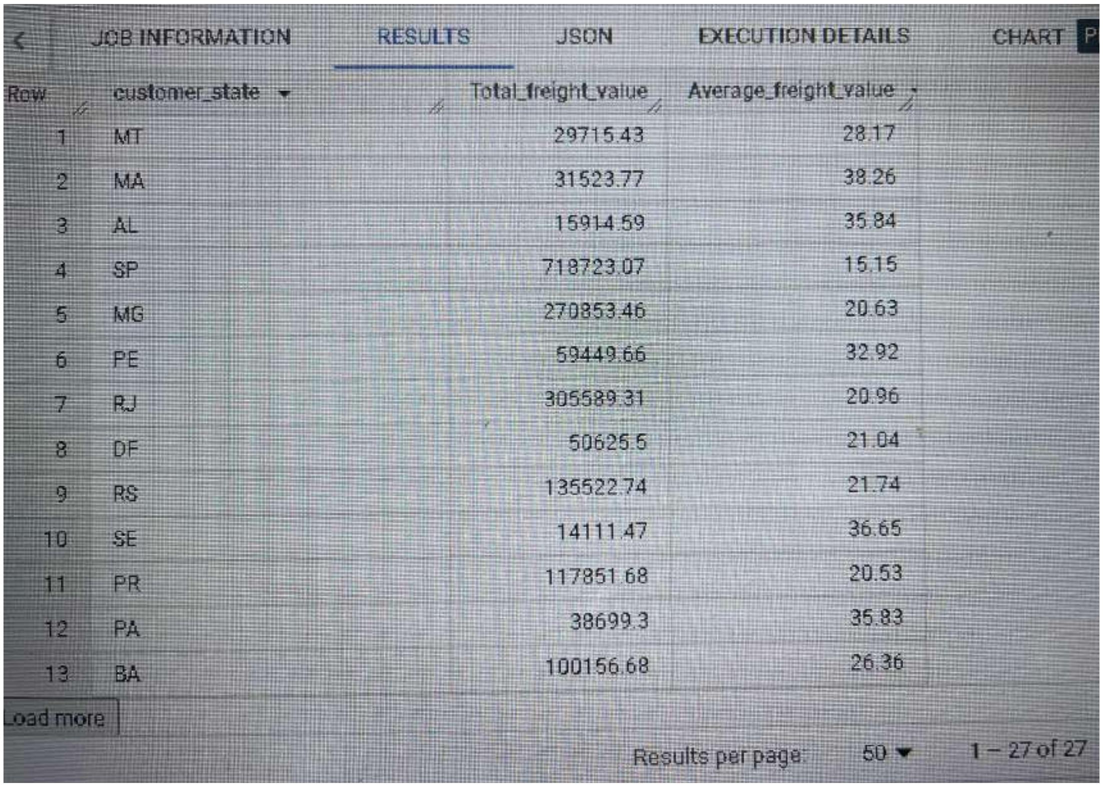
2. Then we will group by on States

3. We'll get the Total and Average value of order freight for each state.



```
select
c.customer_state,
round(sum( oi.freight_value ),2) as Total_freight_value,
round(avg( oi.freight_value ),2) as Average_freight_value
from `t1.orders` as o
join `t1.01` as c
on o.customer_id = c.customer_id
join `t1.order_items` as oi
on oi.order_id = o.order_id
group by c.customer_state
```

RESULT:



Row	customer_state	Total_freight_value	Average_freight_value
1	MT	29715.43	28.17
2	MA	31523.77	38.26
3	AL	15914.59	35.84
4	SP	718723.07	15.15
5	MG	270853.46	20.63
6	PE	59449.66	32.92
7	RJ	305589.31	20.96
8	DF	50625.5	21.04
9	RS	135522.74	21.74
10	SE	14111.47	36.65
11	PR	117851.68	20.53
12	PA	38699.3	35.83
13	BA	100156.68	26.36

Observation:

1. We got total 27 states

2. Where "SP" state has the highest total value of order freight for each state,

3. Where "RR" states havethe the highest average value of order freight for each state,

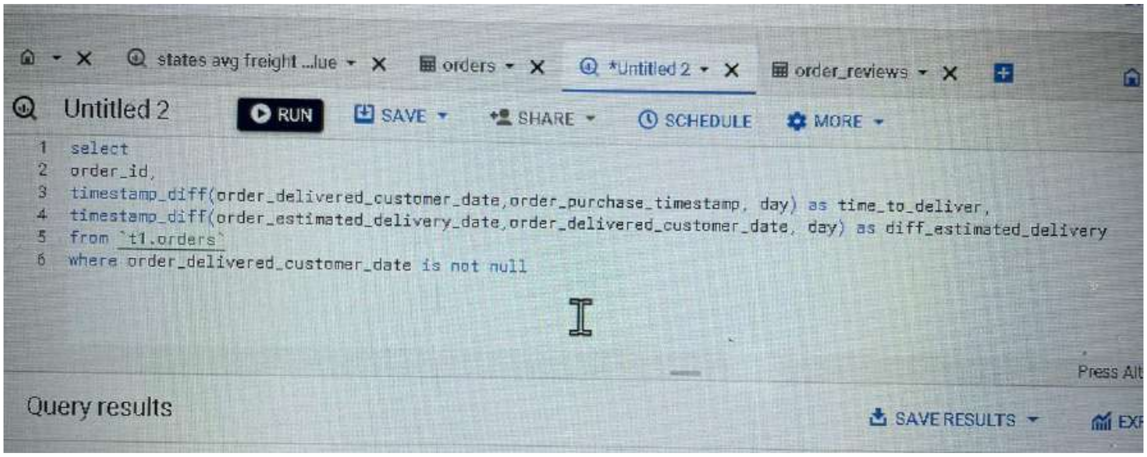
Approach 5

Analysis based on sales, freight and delivery time

Find the number of days taken to deliver each order from the order's purchase date as delivery time.

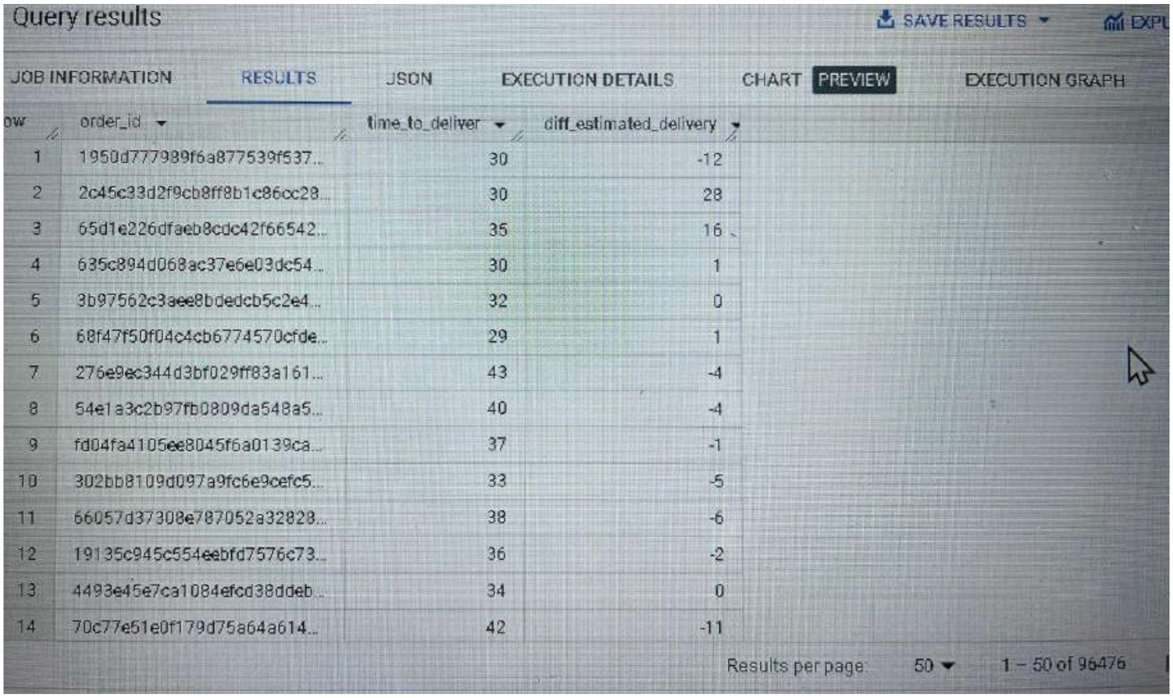
Also, the difference (in days) between the estimated and actual delivery dates of an order

1. From the order table, we will use timestamp_diff function on the columns order_delivered_customer_date and order_purchase_timestamp then fetch day as time_to_deliver
2. Similarly, we will use timestamp_diff function on order_estimated_delivery_date and order_delivered_customer_date then fetch day as diff_estimated_delivery



```
select
order_id,
timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_deliver,
timestamp_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as diff_estimated_delivery
from `t1.orders`
where order_delivered_customer_date is not null
```

Result:



ow	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f537...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54...	30	1
5	3b97562c3aee8bdedcb5c2e4...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161...	43	-4
8	54e1a3c2b97fb0809da548a5...	40	-4
9	fd04fa4105ee8045f6a0139ca...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5
11	66057d37308e787052a32828...	38	-6
12	19135c945c554eebfd7576c73...	36	-2
13	4493e45e7ca1084efcd38ddeb...	34	0
14	70c77e51e0f179d75a64a614...	42	-11

Results per page: 50 ▾ 1 – 50 of 96476

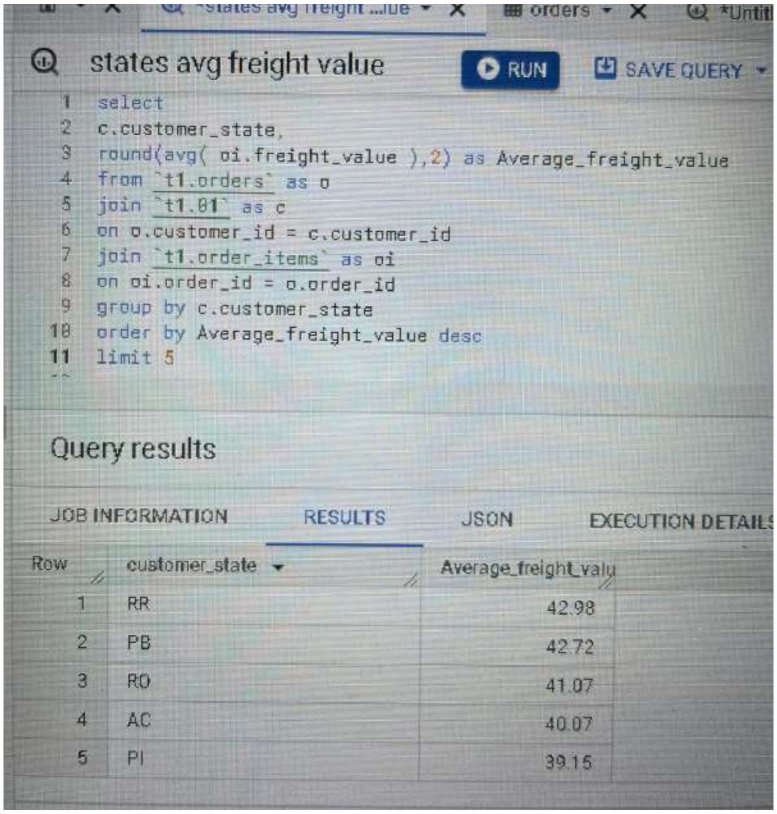
Observation:

1. **Order_id** ca07593549f1816d26a572e06dc took maximum number of days to deliver, i.e., 209 days

2. Order_id 0607f0efea4b566f1eb8f7d3c2397320 delivered 146 days prior the estimated date

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

1. Top 5 states with highest average freight value:



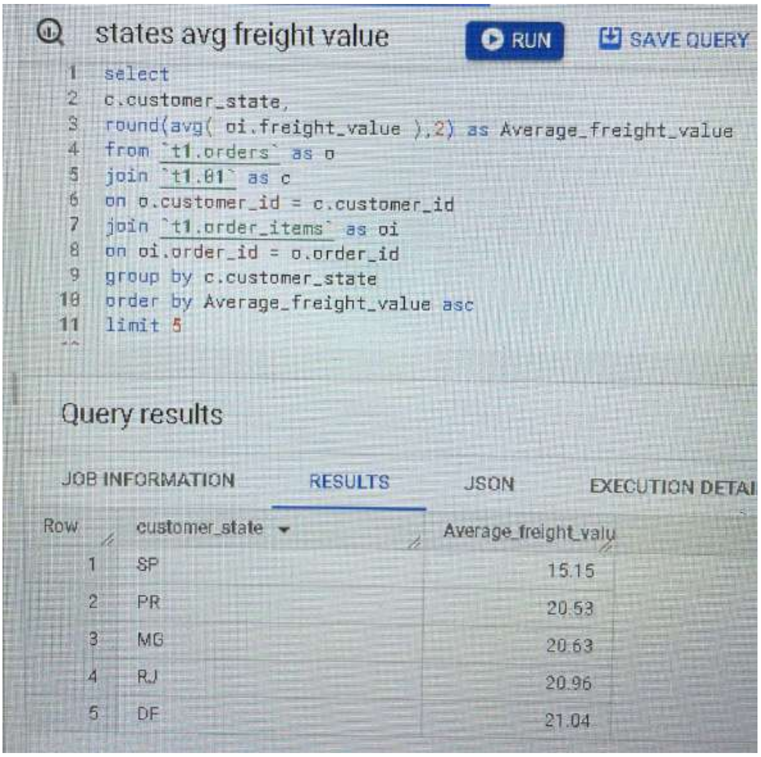
states avg freight value

```
select
c.customer_state,
round(avg( oi.freight_value ),2) as Average_freight_value
from `t1.orders` as o
join `t1.01` as c
on o.customer_id = c.customer_id
join `t1.order_items` as oi
on oi.order_id = o.order_id
group by c.customer_state
order by Average_freight_value desc
limit 5
```

Query results

Row	customer_state	Average_freight_valu
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

2. Top 5 states with lowest average freight value:



states avg freight value

```
select
c.customer_state,
round(avg( oi.freight_value ),2) as Average_freight_value
from `t1.orders` as o
join `t1.01` as c
on o.customer_id = c.customer_id
join `t1.order_items` as oi
on oi.order_id = o.order_id
group by c.customer_state
order by Average_freight_value asc
limit 5
```

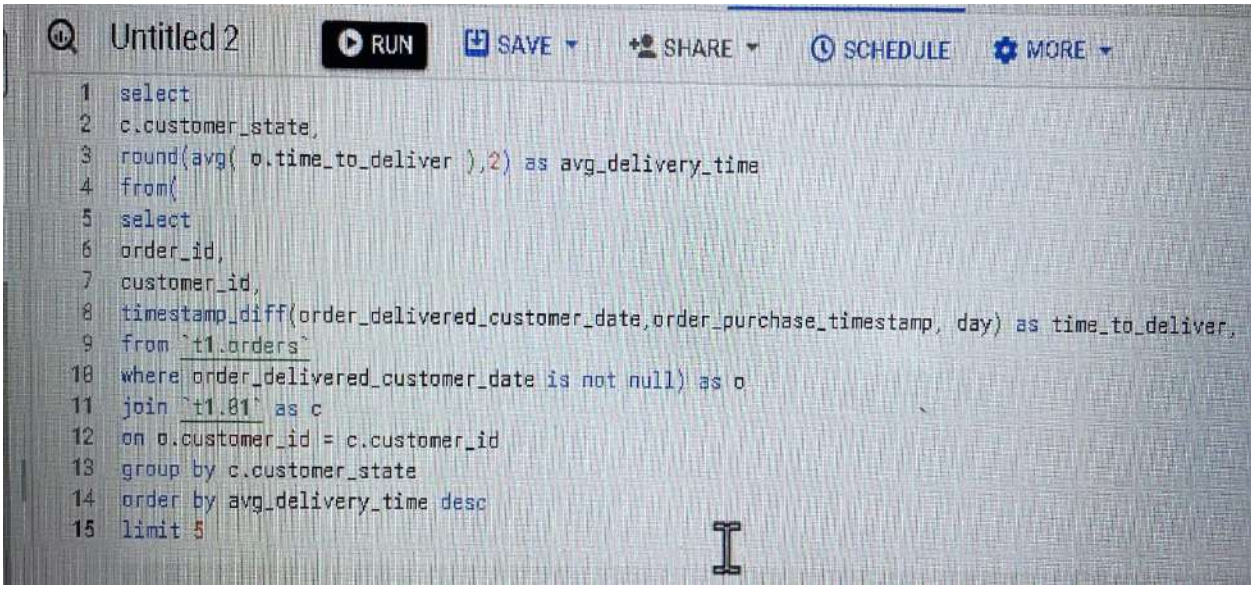
Query results

Row	customer_state	Average_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

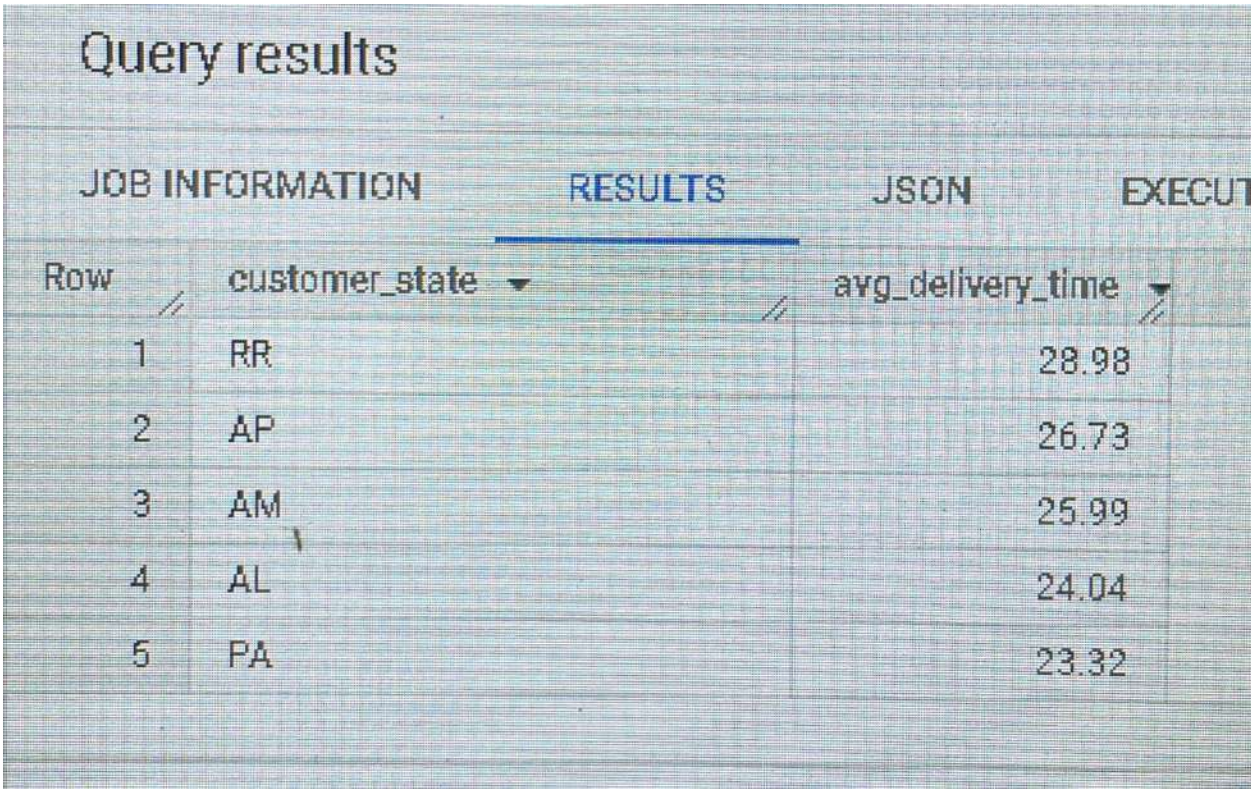
-

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

1. Top 5 states with the highest average delivery time.

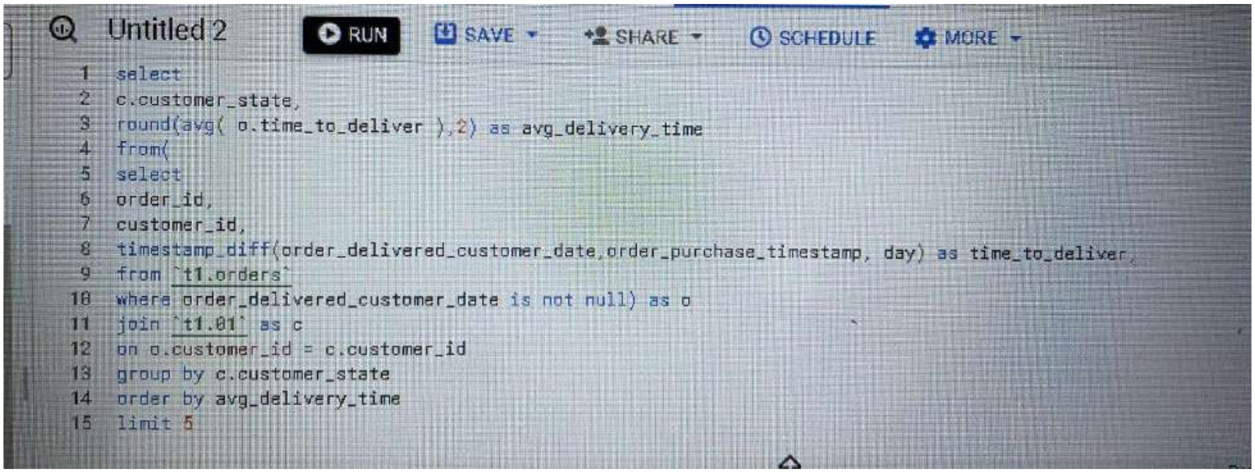


```
select
c.customer_state,
round(avg( o.time_to_deliver ),2) as avg_delivery_time
from(
select
order_id,
customer_id,
timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_deliver,
from `t1.orders`
where order_delivered_customer_date is not null) as o
join `t1.01` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_delivery_time desc
limit 5
```

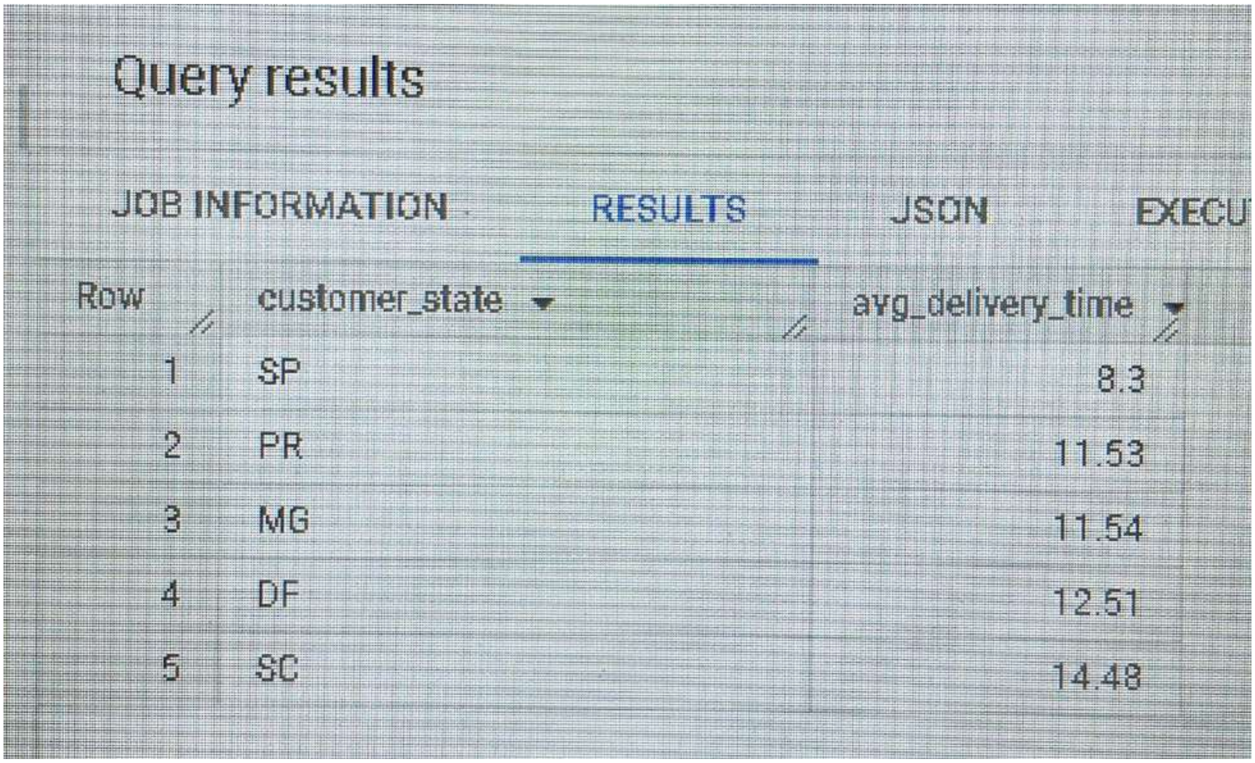


Row	customer_state	avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

2. Top 5 states with the lowest average delivery time



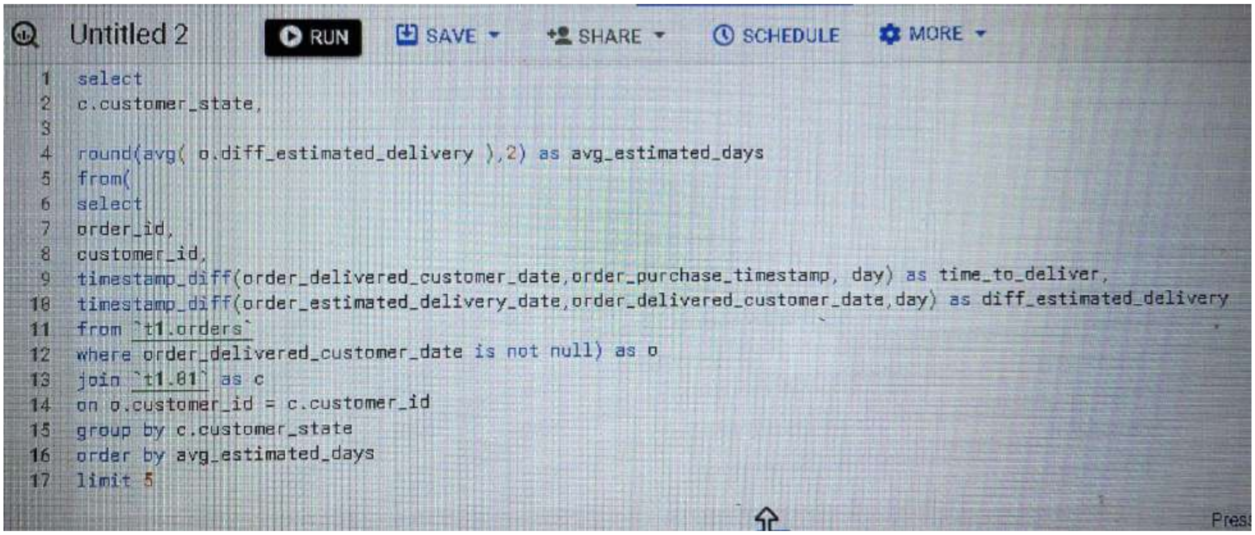
```
select
c.customer_state,
round(avg( o.time_to_deliver ),2) as avg_delivery_time
from(
select
order_id,
customer_id,
timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_deliver,
from `t1.orders`
where order_delivered_customer_date is not null) as o
join `t1.01` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_delivery_time
limit 5
```



Query results

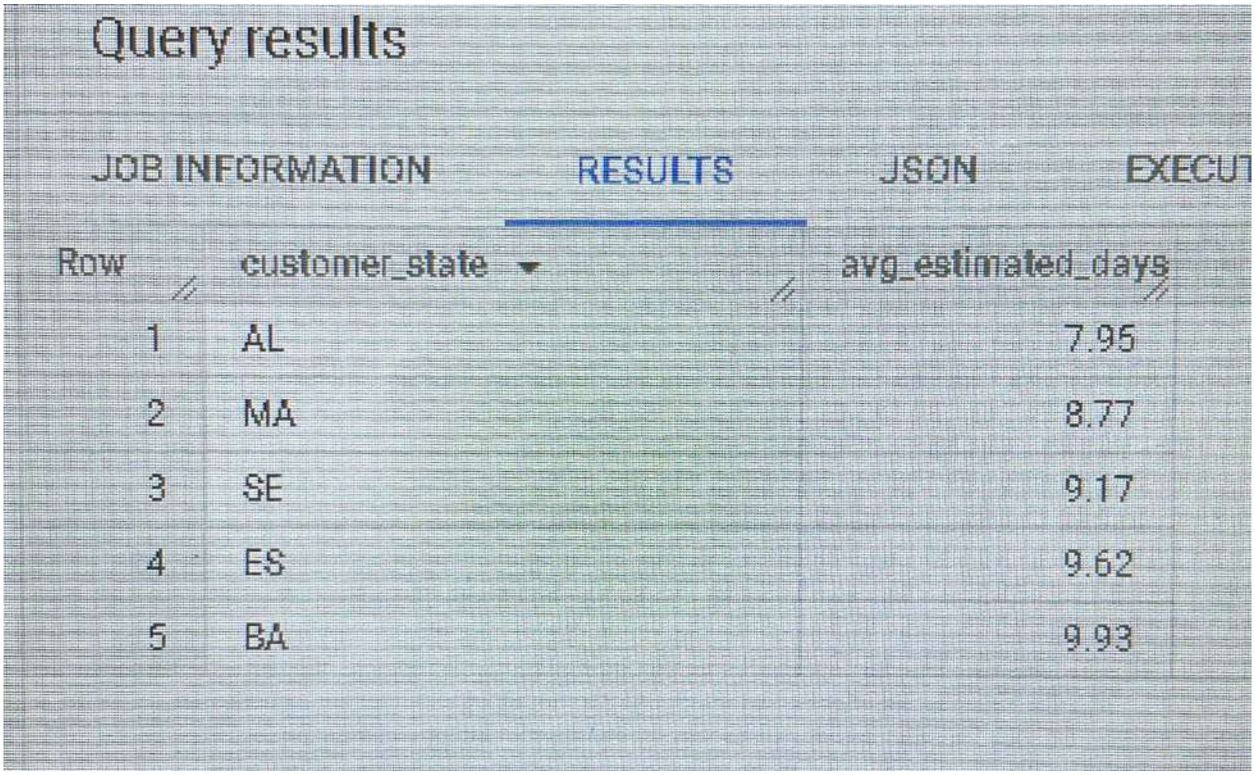
Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

<u>Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.</u>



```sql
select
c.customer_state,

round(avg( o.diff_estimated_delivery ),2) as avg_estimated_days
from(
select
order_id,
customer_id,
timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_deliver,
timestamp_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as diff_estimated_delivery
from `t1.orders`
where order_delivered_customer_date is not null) as o
join `t1.01` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by avg_estimated_days
limit 5
```

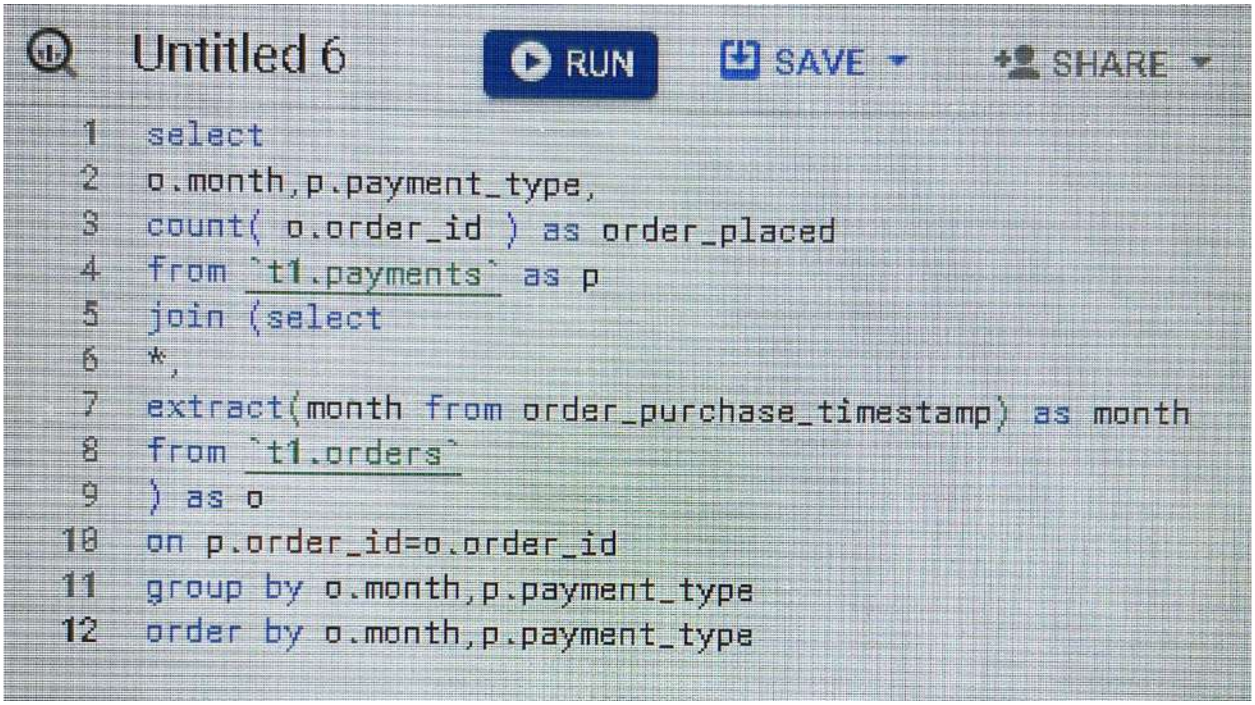


Row	customer_state	avg_estimated_days
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Approach 6

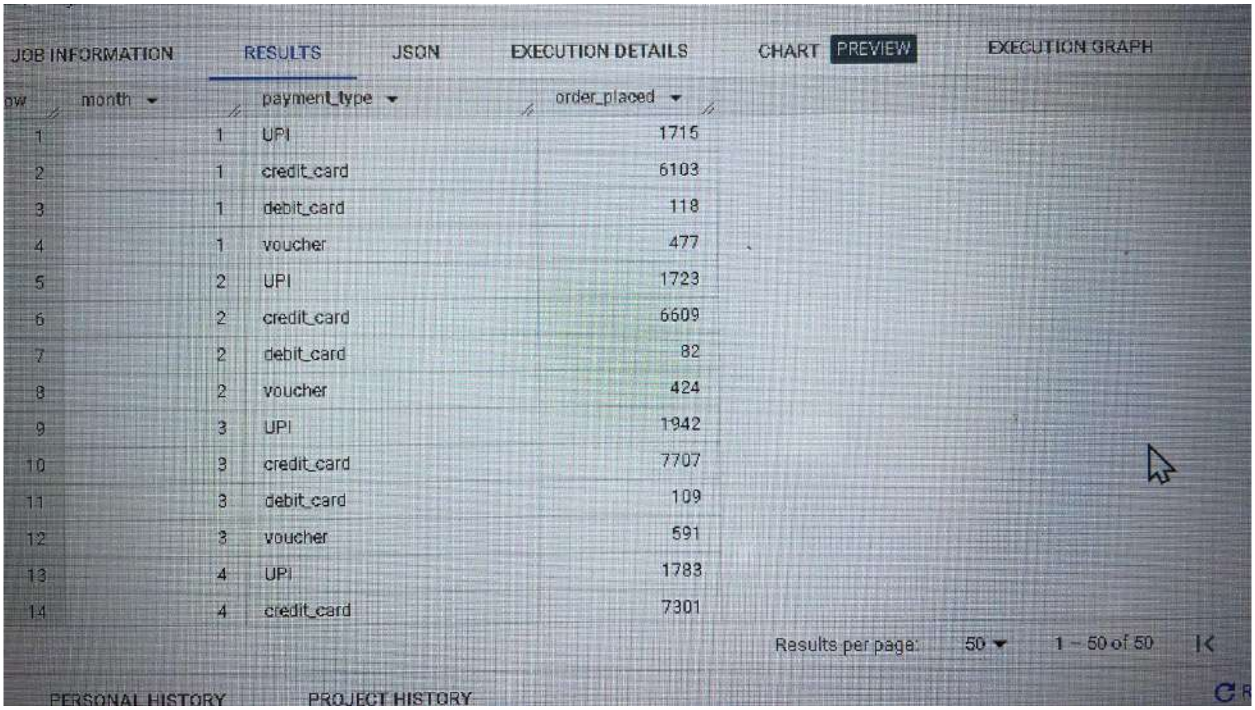
Analysis based on the payments

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



```
select
o.month,p.payment_type,
count( o.order_id ) as order_placed
from `t1.payments` as p
join (select
*,
extract(month from order_purchase_timestamp) as month
from `t1.orders`
) as o
on p.order_id=o.order_id
group by o.month,p.payment_type
order by o.month,p.payment_type
```

Result:



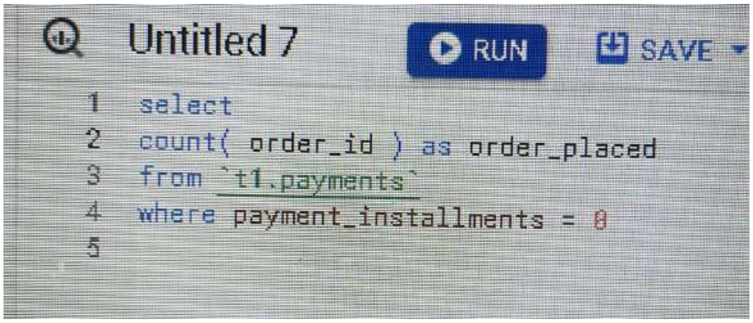
ow	month	payment_type	order_placed
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707
11	3	debit_card	109
12	3	voucher	591
13	4	UPI	1783
14	4	credit_card	7301

Results per page: 50 1 – 50 of 50

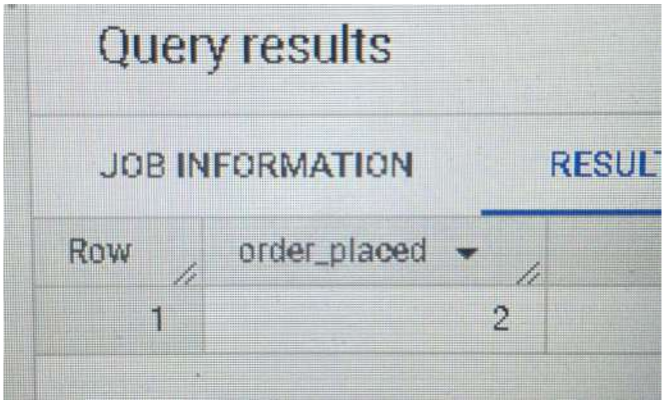
Observation:

1. There are basically five different payment types
2. 1. UPI, 2. Credit card, 3. Debit card, 4. Voucher, 5. Not-defined
3. We got total 50 rows of data for 12 months

<u>Find the no. of orders placed on the basis of the payment installments that have been paid.</u>



```
select
count( order_id ) as order_placed
from `t1.payments`
where payment_installments = 0
```



Row	order_placed
1	2

Observation:

1. There are two orders placed with no instalments

Actionable Insights and Recommendations

1. From Approach 1, we have found that customers tend to order more in Afternoon as compared to anyother time zone
2. So, our focus should be more on the three time ranges specifically, i.e., Afternoon, Morning and Night
3. From Approach 2, we saw that there is growing trend for number of orders placed last year vs current year, which shows a positive sign
4. And in the mid-months of year, there is a high number of orders placed but no sign of monthly seasonality

5. In Approach 3, we have seen how the customers are distributed across the state and on the basis of distribution we can analyse and focus more or less on the basic of number of customers in states
6. As cost of orders increased by more than 130% compared to last year, which is again a good sign.
7. As we got the top 5 states for average freight value,average delivery time and Fast states on order delivery compared to the estimated date of delivery, We can divert our services based of these results
8. In Approach 6, we learn that customers mostly use four types of payment, and the most widely used payment types are Credit cards and UPI.
9. So we can avail of more offers in such payment modes to promote more purchases.
10. And customers are more likely to purchase in installments, so we can focus more on such installment-related schemes.