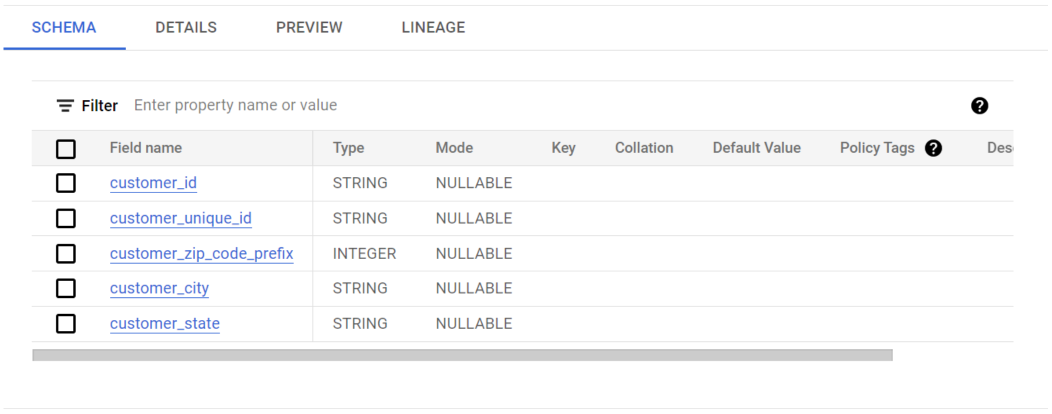
TARGET-SQL

I. 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 :

SQL Syntax:

```
Select column_name, data_type
From target_sql.information_schema.columns
Where table_name = 'customers'
And table_schema = 'target_sql';
```



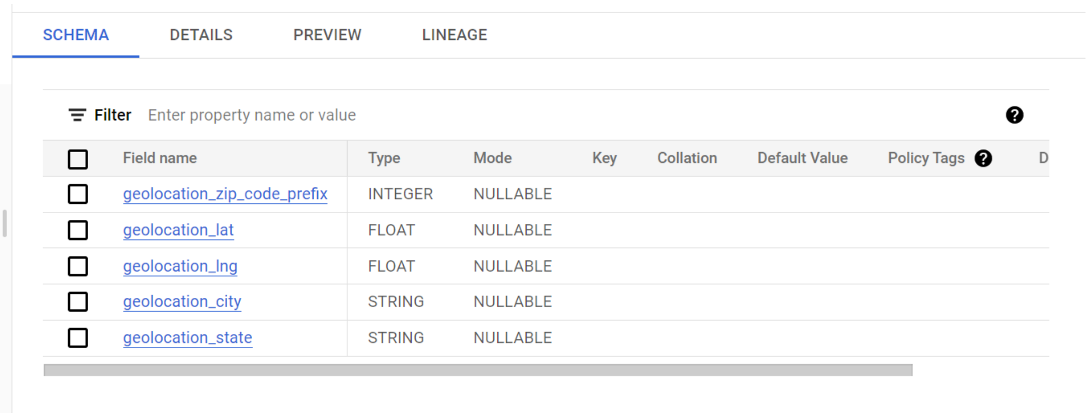
SCHEMA | DETAILS | PREVIEW | LINEAGE

Filter Enter property name or value

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

SQL Syntax:

```
Select column_name, data_type
From target_sql.information_schema.columns
Where table_name = 'geo_location'
And table_schema = 'target_sql';
```



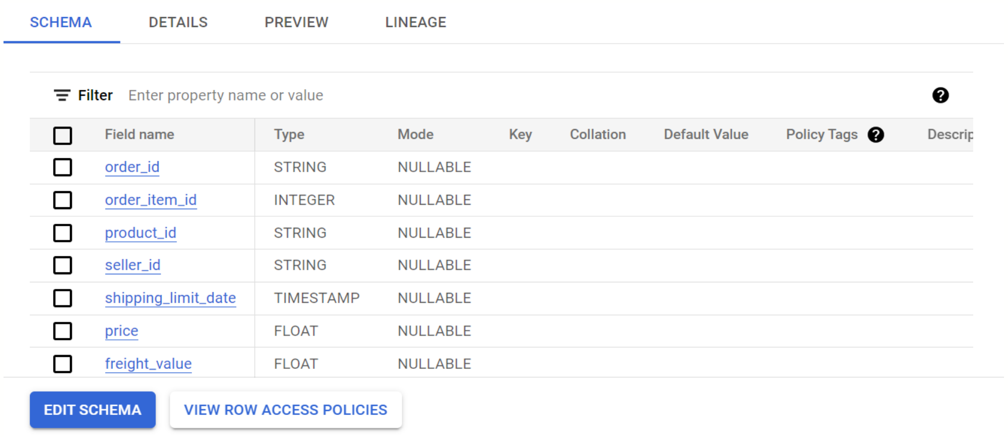
SCHEMA | DETAILS | PREVIEW | LINEAGE

Filter Enter property name or value

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

SQL Syntax:

```
SELECT column_name, data_type
FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'Order_items' AND table_schema = 'Target_SQL';
```


SCHEMA DETAILS PREVIEW LINEAGE

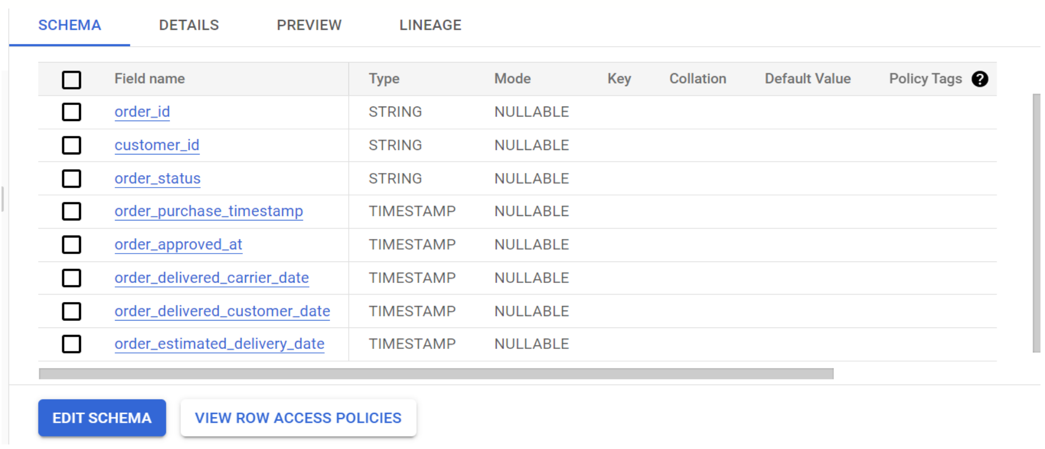
Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Descrip
order_id	STRING	NULLABLE					
order_item_id	INTEGER	NULLABLE					
product_id	STRING	NULLABLE					
seller_id	STRING	NULLABLE					
shipping_limit_date	TIMESTAMP	NULLABLE					
price	FLOAT	NULLABLE					
freight_value	FLOAT	NULLABLE					

EDIT SCHEMA VIEW ROW ACCESS POLICIES

SQL Syntax:

```
Select column_name, data_type
From target_sql.information_schema.columns
Where table_name = 'orders'
And table_schema = 'target_sql';
```

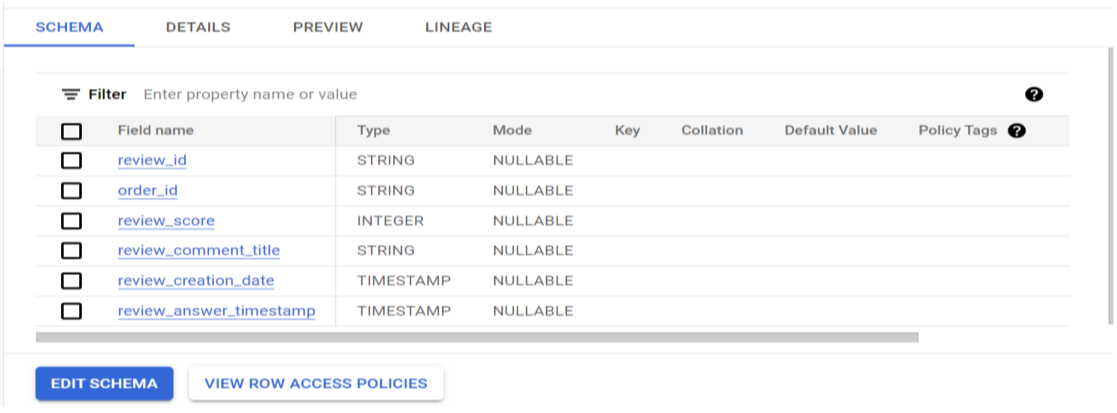

SCHEMA DETAILS PREVIEW LINEAGE

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
order_id	STRING	NULLABLE				
customer_id	STRING	NULLABLE				
order_status	STRING	NULLABLE				
order_purchase_timestamp	TIMESTAMP	NULLABLE				
order_approved_at	TIMESTAMP	NULLABLE				
order_delivered_carrier_date	TIMESTAMP	NULLABLE				
order_delivered_customer_date	TIMESTAMP	NULLABLE				
order_estimated_delivery_date	TIMESTAMP	NULLABLE				

EDIT SCHEMA VIEW ROW ACCESS POLICIES

SQL Syntax:

```
SELECT column_name, data_type
FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'Order_reviews'
AND table_schema = 'Target_SQL';
```


SCHEMA DETAILS PREVIEW LINEAGE

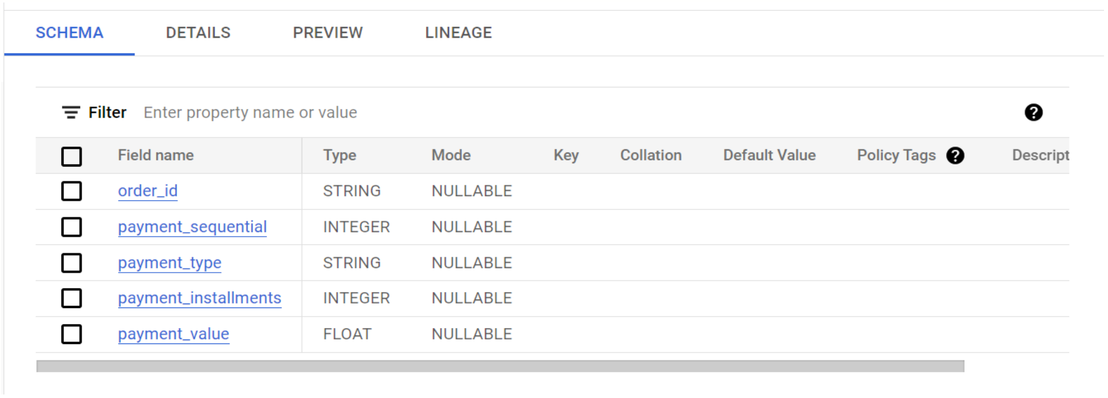
Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
review_id	STRING	NULLABLE				
order_id	STRING	NULLABLE				
review_score	INTEGER	NULLABLE				
review_comment_title	STRING	NULLABLE				
review_creation_date	TIMESTAMP	NULLABLE				
review_answer_timestamp	TIMESTAMP	NULLABLE				

EDIT SCHEMA VIEW ROW ACCESS POLICIES

SQL Syntax:

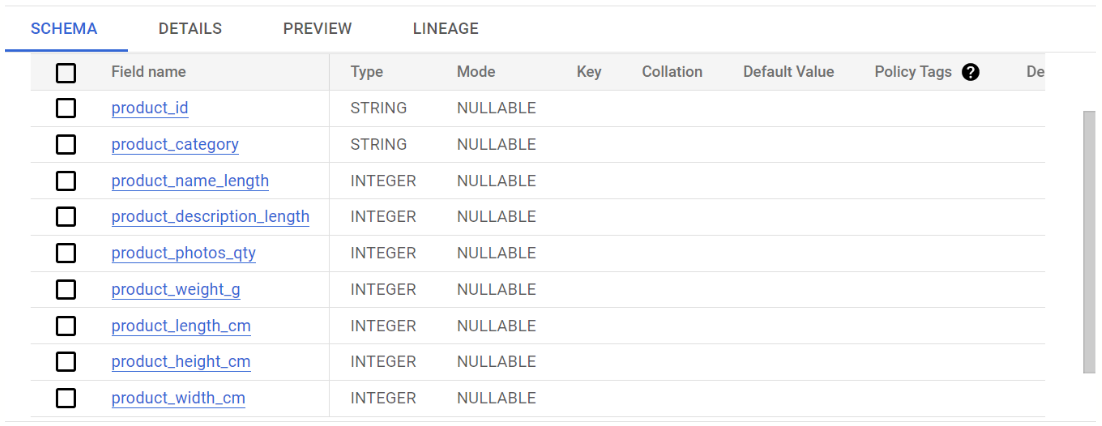
```
SELECT column_name, data_type
FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'Payments'
AND table_schema = 'Target_SQL';
```



Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Descript
order_id	STRING	NULLABLE					
payment_sequential	INTEGER	NULLABLE					
payment_type	STRING	NULLABLE					
payment_installments	INTEGER	NULLABLE					
payment_value	FLOAT	NULLABLE					

SQL Syntax:

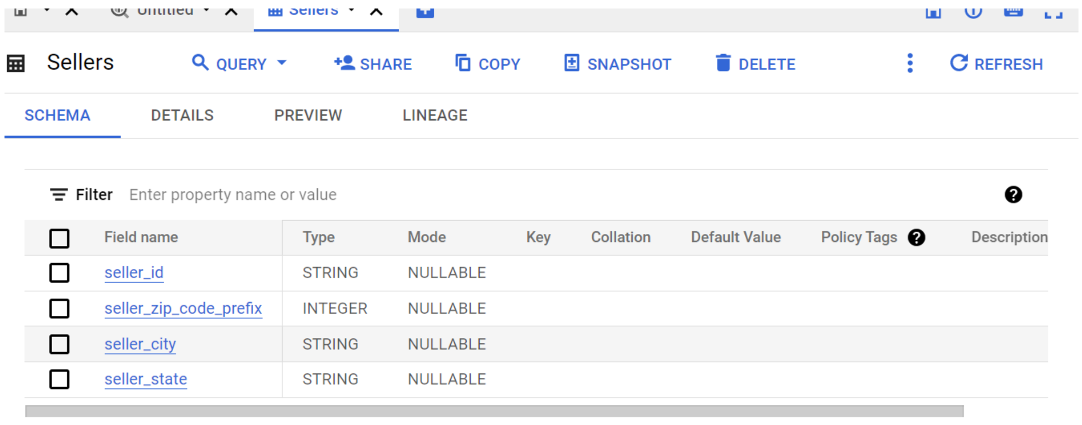
```
Select column_name, data_type
From target_sql.information_schema.columns
Where table_name = 'products'
And table_schema = 'target_sql';
```



Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	De
product_id	STRING	NULLABLE					
product_category	STRING	NULLABLE					
product_name_length	INTEGER	NULLABLE					
product_description_length	INTEGER	NULLABLE					
product_photos_qty	INTEGER	NULLABLE					
product_weight_g	INTEGER	NULLABLE					
product_length_cm	INTEGER	NULLABLE					
product_height_cm	INTEGER	NULLABLE					
product_width_cm	INTEGER	NULLABLE					

SQL Syntax:

```
Select column_name, data_type
From target_sql.information_schema.columns
Where table_name = 'sellers'
And table_schema = 'target_sql';
```



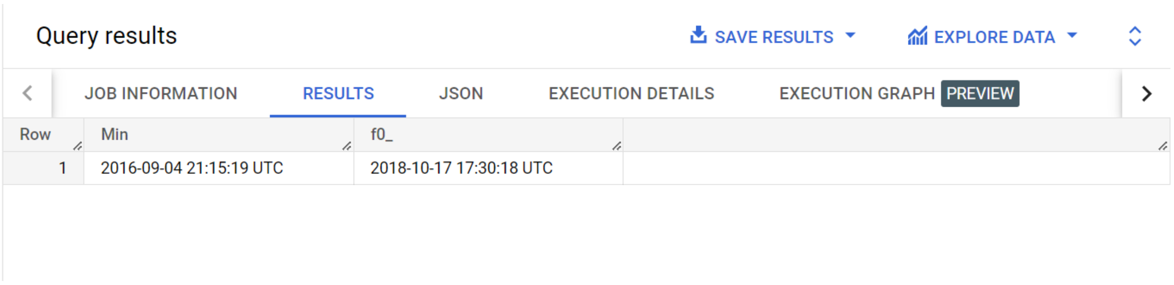
Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
seller_id	STRING	NULLABLE					
seller_zip_code_prefix	INTEGER	NULLABLE					
seller_city	STRING	NULLABLE					
seller_state	STRING	NULLABLE					

- After the basic analysis of Target - data structure the data type of columns are mostly STRING, INTEGER, FLOAT and TIMESTAMP .

2. TIME PERIOD FOR WHICH THE DATA IS GIVEN:

SQL Syntax :

```
Select order_purchase_timestamp
From target_database.orders ;
```



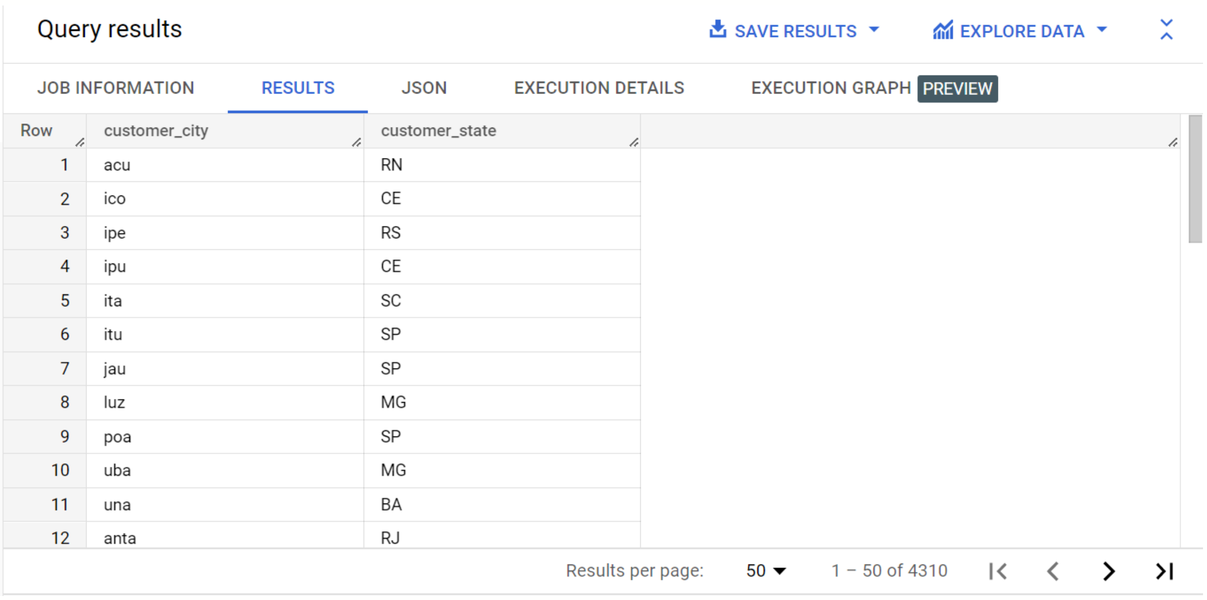
Row	Min	f0_
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

- We have to analyse the data provided in the project which is listed from 2016-09-04 to 2018-10-17 UTC.

3. CITIES AND STATES OF CUSTOMERS ORDERED DURING THE GIVEN PERIOD :

SQL Syntax :

```
select distinct C.customer_city, C.customer_state FROM
Target_Database.Customers as C
join `Target_Database.Orders` as O on O.customer_id = C.customer_id ;
```



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
11	una	BA
12	anta	RJ

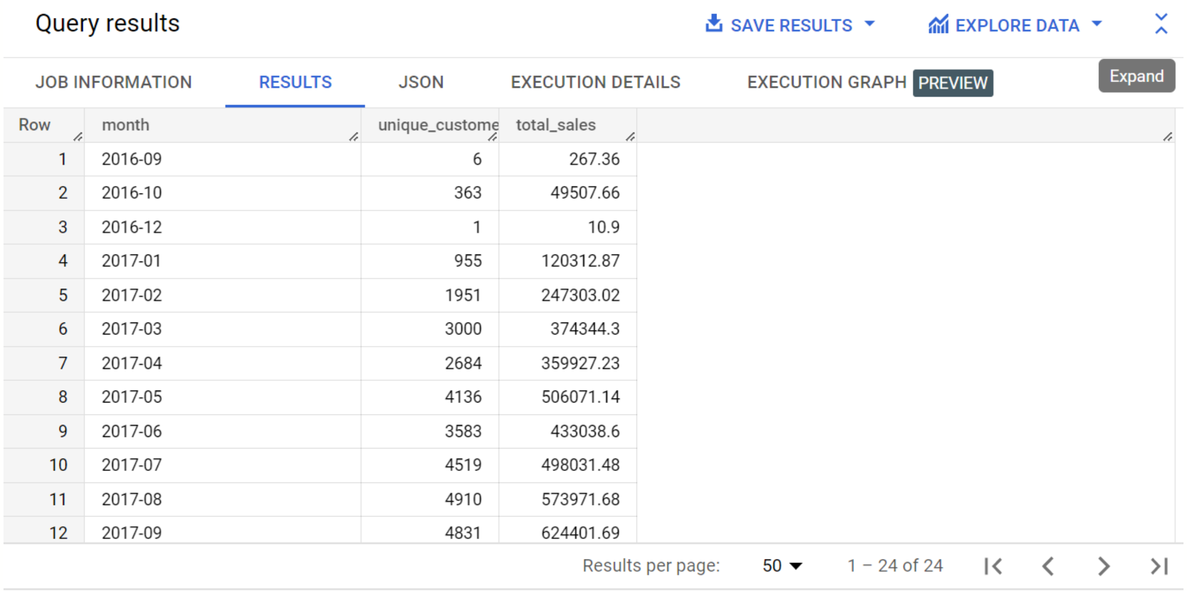
Results per page: 50 1 – 50 of 4310

II. 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?

SQL Syntax:

```
select FORMAT_DATE("%Y-%m", o.order_purchase_timestamp) AS month,
COUNT(c.customer_unique_id) AS unique_customers,
ROUND(SUM(oi.price), 2) AS total_sales
from `Target_Database.Orders`as o
join `Target_Database.Customers`as c on c.customer_id = o.customer_id
join `Target_Database.Order_items`as oi on oi.order_id = o.order_id
group by month
order by month ;
```



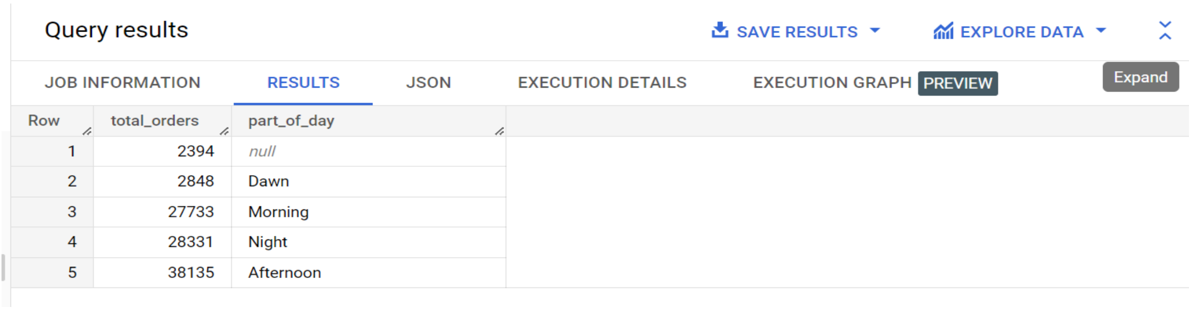
Row	month	unique_custome	total_sales
1	2016-09	6	267.36
2	2016-10	363	49507.66
3	2016-12	1	10.9
4	2017-01	955	120312.87
5	2017-02	1951	247303.02
6	2017-03	3000	374344.3
7	2017-04	2684	359927.23
8	2017-05	4136	506071.14
9	2017-06	3583	433038.6
10	2017-07	4519	498031.48
11	2017-08	4910	573971.68
12	2017-09	4831	624401.69

Results per page: 50 1 – 24 of 24

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

SQL syntax :

```
select count(order_id) as total_orders,
CASE
  when extract (HOUR from order_purchase_timestamp) between 1 and 6 then 'Dawn'
  when extract(HOUR from order_purchase_timestamp) between 7 and 12 then 'Morning'
  when extract(HOUR from order_purchase_timestamp) between 13 and 18 then 'Afternoon'
  when extract(HOUR from order_purchase_timestamp) between 19 and 23 then 'Night'
end as part_of_day
from `Target_Database.Orders`
group by part_of_day
order by total_orders ;
```



Row	total_orders	part_of_day
1	2394	*null*
2	2848	Dawn
3	27733	Morning
4	28331	Night
5	38135	Afternoon

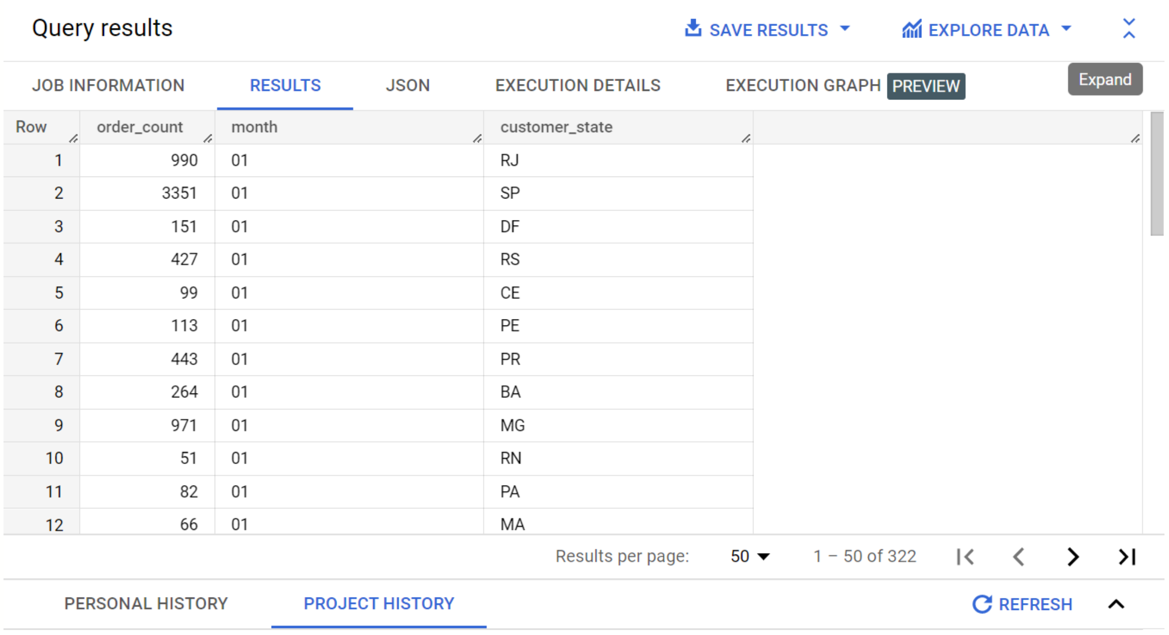
- Brazilians are fond to shop in the Afternoon and second best option is Night.

III. Evolution of E-commerce orders in the Brazil region:

1. Get month on month orders by states :

SQL Syntax :

```
SELECT
count(o.order_id) as order_count,
format_date('%m',order_purchase_timestamp)as month, c.customer_state
from `Target_Database.Customers` as c
right join `Target_Database.Orders` as o
on o.customer_id = c.customer_id
group by month,c.customer_state
order by month ;
```


Query results

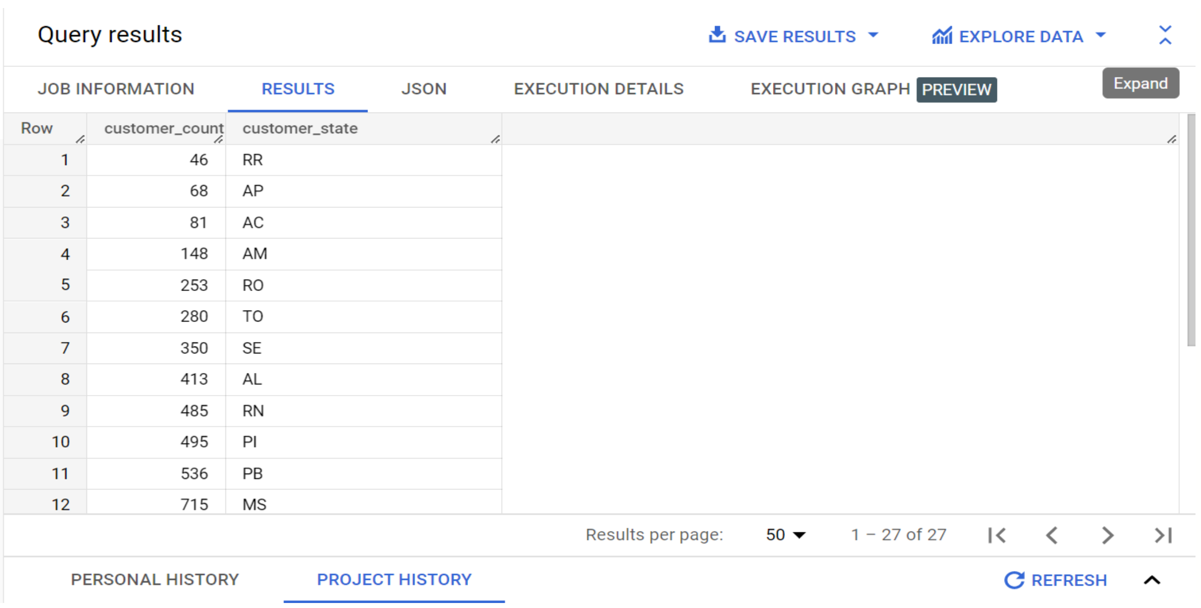
Row	order_count	month	customer_state
1	990	01	RJ
2	3351	01	SP
3	151	01	DF
4	427	01	RS
5	99	01	CE
6	113	01	PE
7	443	01	PR
8	264	01	BA
9	971	01	MG
10	51	01	RN
11	82	01	PA
12	66	01	MA

Results per page: 50 1 – 50 of 322

2. Distribution of customers across the states in Brazil :

SQL Syntax :

```
select count(customer_id) as customer_count, customer_state
from `Target_Database.Customers`
group by customer_state
order by customer_count ;
```



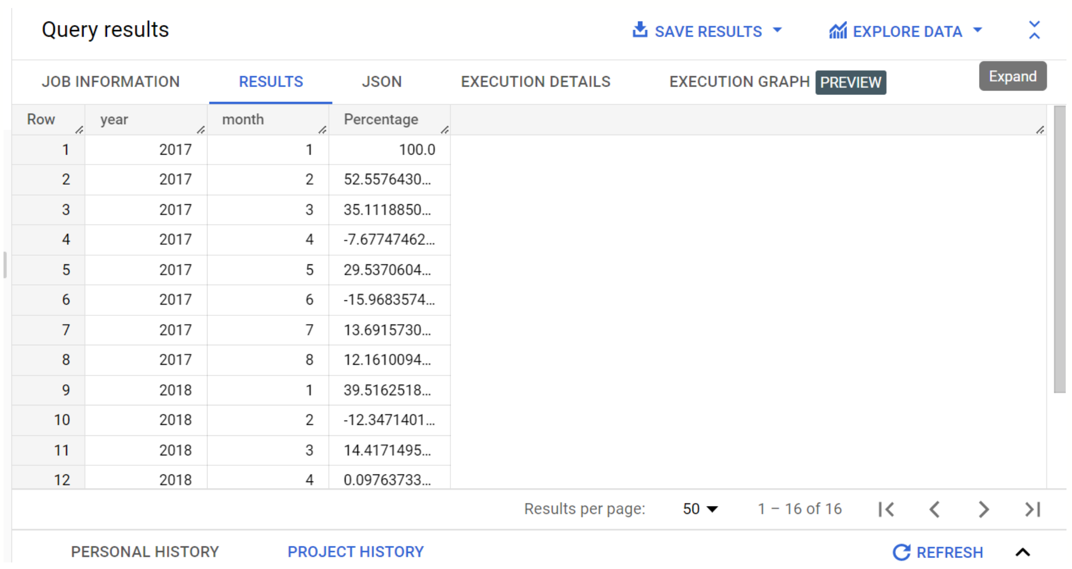
Row	customer_count	customer_state
1	46	RR
2	68	AP
3	81	AC
4	148	AM
5	253	RO
6	280	TO
7	350	SE
8	413	AL
9	485	RN
10	495	PI
11	536	PB
12	715	MS

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

1. 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 :

   SQL Syntax :

```
Select year,month,((total_value-
ifnull(lag(total_value)over(order by year, month),0))/total_value)*100 as
Percentage
From(select sum(payment_value) as total_value,
Extract(year from o.order_purchase_timestamp) as year,
Extract(month from o.order_purchase_timestamp) as month
From target_sql.payments as p
Inner join target_sql.orders as o on p.order_id = o.order_id
Where
Extract(year from o.order_purchase_timestamp) in (2017, 2018) and
Extract(month from o.order_purchase_timestamp) in (1,2,3,4,5,6,7,8)
Group by year,month
Order by year,month)
Order by year,month ;
```

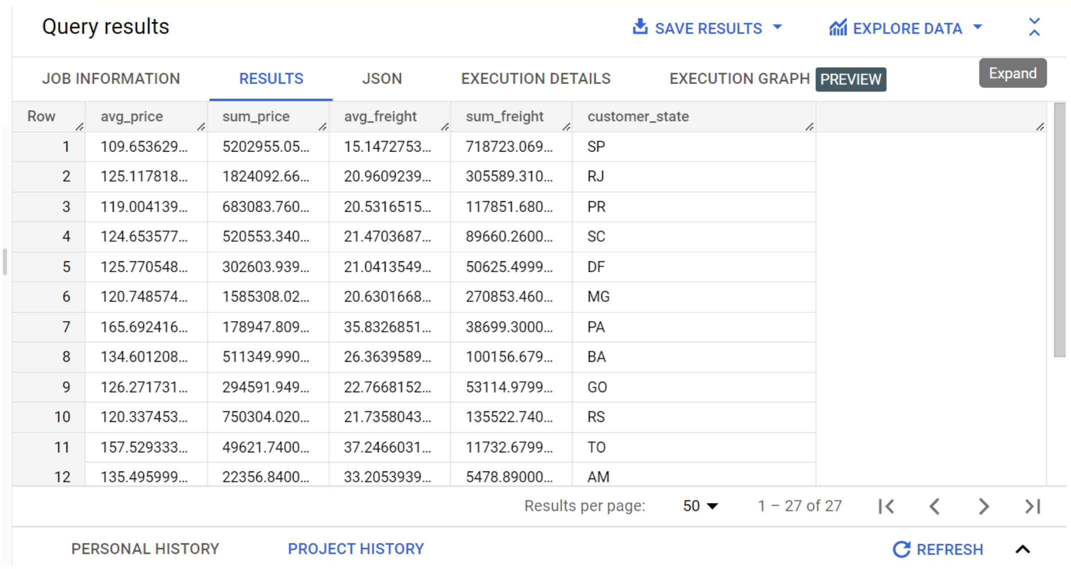


Row	year	month	Percentage
1	2017	1	100.0
2	2017	2	52.5576430...
3	2017	3	35.1118850...
4	2017	4	-7.67747462...
5	2017	5	29.5370604...
6	2017	6	-15.9683574...
7	2017	7	13.6915730...
8	2017	8	12.1610094...
9	2018	1	39.5162518...
10	2018	2	-12.3471401...
11	2018	3	14.4171495...
12	2018	4	0.09763733...

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

SQL Syntax :

```
select
avg(price) as avg_price,
sum(price) as sum_price,
avg(freight_value) as avg_freight,
sum(freight_value) as sum_freight,
customer_state
from `Target_Database.Order_items` as oi
join `Target_Database.Orders` as o on o.order_id = oi.order_id
join `Target_Database.Customers` as c on c.customer_id = o.customer_id
group by c.customer_state ;
```



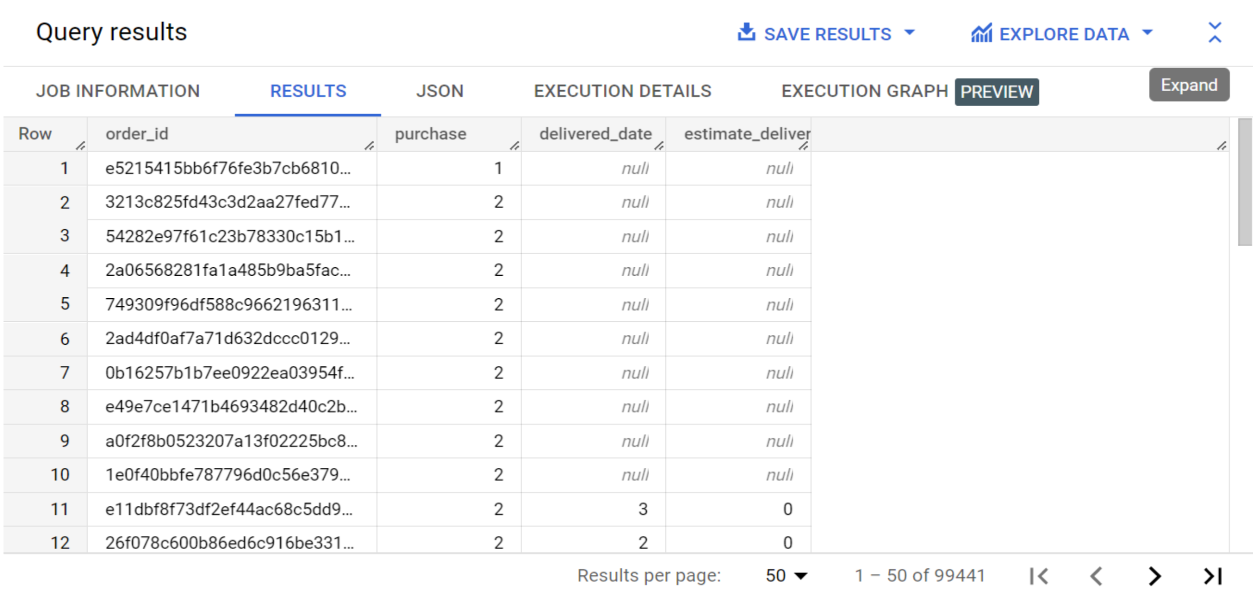
Row	avg_price	sum_price	avg_freight	sum_freight	customer_state
1	109.653629...	5202955.05...	15.1472753...	718723.069...	SP
2	125.117818...	1824092.66...	20.9609239...	305589.310...	RJ
3	119.004139...	683083.760...	20.5316515...	117851.680...	PR
4	124.653577...	520553.340...	21.4703687...	89660.2600...	SC
5	125.770548...	302603.939...	21.0413549...	50625.4999...	DF
6	120.748574...	1585308.02...	20.6301668...	270853.460...	MG
7	165.692416...	178947.809...	35.8326851...	38699.3000...	PA
8	134.601208...	511349.990...	26.3639589...	100156.679...	BA
9	126.271731...	294591.949...	22.7668152...	53114.9799...	GO
10	120.337453...	750304.020...	21.7358043...	135522.740...	RS
11	157.529333...	49621.7400...	37.2466031...	11732.6799...	TO
12	135.495999...	22356.8400...	33.2053939...	5478.89000...	AM

V. Analysis on sales, freight and delivery time :

1. Calculate days between purchasing, delivering and estimated delivery ?

SQL Syntax :

```
select order_id,
datetime_diff(order_estimated_delivery_date,order_purchase_timestamp,day)
as purchase,
datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day)
as delivered_date,
datetime_diff(order_delivered_customer_date,order_estimated_delivery_date,
day) as estimate_delivery
from `Target_Database.Orders`
order by purchase ;
```


Query results

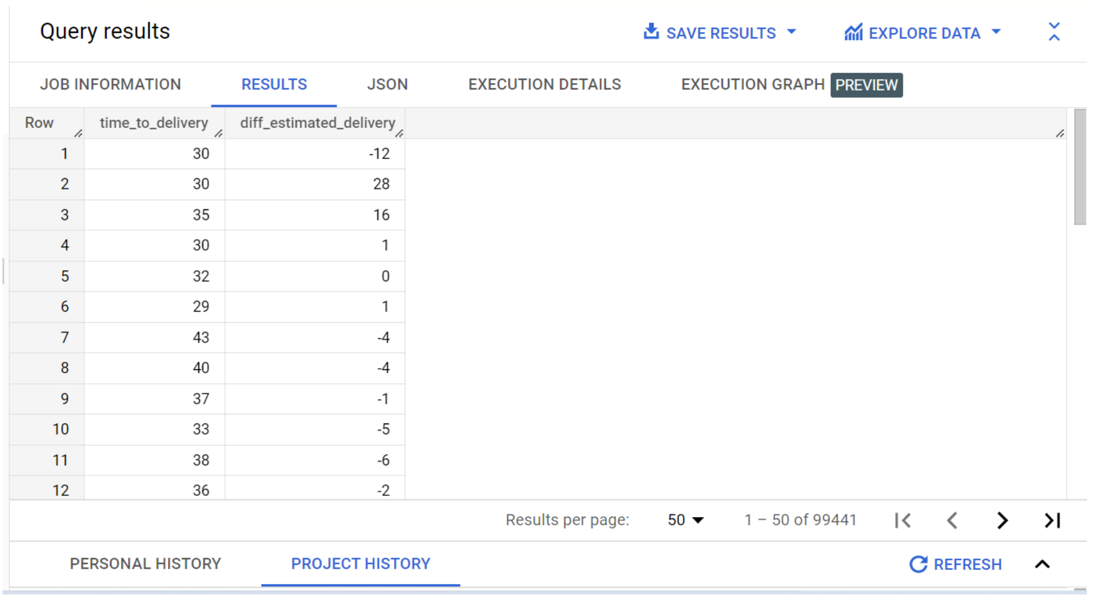
Row	order_id	purchase	delivered_date	estimate_deliver
1	e5215415bb6f76fe3b7cb6810...	1	null	null
2	3213c825fd43c3d2aa27fed77...	2	null	null
3	54282e97f61c23b78330c15b1...	2	null	null
4	2a06568281fa1a485b9ba5fac...	2	null	null
5	749309f96df588c9662196311...	2	null	null
6	2ad4df0af7a71d632dccc0129...	2	null	null
7	0b16257b1b7ee0922ea03954f...	2	null	null
8	e49e7ce1471b4693482d40c2b...	2	null	null
9	a0f2f8b0523207a13f02225bc8...	2	null	null
10	1e0f40bbfe787796d0c56e379...	2	null	null
11	e11dbf8f73df2ef44ac68c5dd9...	2	3	0
12	26f078c600b86ed6c916be331...	2	2	0

Results per page: 50 · 1 – 50 of 99441

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below :
   - time_to_delivery = order_delivered_customer_date- order_purchase_timestamp
   - diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

   SQL Syntax :

```
select
datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
 time_to_delivery,
datetime_diff(order_estimated_delivery_date,order_delivered_customer_date,
day) as diff_estimated_delivery
from `Target_Database.Orders`;
```


Query results

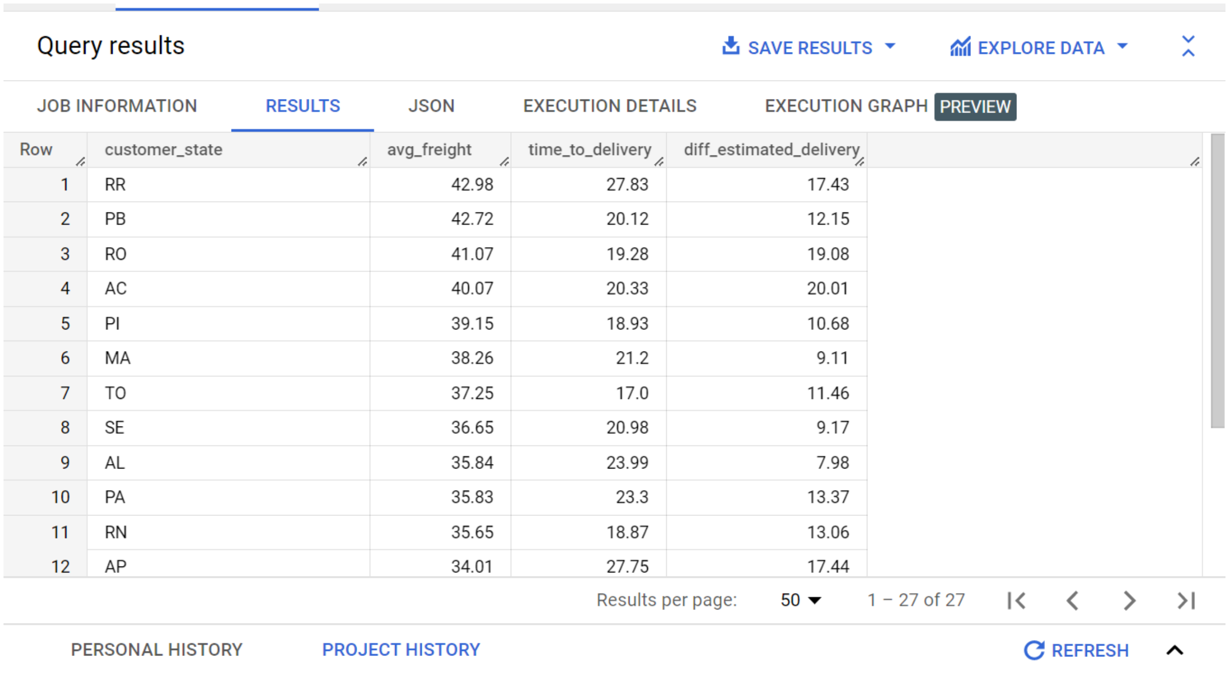
Row	time_to_delivery	diff_estimated_delivery
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5
11	38	-6
12	36	-2

Results per page: 50 · 1 – 50 of 99441

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

SQL Syntax :

```
with mean_data as
(select distinct c.customer_state,
round(avg(ot.freight_value) over(partition by c.customer_state),2)      as
avg_freight,
round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp,day))
over(partition by c.customer_state),2)
as time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,
order_delivered_customer_date,day))
over(partition by c.customer_state),2)as diff_estimated_delivery
from `Target_Database.Orders` AS o join `Target_Database.Customers` AS c on
o.customer_id=c.customer_id
join `Target_Database.Order_items` AS ot on ot.order_id=o.order_id)
select * from mean_data order by avg_freight desc, time_to_delivery desc,
diff_estimated_delivery desc;
```

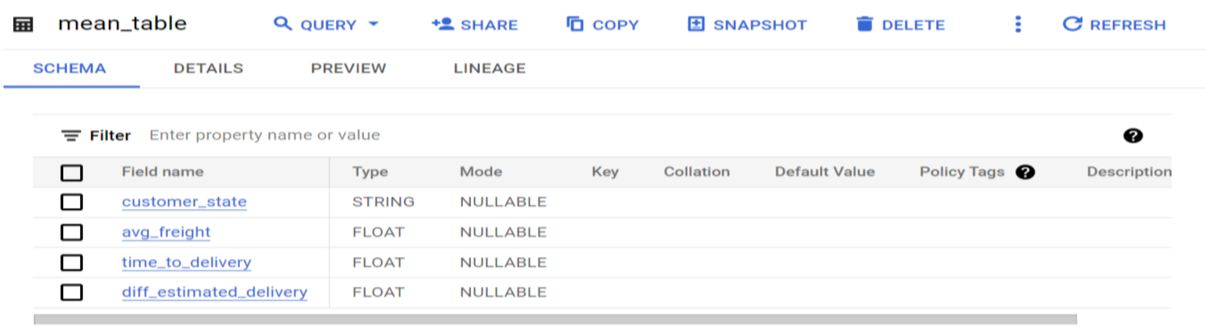

Query results

Row	customer_state	avg_freight	time_to_delivery	diff_estimated_delivery
1	RR	42.98	27.83	17.43
2	PB	42.72	20.12	12.15
3	RO	41.07	19.28	19.08
4	AC	40.07	20.33	20.01
5	PI	39.15	18.93	10.68
6	MA	38.26	21.2	9.11
7	TO	37.25	17.0	11.46
8	SE	36.65	20.98	9.17
9	AL	35.84	23.99	7.98
10	PA	35.83	23.3	13.37
11	RN	35.65	18.87	13.06
12	AP	34.01	27.75	17.44

Results per page: 50 1 – 27 of 27

4. Sort the data to get the following:

- Data saved as a new table – Mean_table to get the following analysis .


mean_table

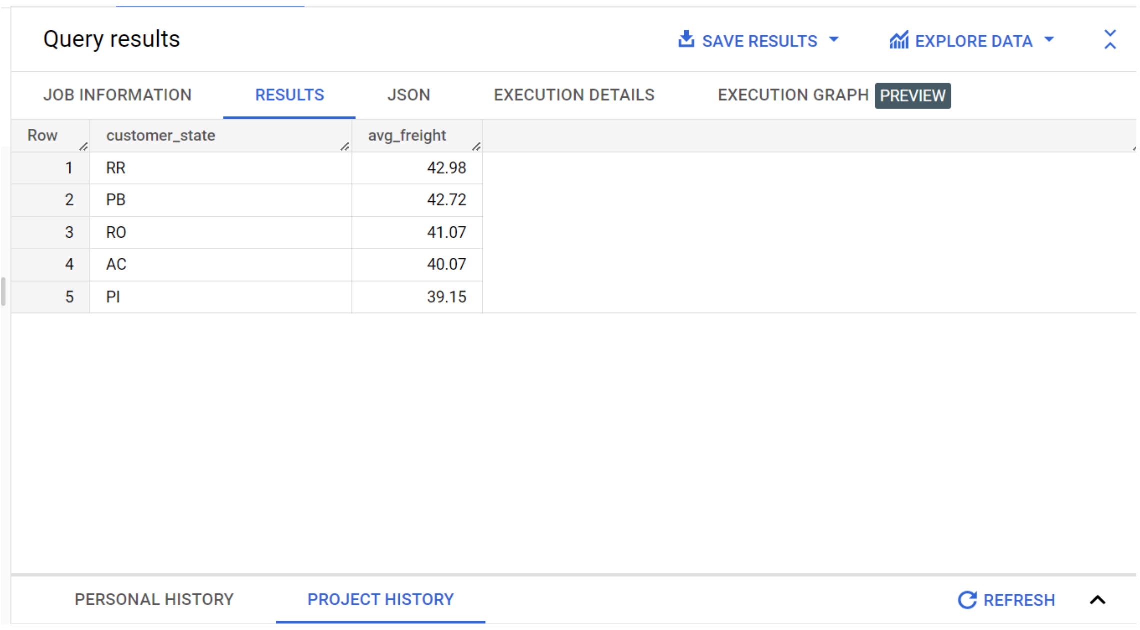
Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
customer_state	STRING	NULLABLE					
avg_freight	FLOAT	NULLABLE					
time_to_delivery	FLOAT	NULLABLE					
diff_estimated_delivery	FLOAT	NULLABLE					

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

SQL Syntax :

```
select customer_state,
avg_freight
from `Target_Database.mean_table`
order by avg_freight desc
limit 5 ;
```

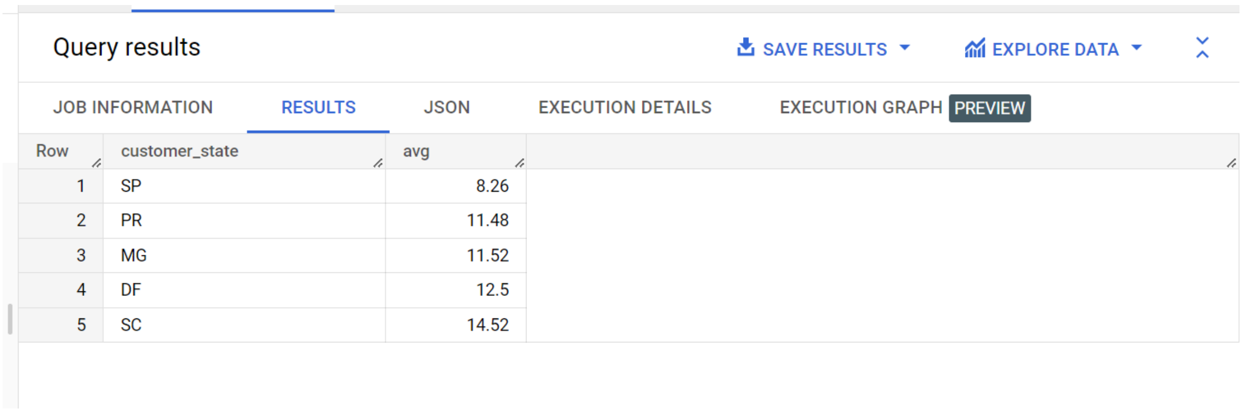


Row	customer_state	avg_freight
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

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

SQL Syntax :

```
select customer_state,
avg(time_to_delivery) as avg
from `Target_Database.mean_table`
group by customer_state
order by avg asc
limit 5 ;
```

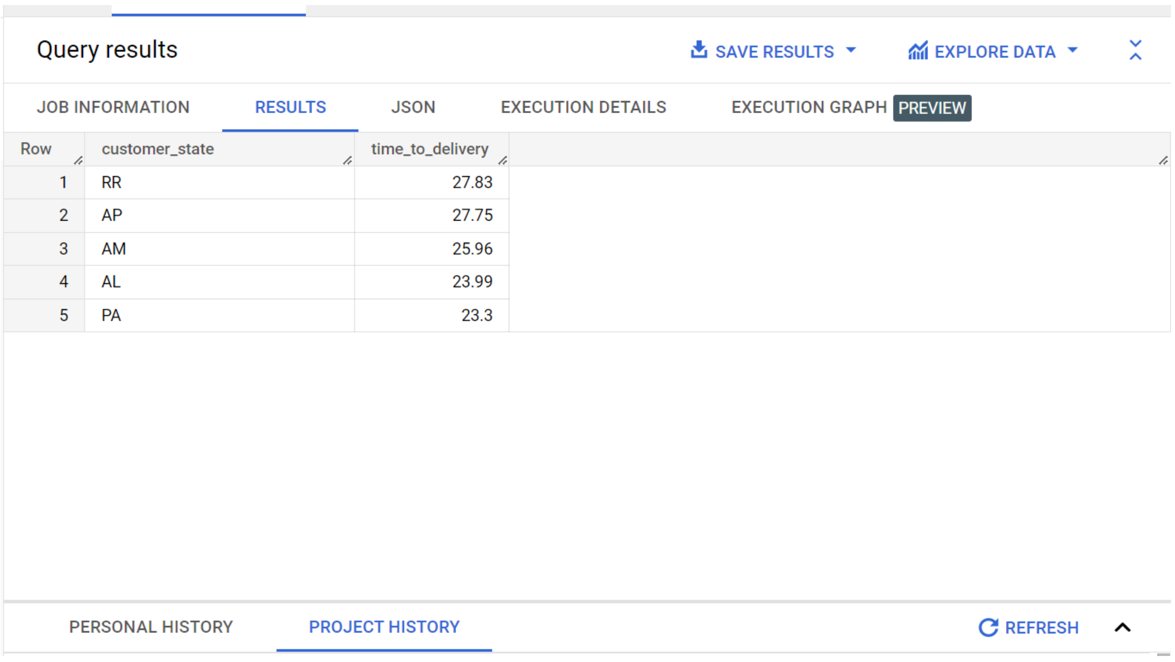


Row	customer_state	avg
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

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

SQL Syntax :

```
SELECT
customer_state,
time_to_delivery
FROM `Target_Database.mean_table`
where time_to_delivery > diff_estimated_delivery
ORDER BY time_to_delivery DESC
LIMIT 5 ;
```



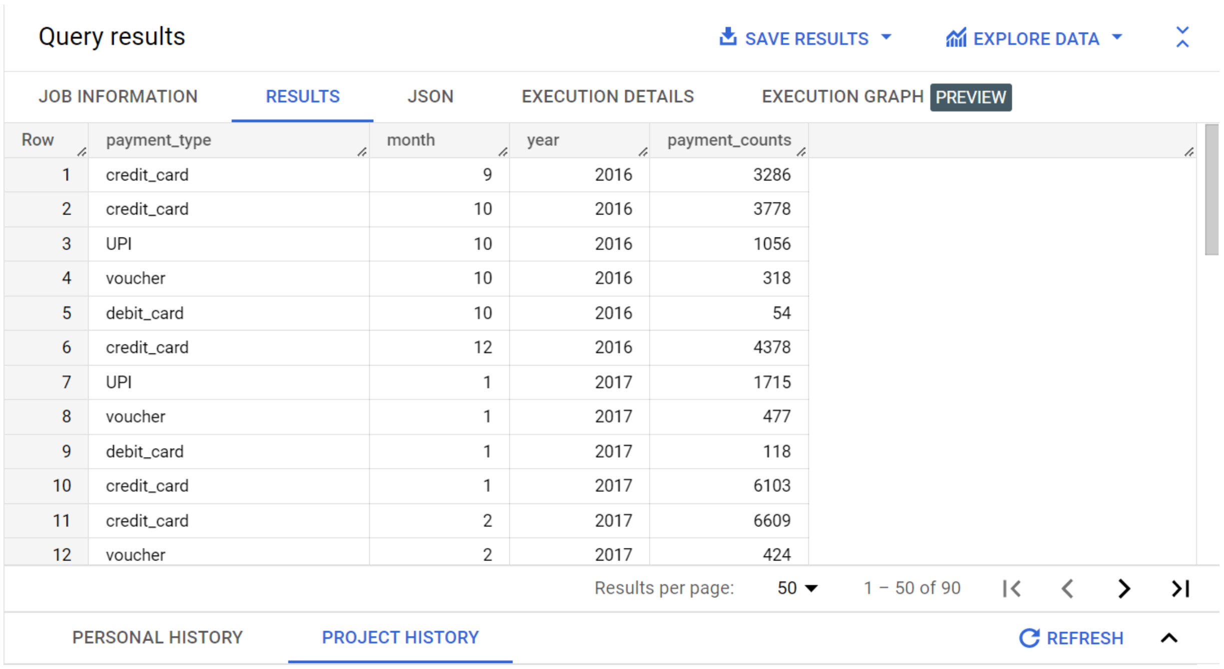
Row	customer_state	time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

VI. Payment type analysis :

1. Month over Month count of orders for different payment types :

SQL Syntax :

```
select
distinct p.payment_type,
extract(month from o.order_purchase_timestamp) as month,
extract(year from o.order_purchase_timestamp) as year,
count(o.order_id) over (partition by p.payment_type,
extract(month from o.order_purchase_timestamp)) as payment_counts
from `Target_Database.Orders` as o
join `Target_Database.Payments` as p
on o.order_id = p.order_id
order by year, month;
```



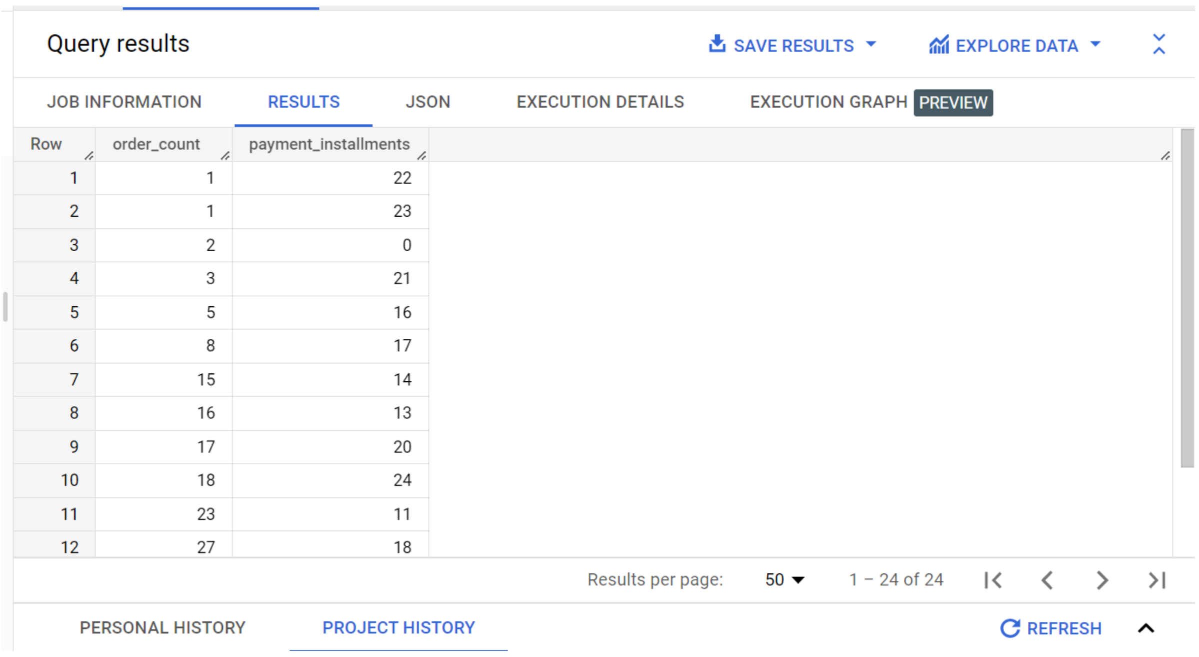
Row	payment_type	month	year	payment_counts
1	credit_card	9	2016	3286
2	credit_card	10	2016	3778
3	UPI	10	2016	1056
4	voucher	10	2016	318
5	debit_card	10	2016	54
6	credit_card	12	2016	4378
7	UPI	1	2017	1715
8	voucher	1	2017	477
9	debit_card	1	2017	118
10	credit_card	1	2017	6103
11	credit_card	2	2017	6609
12	voucher	2	2017	424

Results per page: 50 1 – 50 of 90

2. Count of orders based on the no. of payment installments :

SQL Syntax :

```
select
count(order_id) as order_count,
payment_installments
from `Target_Database.Payments`
group by payment_installments
order by order_count ;
```



Row	order_count	payment_installments
1	1	22
2	1	23
3	2	0
4	3	21
5	5	16
6	8	17
7	15	14
8	16	13
9	17	20
10	18	24
11	23	11
12	27	18

Results per page: 50 1 – 24 of 24

VII. Actionable Insights :

- Definitely there is a Growing trend in Brazil E-commerce market based on total sales happened YoY from 2016 to 2018 by 12k % increase in sales. As per my monthly wise study of growth pattern in sales there are high and low accordingly.
- Highest total_sales was mostly in the month of November and we can see some seasonality with peeks in months of January, March, April, May and November.
- As we can analyse by the results that Brazilians are mostly tend to buy in the afternoon.
- Top 3 customers count accounts from State of SP, RJ, MG – which accounts for 60% of sales
- Count of orders from 2016 to 2018 got a profitable rise in number.

VIII. Recommendations :

- Brazilians people were consistently buying and increasing the sales in all states. So, we have to balance our concentration on high region customers and also on low region customers.
- We recommend to particularly offer some coupons or offers and peek season sale according to the sales_info we analysed in our study.
- SP,RJ and MG states support the highest sales so we can maintain the standard offers as mentioned above and the low sales state need to be concentrated more and have to prioritize accordingly by giving more convenient offers and payback.
- Delivery time should be concentrated and have to decrease the time to delivery.
- Brazilians are fond of shopping at afternoon and night so we got to know the right time to look up at servers and apps without any issues and interruption so that there shopping will be hassel-free.