Client: Retail store chain

Problem Statement: Business case focuses on the operations of retail store in Brazil about 100,000 orders placed between 2016 and 2018.

Why: To provide valuable insights into retail store'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.

Resources: Multiple datasets are provided like customers, sellers, order_items, geolocation, payments, reviews, orders, products.

The column description:

The **customers.csv** contain following features:

Features	**Description**
customer_id	ID of the consumer who made the purchase
customer_unique_id	Unique ID of the consumer
customer_zip_code_prefix	Zip Code of consumer’s location
customer_city	Name of the City from where order is made
customer_state	State Code from where order is made (Eg. são paulo - SP)

The **sellers.csv** contains following features:

Features	**Description**
seller_id	Unique ID of the seller registered
seller_zip_code_prefix	Zip Code of the seller’s location
seller_city	Name of the City of the seller
seller_state	State Code (Eg. são paulo - SP)

The **order_items.csv** contain following features:

Features	**Description**
order_id	A Unique ID of order made by the consumers
order_item_id	A Unique ID given to each item ordered in the order
product_id	A Unique ID given to each product available on the site
seller_id	Unique ID of the seller registered in Target
shipping_limit_date	The date before which the ordered product must be shipped
price	Actual price of the products ordered
freight_value	Price rate at which a product is delivered from one point to another

The **geolocations.csv** contain following features:

Features	**Description**
geolocation_zip_code_prefix	First 5 digits of Zip Code
geolocation_lat	Latitude
geolocation_lng	Longitude
geolocation_city	City
geolocation_state	State

The **payments.csv** contain following features:

Features	**Description**
order_id	A Unique ID of order made by the consumers
payment_sequential	Sequences of the payments made in case of EMI
payment_type	Mode of payment used (Eg. Credit Card)
payment_installments	Number of installments in case of EMI purchase
payment_value	Total amount paid for the purchase order

The **orders.csv** contain following features:

Features	**Description**
order_id	A Unique ID of order made by the consumers
customer_id	ID of the consumer who made the purchase
order_status	Status of the order made i.e. delivered, shipped, etc.
order_purchase_timestamp	Timestamp of the purchase
order_delivered_carrier_date	Delivery date at which carrier made the delivery
order_delivered_customer_date	Date at which customer got the product
order_estimated_delivery_date	Estimated delivery date of the products

The **reviews.csv** contain following features:

Features	**Description**
review_id	ID of the review given on the product ordered by the order id
order_id	A Unique ID of order made by the consumers
review_score	Review score given by the customer for each order on a scale of 1-5
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order
review_creation_date	Timestamp of the review when it is created
review_answer_timestamp	Timestamp of the review answered

The **products.csv** contain following features:

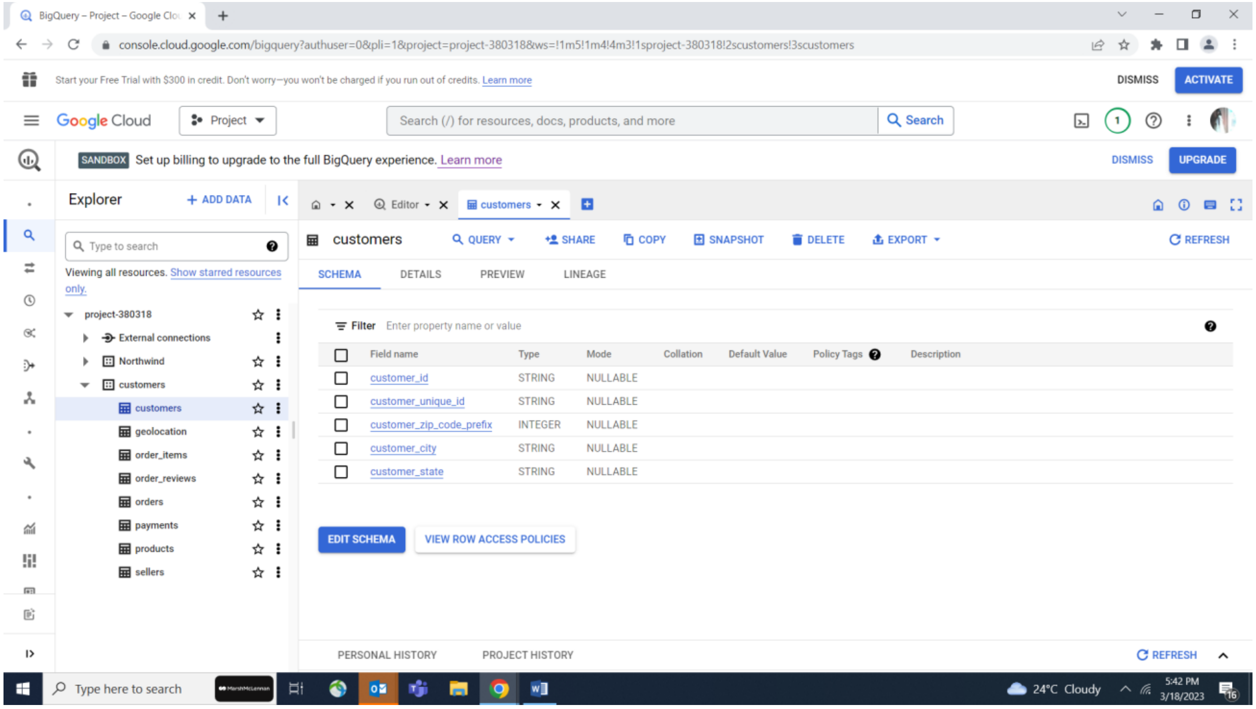
Features	Description
product_id	A Unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	Length of the string which specifies the name given to the products ordered
product_description_lenght	Length of the description written for each product ordered on the site
product_photos_qty	Number of photos of each product ordered available on the shopping portal
product_weight_g	Weight of the products ordered in grams
product_length_cm	Length of the products ordered in centimeters
product_height_cm	Height of the products ordered in centimeters
product_width_cm	Width of the product ordered in centimeters

1.1 Data type of columns in a table:

Explanation: I uploaded all CSVs on big query sandbox which we can see on left.

After clicking on each file, we can check their data type on right.

Example on below screenshots:

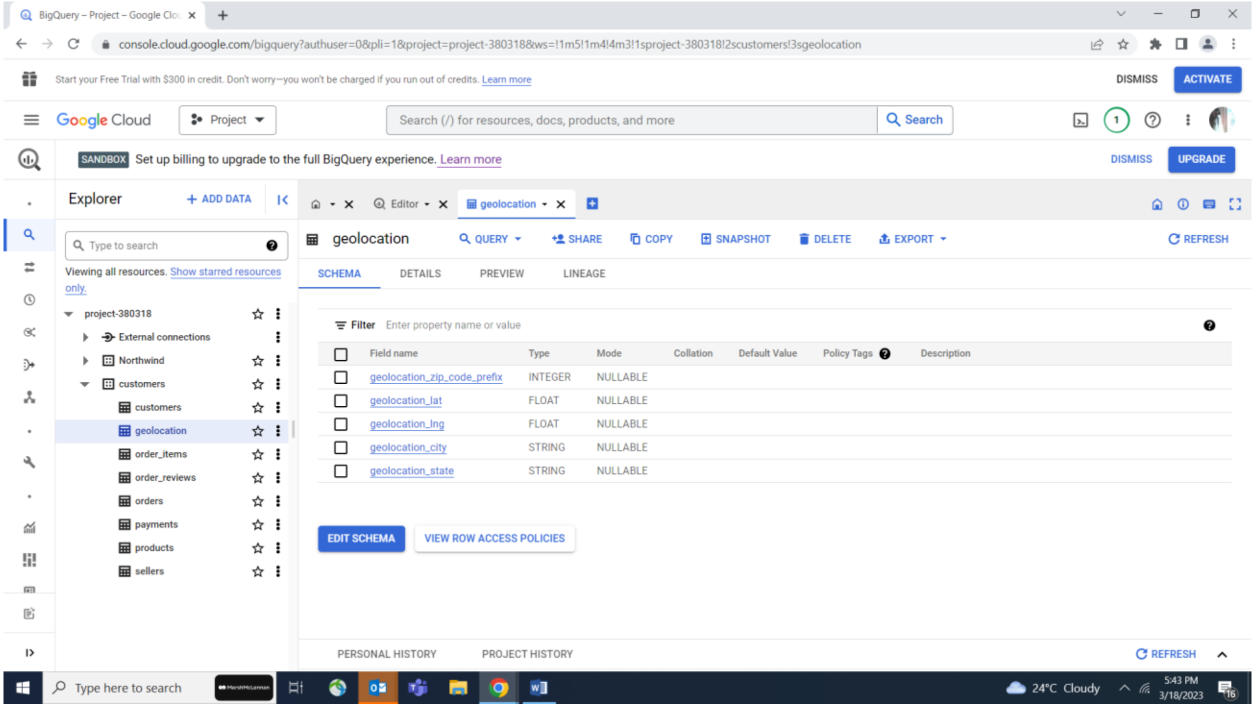


Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more

SANDBOX Set up billing to upgrade to the full BigQuery experience. Learn more

customers

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


BigQuery – Project – Google Clo
console.cloud.google.com/bigquery?authuser=0&pli=1&project=project-380318&ws=!1m5!1m4!4m3!1sproject-380318!2scustomers!3sgeolocation
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
DISMISS
ACTIVATE
Google Cloud
Project
Search (/) for resources, docs, products, and more
Search
SANDBOX
Set up billing to upgrade to the full BigQuery experience. Learn more
DISMISS
UPGRADE
Explorer
+ ADD DATA
Type to search
Viewing all resources. Show starred resources only.
project-380318
External connections
Northwind
customers
customers
geolocation
order_items
order_reviews
orders
payments
products
sellers
Editor
geolocation
geolocation
QUERY
SHARE
COPY
SNAPSHOT
DELETE
EXPORT
REFRESH
SCHEMA
DETAILS
PREVIEW
LINEAGE
Filter
Enter property name or value
EDIT SCHEMA
VIEW ROW ACCESS POLICIES
PERSONAL HISTORY
PROJECT HISTORY
REFRESH
Type here to search
24°C Cloudy
5:43 PM
3/18/2023

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
geolocation_zip_code_prefix	INTEGER	NULLABLE				
geolocation_lat	FLOAT	NULLABLE				
geolocation_lng	FLOAT	NULLABLE				
geolocation_city	STRING	NULLABLE				
geolocation_state	STRING	NULLABLE				

1.2 Time period for which the data is given:

Explanation:

This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil i.e. from **2016-09-04 to 2018-10-17**

Query:

➢ To see first date:

```
SELECT extract(year from order_purchase_timestamp) as year,extract(month from order_purchase_timestamp) month, date(order_purchase_timestamp) as date
 FROM `project-380318.customers.orders`
order by year ,month, date
limit 1
```

➢ To see last date:

```
SELECT extract(year from order_purchase_timestamp) as year,extract(month from order_purchase_timestamp) month, date(order_purchase_timestamp) as date
FROM `project-380318.customers.orders`
order by year desc,month desc,date desc
limit 1
```

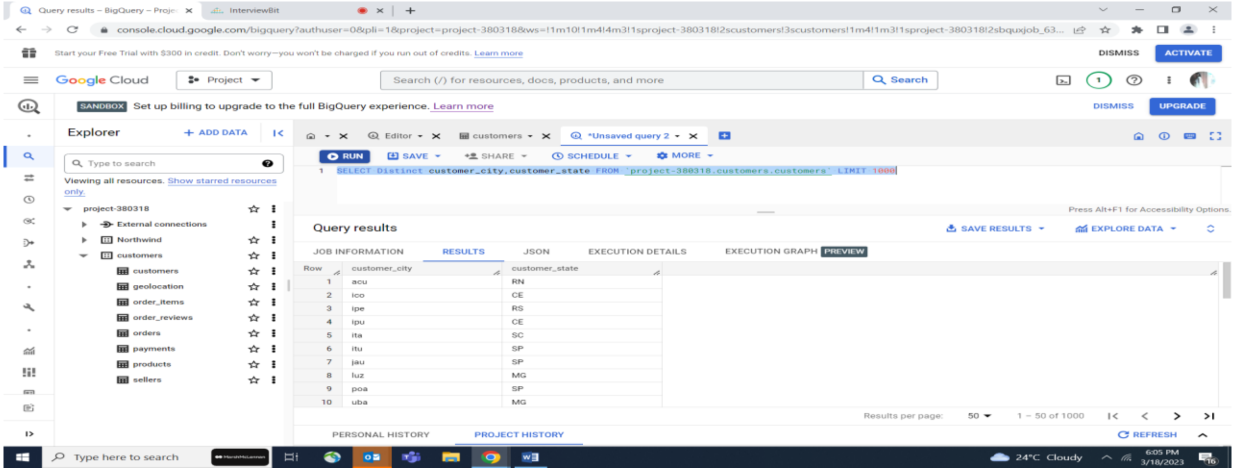
1.3 Cities and States of customers ordered during the given period:

Explanation:

Query:

```
SELECT Distinct customer_city,customer_state
FROM `project-380318.customers.customers`
```

Screenshot for few lines:



```
SELECT Distinct customer_city,customer_state FROM `project-380318.customers.customers` LIMIT 1000
```

Query results

Row	customer_city	customer_state
1	acu	RN
2	ico	CE
3	ipe	RS
4	ipu	CE
5	ita	SC
6	itu	SP
7	jau	SP
8	luz	MG
9	poa	SP
10	uba	MG

Results per page: 50 1 – 50 of 1000

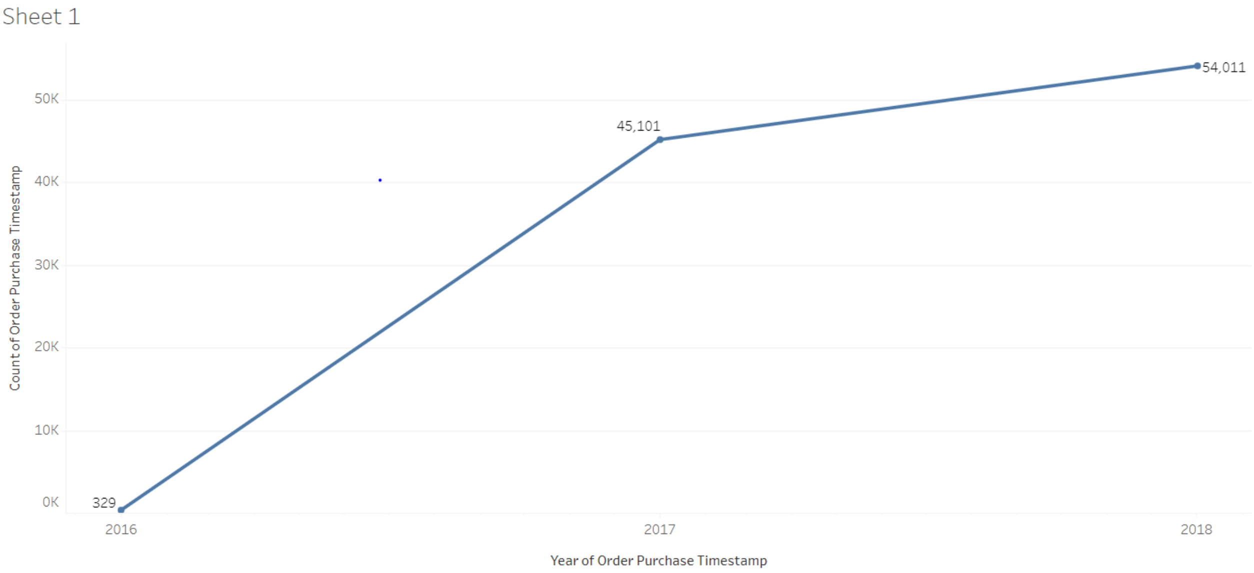
Query results – BigQuery – Proje
InterviewBit
console.cloud.google.com/bigquery?authuser=0&pli=1&project=project-380318&ws=!1m10!1m4!4m3!1sproject-380318!2scustomers!3scustomers!1m4!1m3!1sproject-380318!2sbquxjob_63...
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
DISMISS
ACTIVATE
Google Cloud
Project
Search (/) for resources, docs, products, and more
Search
SANDBOX
Set up billing to upgrade to the full BigQuery experience. Learn more
DISMISS
UPGRADE
Explorer
+ ADD DATA
Editor
customers
*Unsaved query 2
RUN
SAVE
SHARE
SCHEDULE
MORE
Type to search
Viewing all resources. Show starred resources only.
project-380318
External connections
Northwind
customers
customers
geolocation
order_items
order_reviews
orders
payments
products
sellers
Press Alt+F1 for Accessibility Options.
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
JSON
EXECUTION DETAILS
EXECUTION GRAPH
PREVIEW
PERSONAL HISTORY
PROJECT HISTORY
REFRESH
Type here to search
24°C Cloudy
6:05 PM
3/18/2023

2. In-depth Exploration:

2.1.1 Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

Explanation:

Yes, when we see trend from 2016 to 2018, there is a huge jump in order purchase between years. This shows on growing trend in e-commerce business of Target in Brazil.

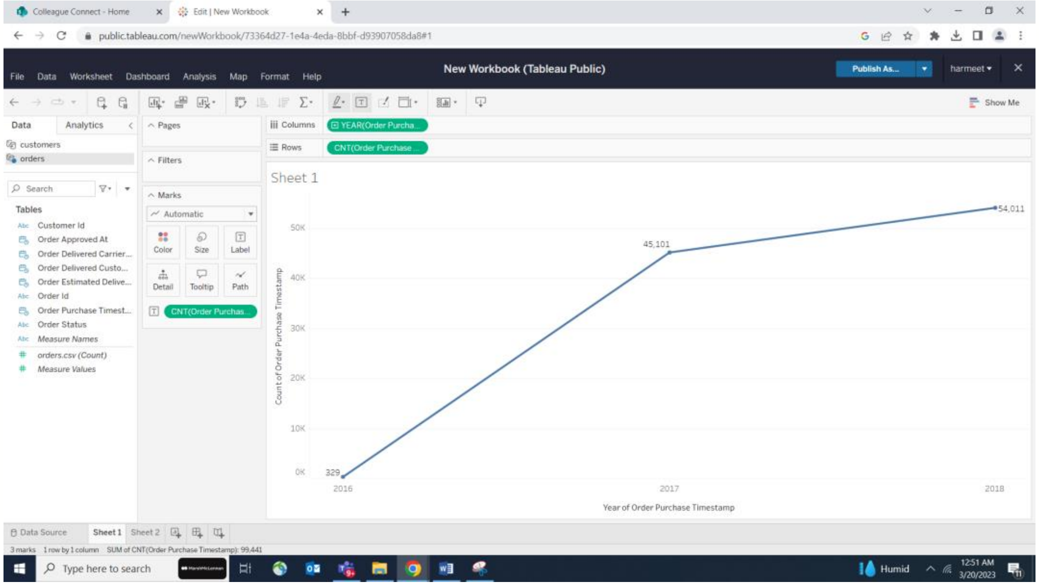


Tools used:

SQL (to check the total orders made in each month for every year):

```sql
select year,month,count(month)
from
(SELECT extract(year from order_purchase_timestamp) as year,extract(month from order_purchase_timestamp) month,
 FROM `project-380318.customers.orders`
order by year,month) tbl
group by year , month
order by year, month
```

Tableau (to visual the trend yearly):



public.tableau.com/newWorkbook/73364d27-1e4a-4eda-8bbf-d93907058da8#1

New Workbook (Tableau Public)

Columns: YEAR(Order Purcha...

Rows: CNT(Order Purchase ...

Marks: Automatic

CNT(Order Purchas...

Sheet 1

3 marks   1 row by 1 column   SUM of CNT(Order Purchase Timestamp): 99,441

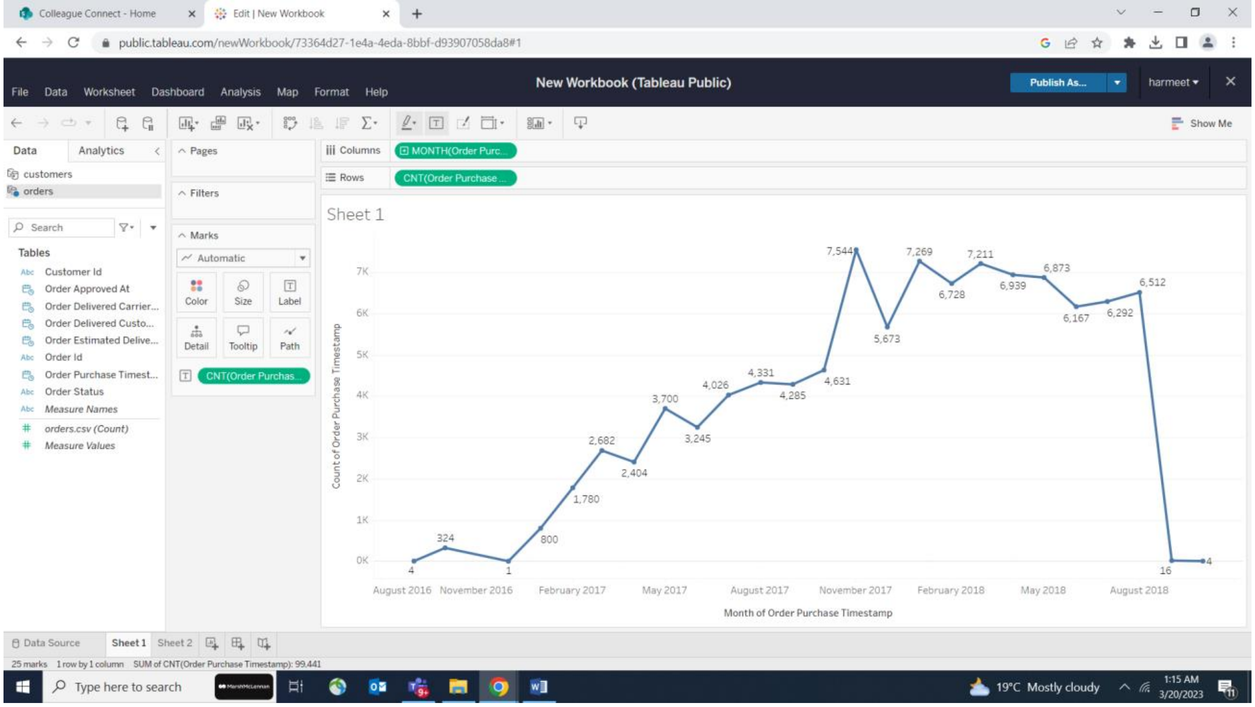
2.1.2 Can we see some seasonality with peaks at specific months?

We can see sudden drop in December 2016, September 2018 and October 2018 while maximum order was done in November 2017.

In 2017, we see upward trend only except a few drops in the months of April, June and December.

Explanation:

- Tableau (to visualize the trend month-wise and to analyze peaks and drops)
- Screenshot from Tableau:



25 marks 1 row by 1 column SUM of CNT(Order Purchase Timestamp): 99,441

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

Explanation:

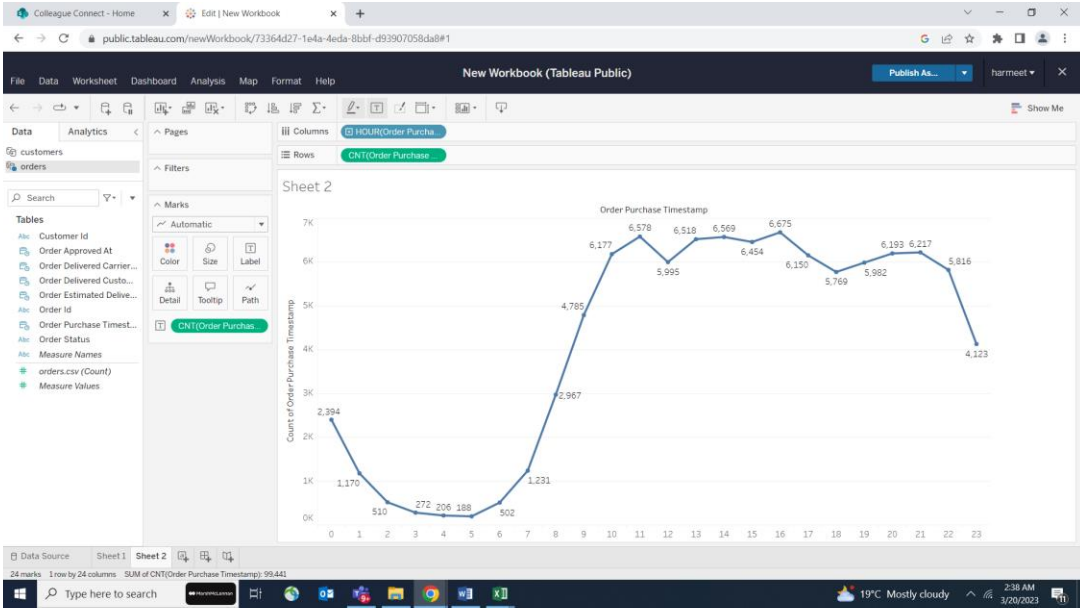
Most of the Brazilian prefer to buy after 10 A.M. till night around 9 or 10 P.M. A very few Brazilian buy between mid night till morning. **The maximum orders are placed during afternoons.**

Tools:

SQL query (to see top 10 order timing):

```
SELECT extract(hour from order_purchase_timestamp) as hour,count(order_purchase_timestamp) count
 FROM `project-380318.customers.orders`
group by hour
order by count desc
limit 10
```

Tableau (to visualize trend):



24 marks   1 row by 24 columns   SUM of CNT(Order Purchase Timestamp): 99,441

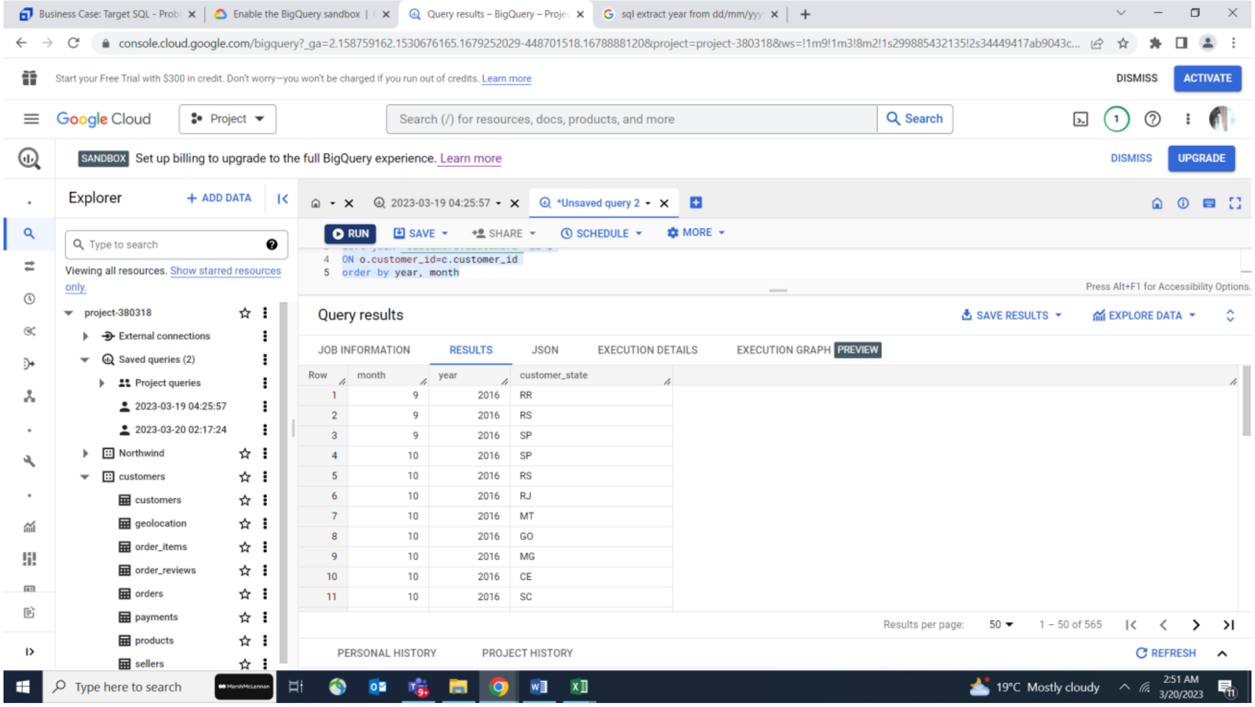
3.1. Get month on month orders by states:

Explanation:

SQL query:

```
select Distinct extract(month from order_purchase_timestamp) as month,extract(year from order_
purchase_timestamp) as year, c.customer_state
from `customers.orders` as o
left join `customers.customers` as c
ON o.customer_id=c.customer_id
order by year, month
```

Screenshot of result:



```
ON o.customer_id=c.customer_id
order by year, month
```

Query results

Row	month	year	customer_state
1	9	2016	RR
2	9	2016	RS
3	9	2016	SP
4	10	2016	SP
5	10	2016	RS
6	10	2016	RJ
7	10	2016	MT
8	10	2016	GO
9	10	2016	MG
10	10	2016	CE
11	10	2016	SC

Results per page: 50 1 – 50 of 565

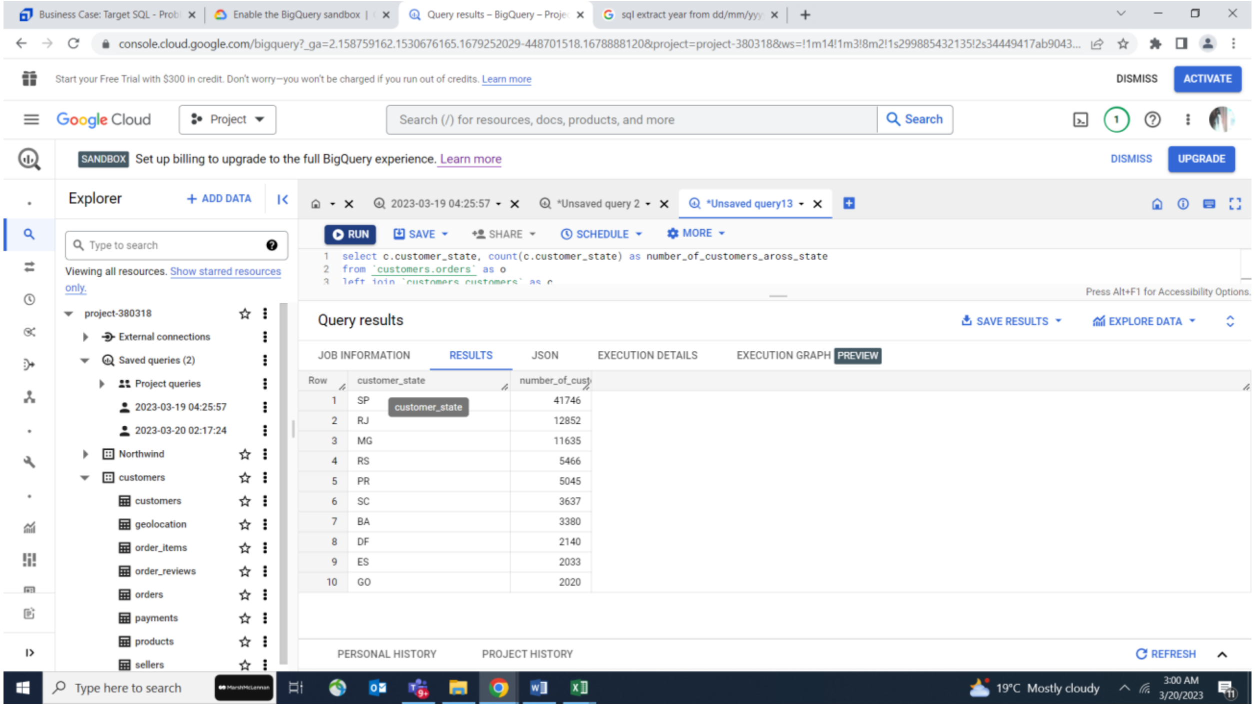
3.2 Distribution of customers across the states in Brazil

Explanation:

SQL query (to find the total number of customers purchased items from each state):

```
select c.customer_state, count(c.customer_state) as number_of_customers_aross_state
from `customers.orders` as o
left join `customers.customers` as c
ON o.customer_id=c.customer_id
group by customer_state
order by number_of_customers_aross_state desc
limit 10
```

Screenshot of result (showing top 10 states from where customers order in Brazil):



```
select c.customer_state, count(c.customer_state) as number_of_customers_aross_state
from `customers.orders` as o
left join `customers.customers` as c
```

Query results

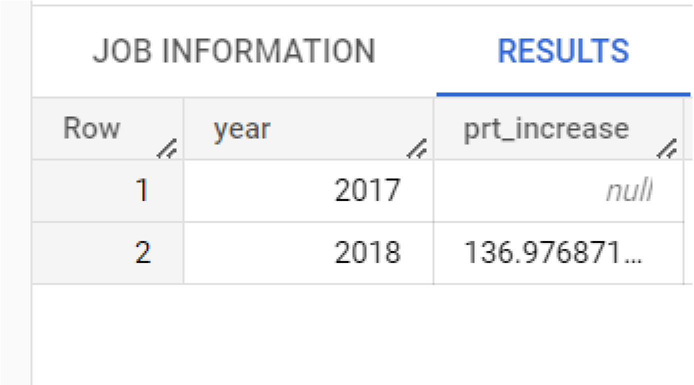
Row	customer_state	number_of_cust
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

4.1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

Sql query:

```
select tbl3.year,((tbl3.total_cost_year-tbl3.leading_year)/leading_year)*100 as prt_increase
from
(select tbl2.year, tbl2.total_cost_year,lead(tbl2.total_cost_year) over(order by tbl2.year des
c) as leading_year
from
(select tbl1.year,sum(total_cost) as total_cost_year
from
(select tbl.year,tbl.month,sum(payment_value) as total_cost
from
(select o.order_id,p.payment_value,extract(year from o.order_purchase_timestamp) as year, extr
act(month from o.order_purchase_timestamp) as month
from
`customers.orders` as o
left join `customers.payments` as p
ON o.order_id=p.order_id
order by year) tbl
where year IN (2017,2018) and month >=1 and month<=8
group by year,month) tbl1
group by year)tbl2)tbl3
```

Screenshot



Row	year	prt_increase
1	2017	null
2	2018	136.976871...

Analysis: there is a 20% increase in total cost of orders from

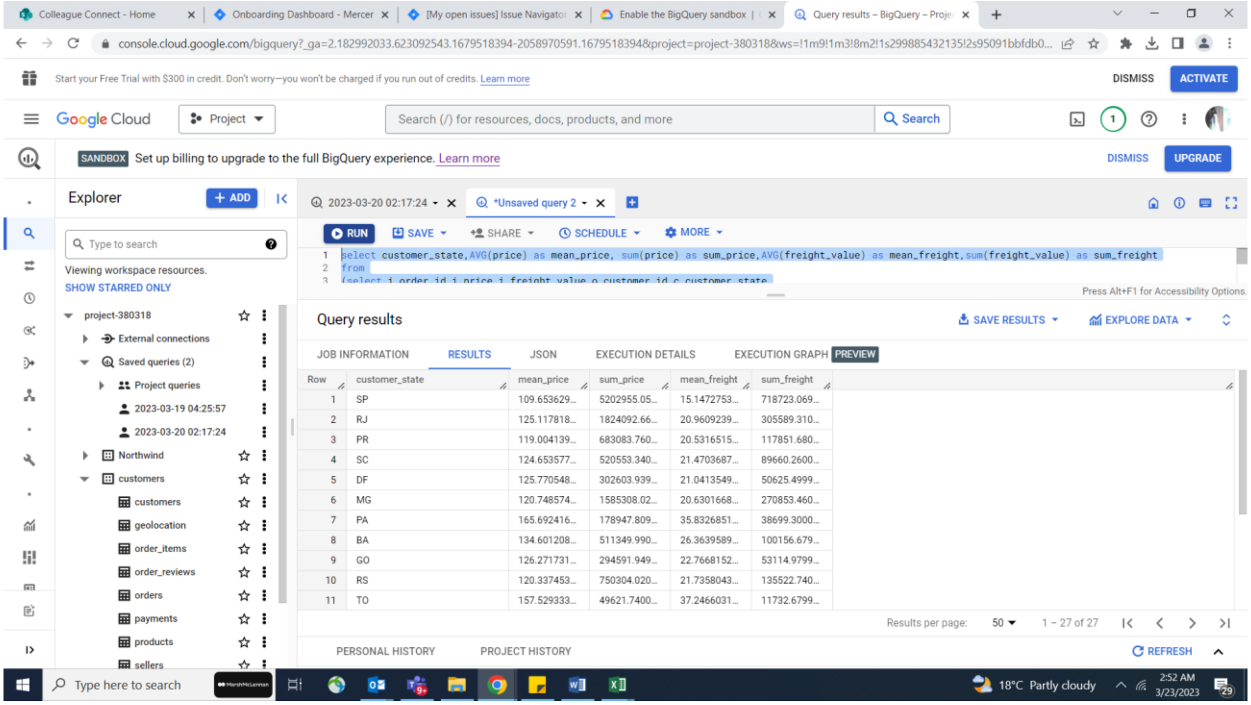
4.2. Mean & Sum of price and freight value by customer state:

Explanation:

SQL query:

```
select customer_state,AVG(price) as mean_price, sum(price) as sum_price,AVG(freight_value) as
mean_freight,sum(freight_value) as sum_freight
from
(select i.order_id,i.price,i.freight_value,o.customer_id,c.customer_state
from `customers.order_items` as i
left join `customers.orders` as o
ON i.order_id=o.order_id
left join `customers.customers` as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
```

Screenshot for result:



```
select customer_state,AVG(price) as mean_price, sum(price) as sum_price,AVG(freight_value) as mean_freight,sum(freight_value) as sum_freight
from
(select i.order_id,i.price,i.freight_value,o.customer_id,c.customer_state
```

Query results

Row	customer_state	mean_price	sum_price	mean_freight	sum_freight
1	SP	109.653629...	5202955.05...	15.1472753...	718723.069...
2	RJ	125.117818...	1824092.66...	20.9609239...	305589.310...
3	PR	119.004139...	683083.760...	20.5316515...	117851.680...
4	SC	124.653577...	520553.340...	21.4703687...	89660.2600...
5	DF	125.770548...	302603.939...	21.0413549...	50625.4999...
6	MG	120.748574...	1585308.02...	20.6301668...	270853.460...
7	PA	165.692416...	178947.809...	35.8326851...	38699.3000...
8	BA	134.601208...	511349.990...	26.3639589...	100156.679...
9	GO	126.271731...	294591.949...	22.7668152...	53114.9799...
10	RS	120.337453...	750304.020...	21.7358043...	135522.740...
11	TO	157.529333...	49621.7400...	37.2466031...	11732.6799...

Results per page: 50 1 – 27 of 27

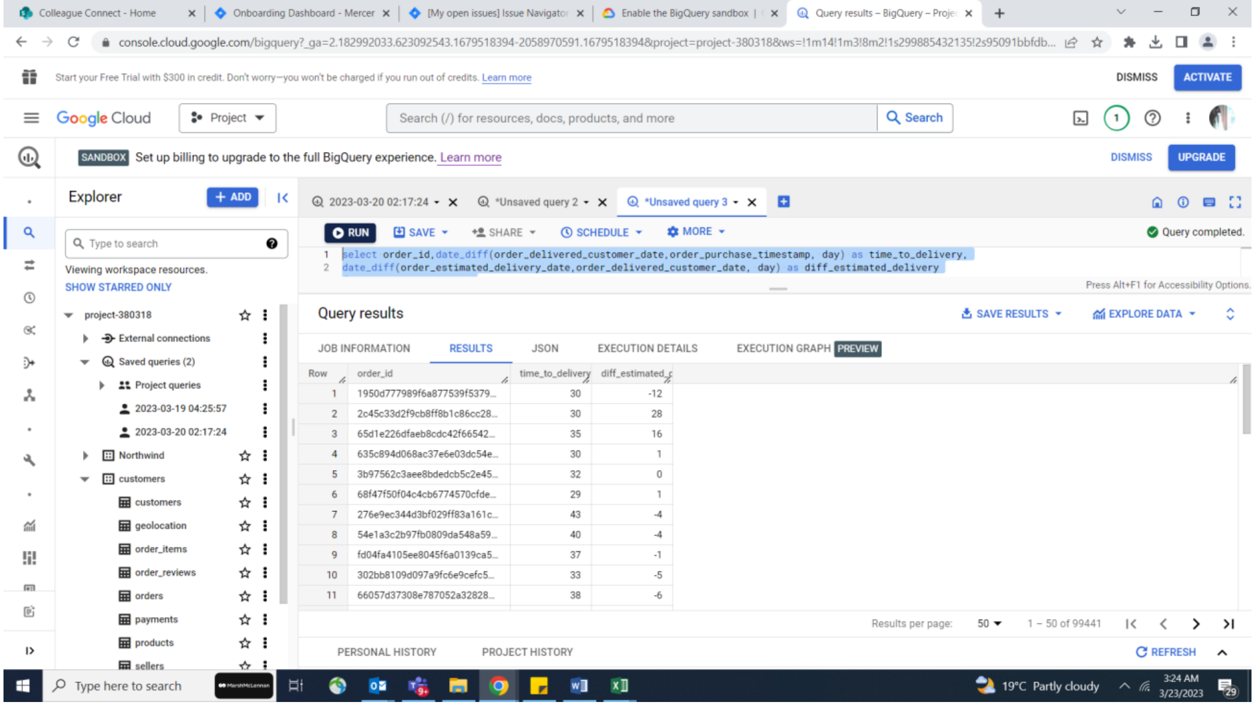
5.1 and 5.2. Calculate days between purchasing, delivering and estimated delivery:

Explanation:

SQL query:

```
select order_id,date_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time
_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as diff_estimated_
delivery
from `customers.orders`
```

Screenshot of result:



```
select order_id,date_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as diff_estimated_delivery
```

Row	order_id	time_to_delivery	diff_estimated_c
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5
11	66057d37308e787052a32828...	38	-6

Analysis:

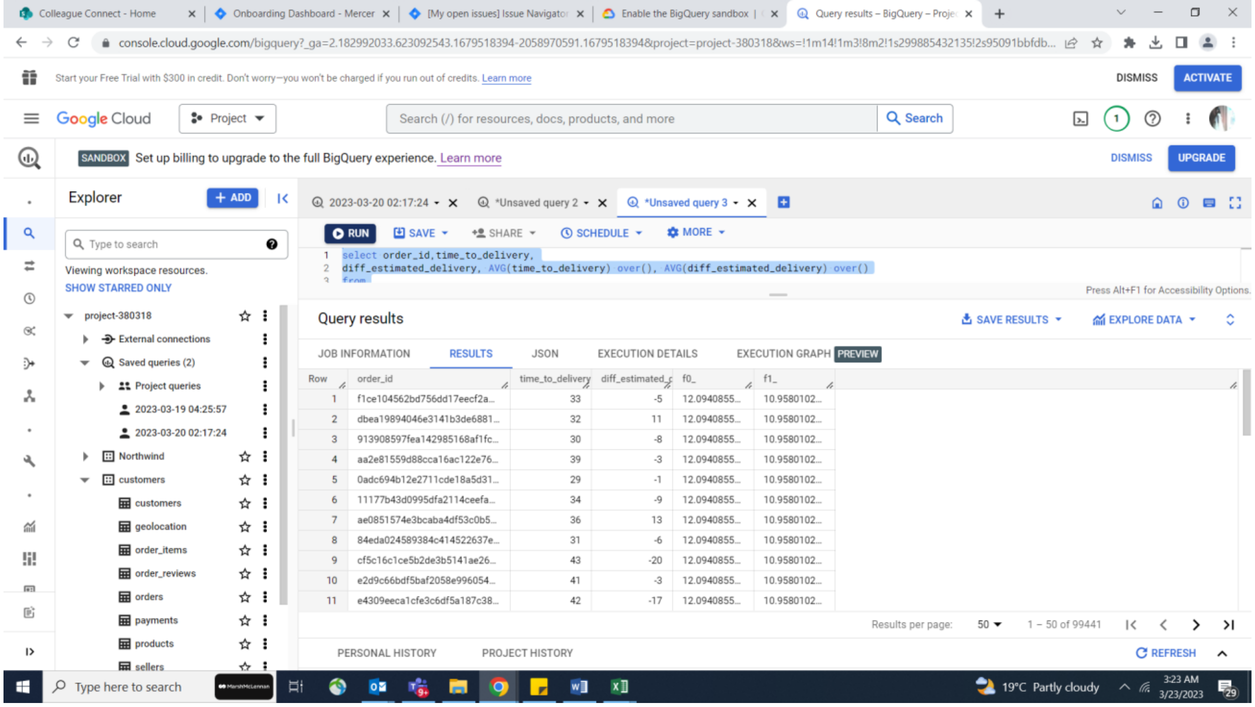
On an average, orders have delivered within 12 days after purchasing and delivery is done 10 days before estimated days.

Supported query:

```
select order_id,time_to_delivery,
diff_estimated_delivery, AVG(time_to_delivery) over(), AVG(diff_estimated_delivery) over()
from
```

```
(select order_id,date_diff(order_delivered_customer_date,order_purchase_timestamp, day) as tim
e_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) as diff_estimated_
delivery
from `customers.orders`) tbl
```

Screenshot for result:



```
select order_id,time_to_delivery,
diff_estimated_delivery, AVG(time_to_delivery) over(), AVG(diff_estimated_delivery) over()
from
```

Query results

Row	order_id	time_to_delivery	diff_estimated_c	f0_	f1_
1	f1ce104562bd756dd17eecf2a...	33	-5	12.0940855...	10.9580102...
2	dbea19894046e3141b3de6881...	32	11	12.0940855...	10.9580102...
3	913908597fea142985168af1fc...	30	-8	12.0940855...	10.9580102...
4	aa2e81559d88cca16ac122e76...	39	-3	12.0940855...	10.9580102...
5	0adc694b12e2711cde18a5d31...	29	-1	12.0940855...	10.9580102...
6	11177b43d0995dfa2114ceefa...	34	-9	12.0940855...	10.9580102...
7	ae0851574e3bcaba4df53c0b5...	36	13	12.0940855...	10.9580102...
8	84eda024589384c414522637e...	31	-6	12.0940855...	10.9580102...
9	cf5c16c1ce5b2de3b5141ae26...	43	-20	12.0940855...	10.9580102...
10	e2d9c66bdf5baf2058e996054...	41	-3	12.0940855...	10.9580102...
11	e4309eeca1cfe3c6df5a187c38...	42	-17	12.0940855...	10.9580102...

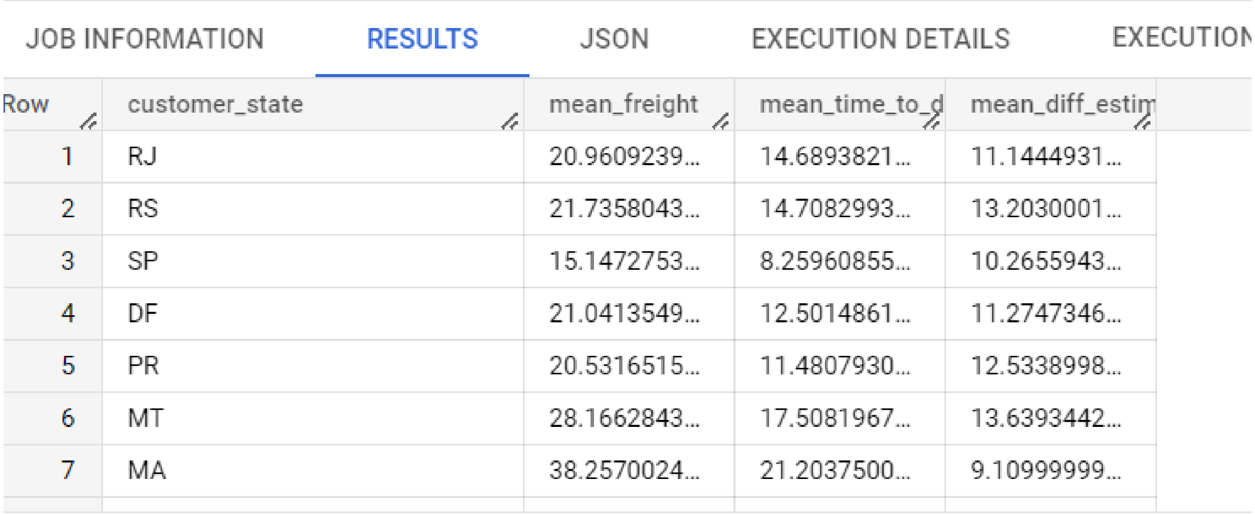
Results per page: 50 1 – 50 of 99441

5.3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery:

SQL query:

```
select customer_state,AVG(freight_value) as mean_freight,AVG(time_to_delivery) as mean_time_to
_delivery, AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
```

Screenshot of result:



Row	customer_state	mean_freight	mean_time_to_d	mean_diff_estim
1	RJ	20.9609239...	14.6893821...	11.1444931...
2	RS	21.7358043...	14.7082993...	13.2030001...
3	SP	15.1472753...	8.25960855...	10.2655943...
4	DF	21.0413549...	12.5014861...	11.2747346...
5	PR	20.5316515...	11.4807930...	12.5338998...
6	MT	28.1662843...	17.5081967...	13.6393442...
7	MA	38.2570024...	21.2037500...	9.10999999...

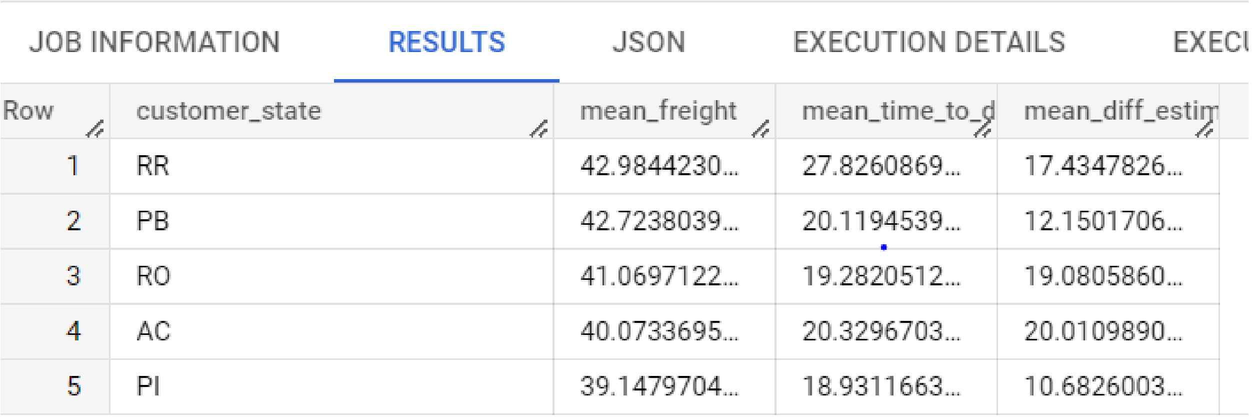
5.5 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5:

Highest:

SQL query:

```
select customer_state,AVG(freight_value) as mean_freight,AVG(time_to_delivery) as mean_time_to
_delivery, AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
order by mean_freight desc
limit 5
```

Screenshot:



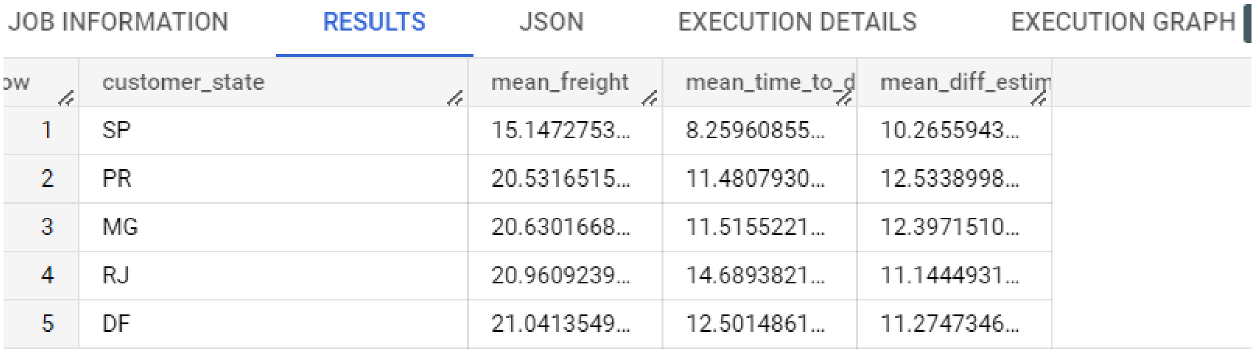
Row	customer_state	mean_freight	mean_time_to_d	mean_diff_estim
1	RR	42.9844230...	27.8260869...	17.4347826...
2	PB	42.7238039...	20.1194539...	12.1501706...
3	RO	41.0697122...	19.2820512...	19.0805860...
4	AC	40.0733695...	20.3296703...	20.0109890...
5	PI	39.1479704...	18.9311663...	10.6826003...

Lowest:

SQL query:

```
select customer_state,AVG(freight_value) as mean_freight,AVG(time_to_delivery) as mean_time_to
_delivery, AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
order by mean_freight
limit 5
```

Screenshot:



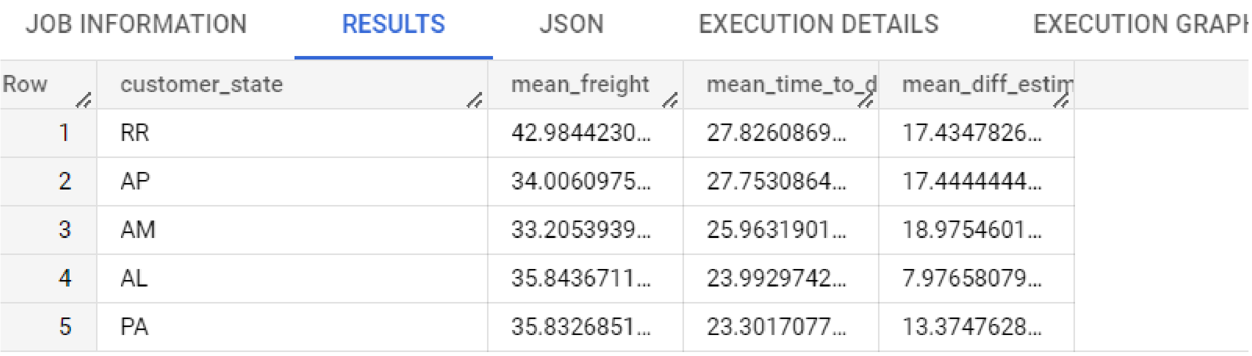
ow	customer_state	mean_freight	mean_time_to_d	mean_diff_estim
1	SP	15.1472753...	8.25960855...	10.2655943...
2	PR	20.5316515...	11.4807930...	12.5338998...
3	MG	20.6301668...	11.5155221...	12.3971510...
4	RJ	20.9609239...	14.6893821...	11.1444931...
5	DF	21.0413549...	12.5014861...	11.2747346...

5.6 Top 5 states with highest/lowest average time to delivery:

Highest:

```
select customer_state,AVG(freight_value) as mean_freight,AVG(time_to_delivery) as mean_time_to
_delivery, AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
order by mean_time_to_delivery desc
limit 5
```

screenshot:



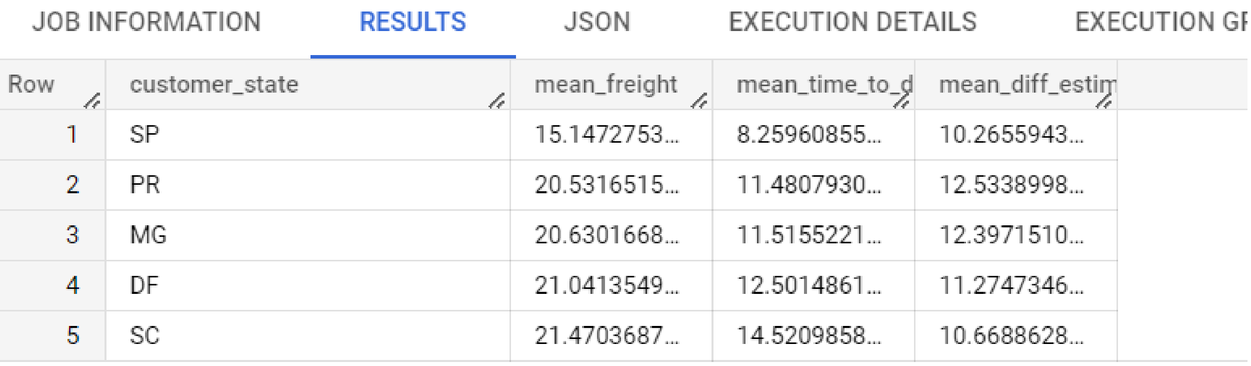
Row	customer_state	mean_freight	mean_time_to_d	mean_diff_estim
1	RR	42.9844230...	27.8260869...	17.4347826...
2	AP	34.0060975...	27.7530864...	17.4444444...
3	AM	33.2053939...	25.9631901...	18.9754601...
4	AL	35.8436711...	23.9929742...	7.97658079...
5	PA	35.8326851...	23.3017077...	13.3747628...

Lowest:

SQL query:

```
select customer_state,AVG(freight_value) as mean_freight,AVG(time_to_delivery) as mean_time_to
_delivery, AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state
order by mean_time_to_delivery
limit 5
```

screenshot:



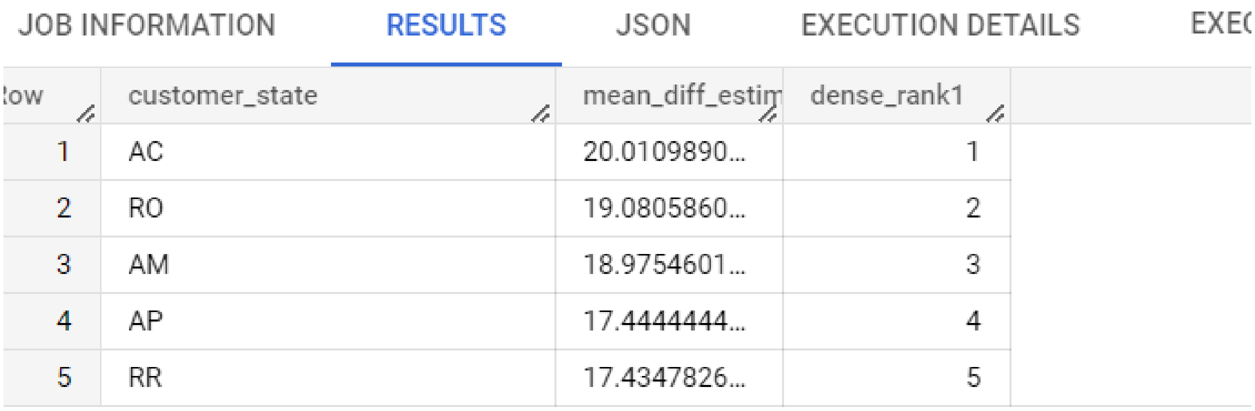
Row	customer_state	mean_freight	mean_time_to_d	mean_diff_estim
1	SP	15.1472753...	8.25960855...	10.2655943...
2	PR	20.5316515...	11.4807930...	12.5338998...
3	MG	20.6301668...	11.5155221...	12.3971510...
4	DF	21.0413549...	12.5014861...	11.2747346...
5	SC	21.4703687...	14.5209858...	10.6688628...

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

Top 5 with really fast delivery:

```
select customer_state, mean_diff_estimated_delivery,dense_rank() over(order by tbl1.mean_diff_
estimated_delivery desc) as dense_rank1
from
(select customer_state,AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estima
ted_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state) tbl1
order by dense_rank1
limit 5
```

Screenshot:



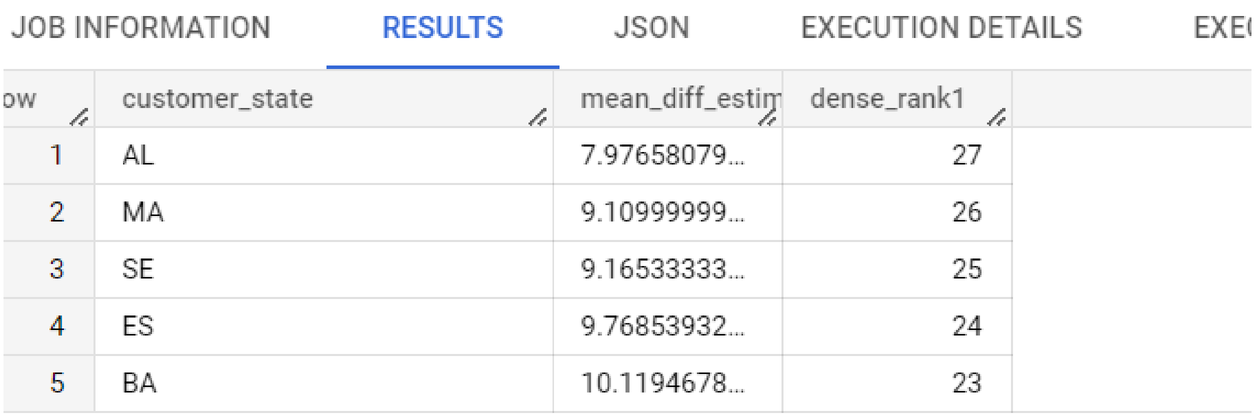
Row	customer_state	mean_diff_estim	dense_rank1
1	AC	20.0109890...	1
2	RO	19.0805860...	2
3	AM	18.9754601...	3
4	AP	17.4444444...	4
5	RR	17.4347826...	5

States with ‘not so fast’ delivery:

SQL query:

```
select customer_state, mean_diff_estimated_delivery,dense_rank() over(order by tbl1.mean_diff_estimated_delivery desc) as dense_rank1
from
(select customer_state,AVG(diff_estimated_delivery) as mean_diff_estimated_delivery
from
(select o.order_id,date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)
as time_to_delivery,
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as diff_estimated_delivery, i.freight_value,c.customer_state
from `customers.order_items` as i
right join `customers.orders`as o
ON o.order_id=i.order_id
left join `customers.customers`as c
ON o.customer_id=c.customer_id) tbl
group by customer_state) tbl1
order by dense_rank1 desc
limit 5
```

screenshot:



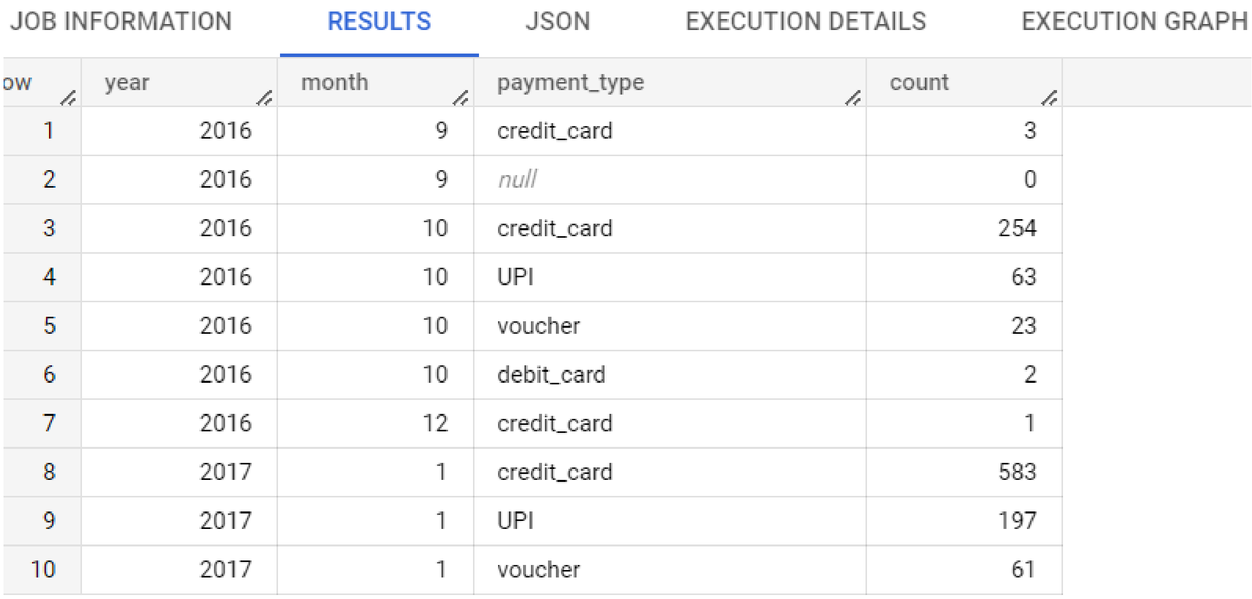
ow	customer_state	mean_diff_estim	dense_rank1
1	AL	7.97658079...	27
2	MA	9.10999999...	26
3	SE	9.16533333...	25
4	ES	9.76853932...	24
5	BA	10.1194678...	23

6.1Month over Month count of orders for different payment types

Sql query:

```
select year,month, payment_type,count(payment_type) as count
from
(select o.order_id,extract(month from o.order_purchase_timestamp) as month,extract (year from
o.order_purchase_timestamp) as year,p.payment_type
from `customers.orders`as o
left join `customers.payments`as p
ON o.order_id=p.order_id)tbl
group by year,month,payment_type
order by year,month
```

screenshot:



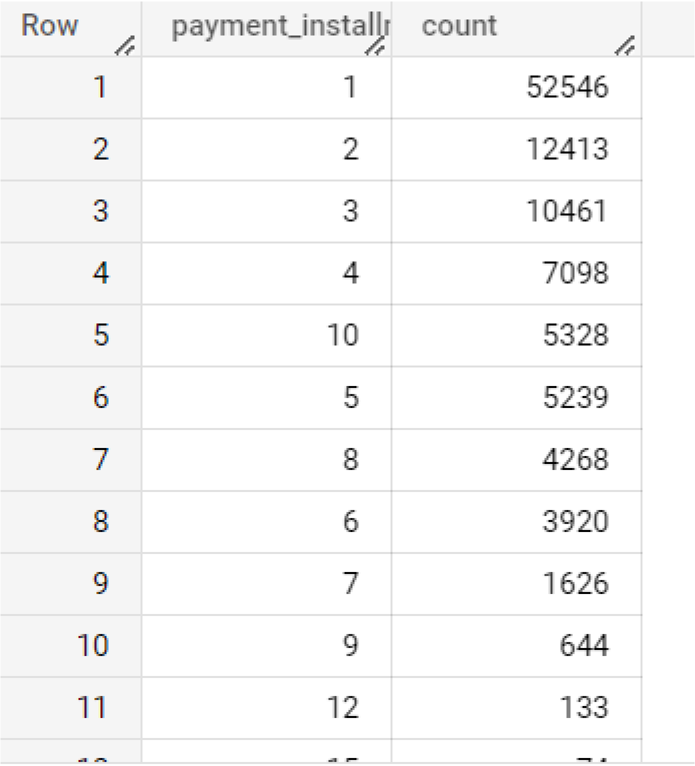
ow	year	month	payment_type	count
1	2016	9	credit_card	3
2	2016	9	*null*	0
3	2016	10	credit_card	254
4	2016	10	UPI	63
5	2016	10	voucher	23
6	2016	10	debit_card	2
7	2016	12	credit_card	1
8	2017	1	credit_card	583
9	2017	1	UPI	197
10	2017	1	voucher	61

6.2Count of orders based on the no. of payment installments

SQL query:

```
select payment_installments,count(order_id) as count
from
(select o.order_id,p.payment_installments
from `customers.orders`as o
left join `customers.payments`as p
ON o.order_id=p.order_id)tbl
group by payment_installments
order by count desc
```

Screenshot:



Row	payment_install	count
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133