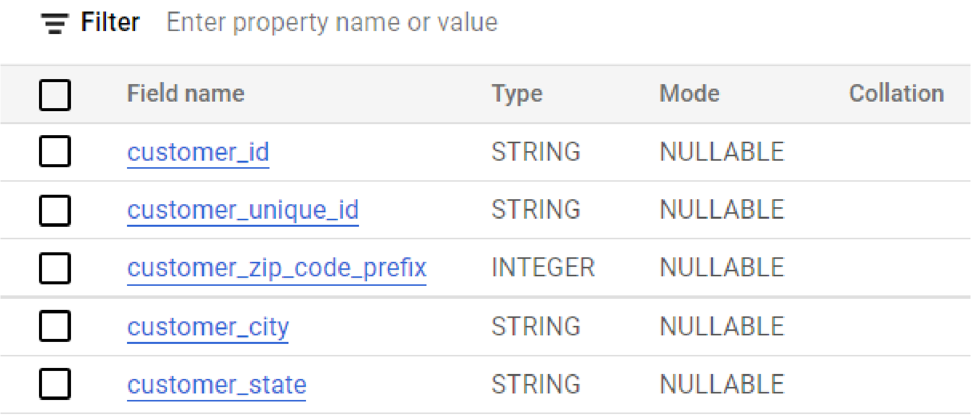
1) Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

1. Data type of columns in a table

We have below table -

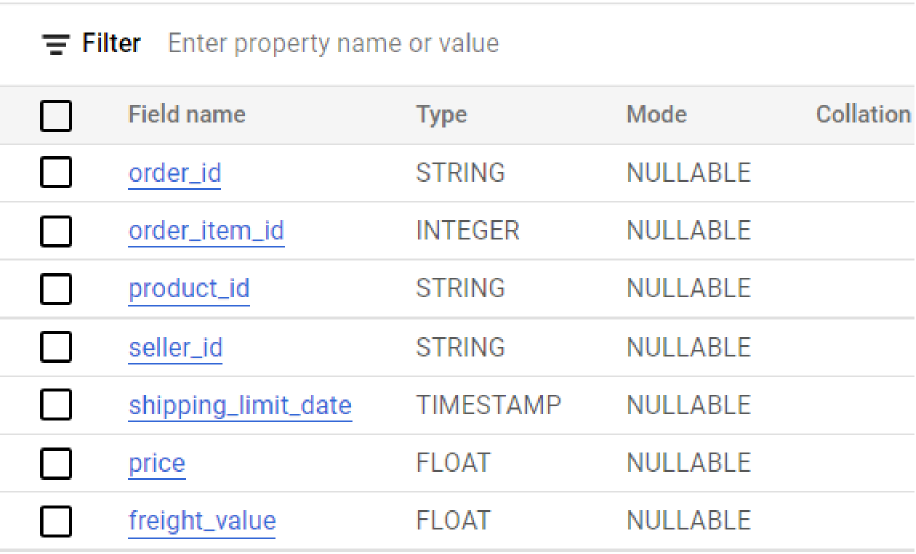
1. Customers



Filter Enter property name or value

□	Field name	Type	Mode	Collation
□	customer_id	STRING	NULLABLE	
□	customer_unique_id	STRING	NULLABLE	
□	customer_zip_code_prefix	INTEGER	NULLABLE	
□	customer_city	STRING	NULLABLE	
□	customer_state	STRING	NULLABLE	

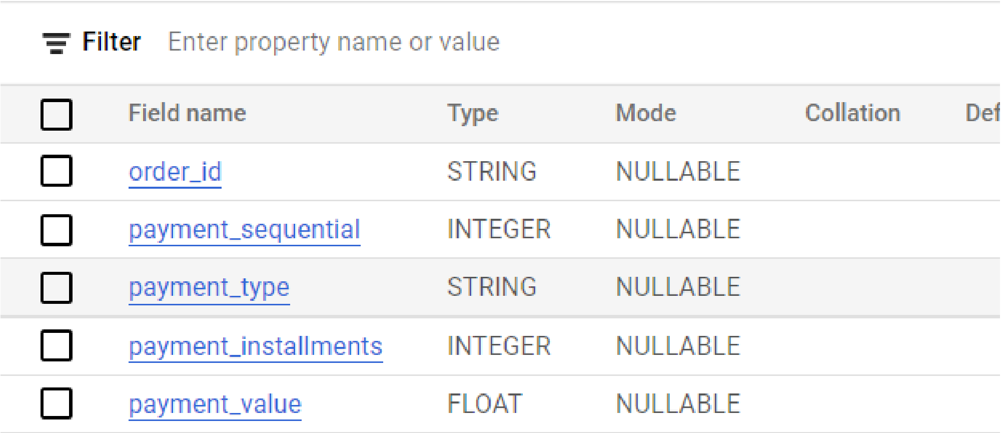
2. order_items



Filter Enter property name or value

□	Field name	Type	Mode	Collation
□	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	

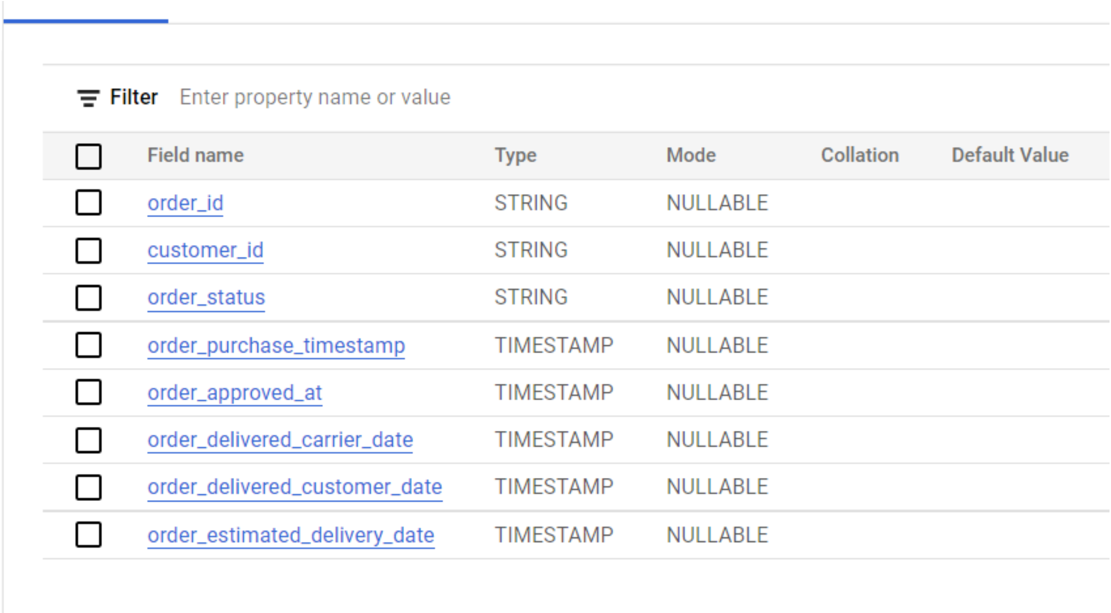
3. payments



Filter Enter property name or value

□	Field name	Type	Mode	Collation	Def
□	order_id	STRING	NULLABLE		
□	payment_sequential	INTEGER	NULLABLE		
□	payment_type	STRING	NULLABLE		
□	payment_installments	INTEGER	NULLABLE		
□	payment_value	FLOAT	NULLABLE		

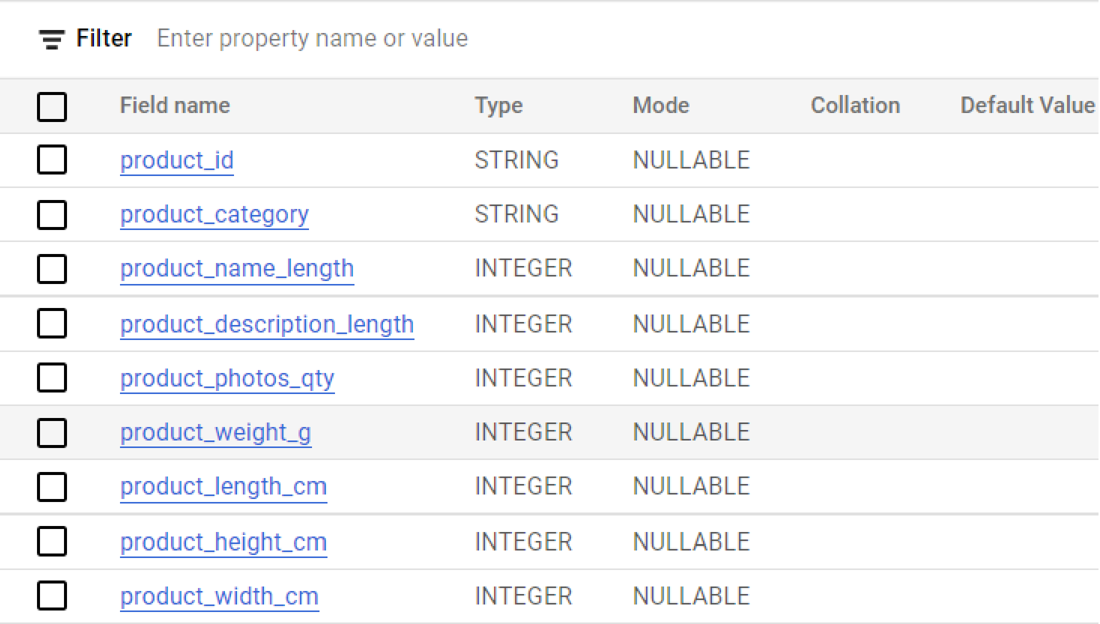
4. orders



Filter Enter property name or value

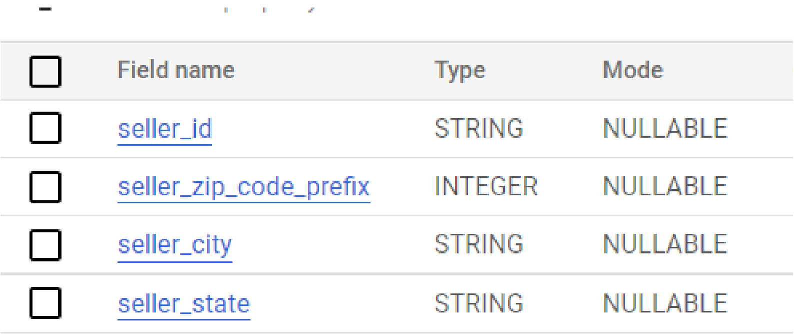
□	Field name	Type	Mode	Collation	Default Value
□	order_id	STRING	NULLABLE		
□	customer_id	STRING	NULLABLE		
□	order_status	STRING	NULLABLE		
□	order_purchase_timestamp	TIMESTAMP	NULLABLE		
□	order_approved_at	TIMESTAMP	NULLABLE		
□	order_delivered_carrier_date	TIMESTAMP	NULLABLE		
□	order_delivered_customer_date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

5. Products



□	Field name	Type	Mode	Collation	Default Value
□	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		

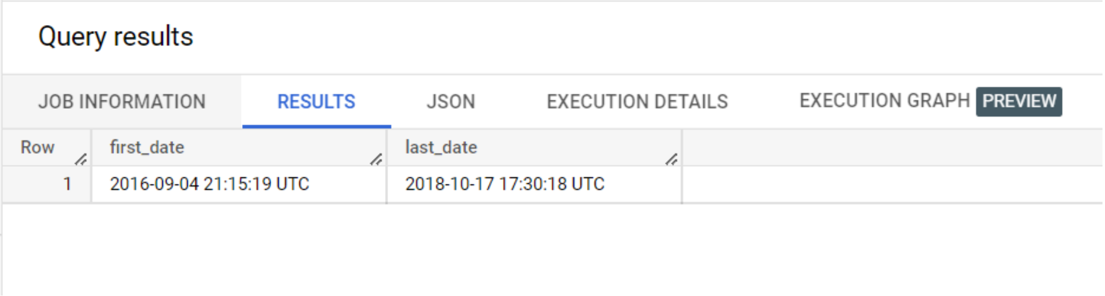
6. Sellers



□	Field name	Type	Mode
□	seller_id	STRING	NULLABLE
□	seller_zip_code_prefix	INTEGER	NULLABLE
□	seller_city	STRING	NULLABLE
□	seller_state	STRING	NULLABLE

2. Time period for which the data is given -
to get the time period, have use the max and min function on orders table

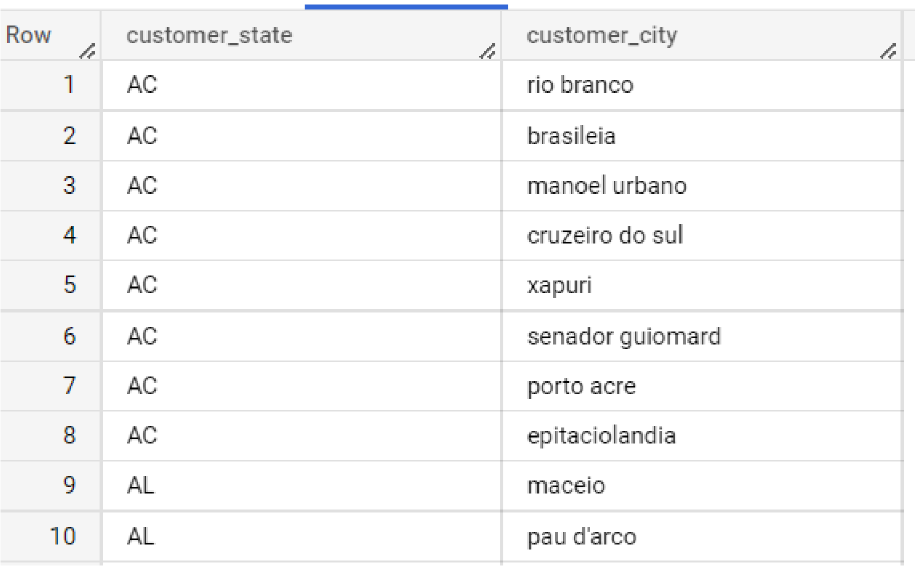
```
SELECT min(order_purchase_timestamp) AS first_date, max(order_purchase_timestamp) AS last_date FROM `target
-368217.Target.orders`;
```


Query results

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

Cities and States covered in the dataset -To check the cities we have use the customer table and order table to find out the data for the particular date range

```
SELECT c.customer_state, c.customer_city FROM target-368217.Target.orders AS o JOIN target-
368217.Target.customers AS c ON c.customer_id = o.customer_id where o.order_purchase_timestamp BETWEEN '201
6-09-04' AND '2018-10-17' GROUP BY c.customer_state, c.customer_city  ORDER BY c.customer_state;
```



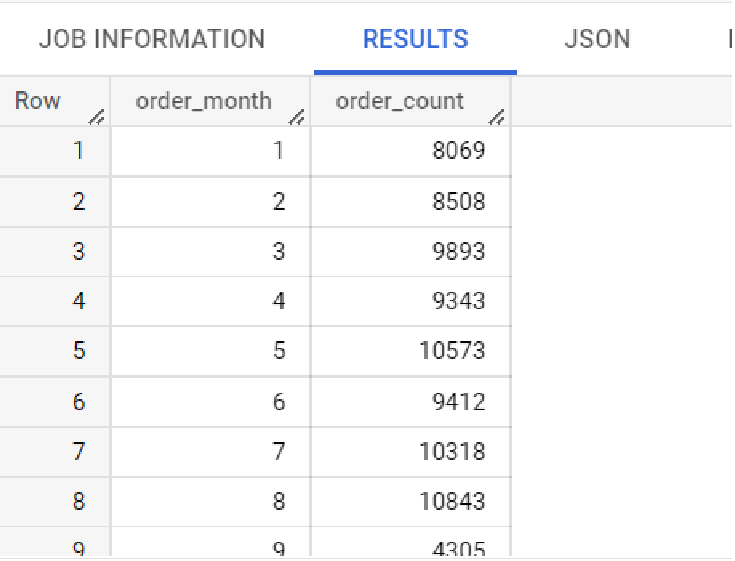
Row	customer_state	customer_city
1	AC	rio branco
2	AC	brasileia
3	AC	manoel urbano
4	AC	cruzeiro do sul
5	AC	xapuri
6	AC	senador guiomard
7	AC	porto acre
8	AC	epitaciolandia
9	AL	maceio
10	AL	pau d'arco

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?

As per the details checked the seasonable trend is from MAY- AUG which has the higest count. I have use the below query to check the trend

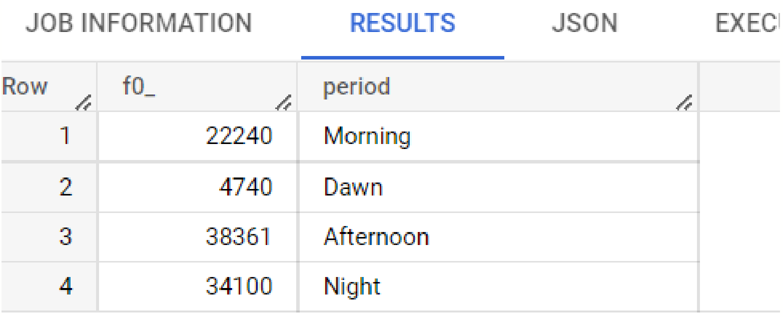
```
--
SELECT * FROM(
SELECT EXTRACT(month FROM order_purchase_timestamp) as order_month,count(*) AS order_count FROM target-
368217.Target.orders GROUP BY EXTRACT(month FROM order_purchase_timestamp)) as x order by  x.order_month,
x.order_month;
```



Row	order_month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305

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

```
SELECT
  COUNT(*),
  period
FROM (
  SELECT
    order_id,
    CASE
      WHEN TIME(order_purchase_timestamp) BETWEEN "00:00:00" AND "05:59:59" THEN "Dawn"
      WHEN TIME(order_purchase_timestamp) BETWEEN "06:00:00" AND "11:59:59" THEN "Morning"
      WHEN TIME(order_purchase_timestamp) BETWEEN "12:00:00" AND "17:59:59" THEN "Afternoon"
      WHEN TIME(order_purchase_timestamp) BETWEEN "18:00:00" AND "23:59:59" THEN "Night"
    END AS period
  FROM target-368217.Target.orders
  ) AS ord
GROUP BY period;
```



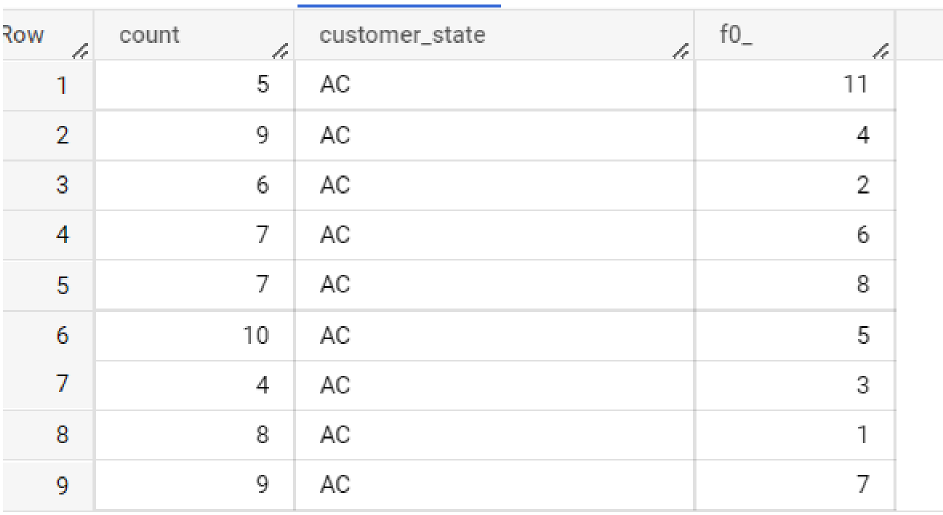
Row	f0_	period
1	22240	Morning
2	4740	Dawn
3	38361	Afternoon
4	34100	Night

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

1. Get month on month orders by region, states

Since we have region Brazil only have added the filter on states and month to get the month wise details -

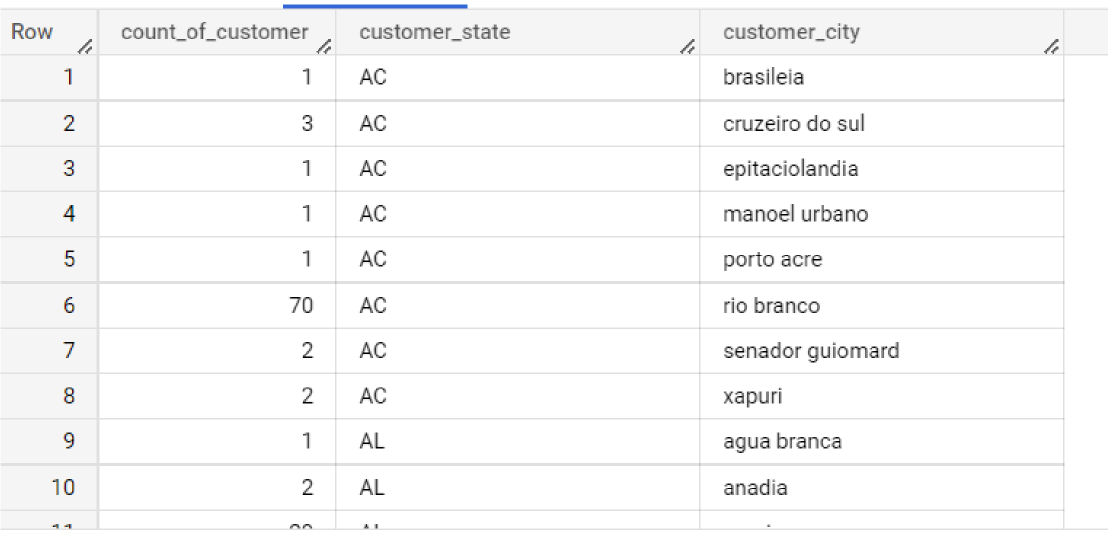
```
SELECT count(*) AS count, c.customer_state, EXTRACT(month from o.order_purchase_timestamp)  FROM target-368217.Target.orders AS o JOIN target-368217.Target.customers c ON c.customer_id = o.customer_id
GROUP BY c.customer_state, EXTRACT (month from o.order_purchase_timestamp)
ORDER BY c.customer_state
```



Row	count	customer_state	f0_
1	5	AC	11
2	9	AC	4
3	6	AC	2
4	7	AC	6
5	7	AC	8
6	10	AC	5
7	4	AC	3
8	8	AC	1
9	9	AC	7

2. How are customers distributed in Brazil -

```
SELECT count(DISTINCT(c.customer_id)) AS count_of_customer, c.customer_state, c.customer_city FROM  target-
368217.Target.customers c
GROUP BY c.customer_state, c.customer_city
ORDER BY c.customer_state, c.customer_city
```



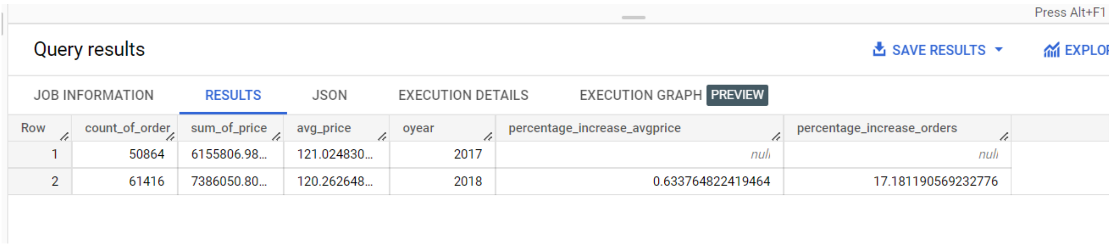
Row	count_of_customer	customer_state	customer_city
1	1	AC	brasileia
2	3	AC	cruzeiro do sul
3	1	AC	epitaciolandia
4	1	AC	manoel urbano
5	1	AC	porto acre
6	70	AC	rio branco
7	2	AC	senador guiomard
8	2	AC	xapuri
9	1	AL	agua branca
10	2	AL	anadia

4.Impact on Economy: Analyze the money movemented 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)

```
with  year_wise_order AS
(
SELECT oi.order_id,oi.price, EXTRACT(year from order_purchase_timestamp) as oyear FROM target-
368217.Target.orders AS o JOIN target-
368217.Target.order_items AS oi ON o.order_id=oi.order_id  WHERE order_purchase_timestamp BETWEEN '2017-01-
01' AND '2018-10-31'
)

SELECT count(m.order_id) AS count_of_order,sum(m.price) AS sum_of_price ,avg(m.price) AS avg_price, m.oyear,
-(avg(m.price) -
 (LAG(avg(m.price)) OVER (ORDER BY m.oyear)))/avg (m.price) * 100 AS percentage_increase_avgprice,
(count(m.order_id) -
 (LAG(count(m.order_id)) OVER (ORDER BY m.oyear)))/count (m.order_id) * 100 AS percentage_increase_orders
 FROM year_wise_order as m GROUP BY m.oyear order by m.oyear;
```



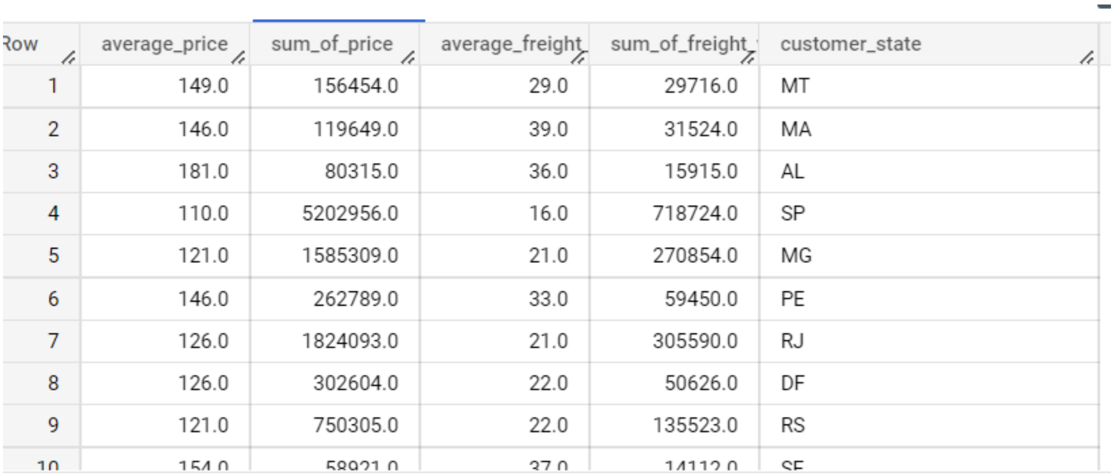
Query results

RESULTS

Row	count_of_order	sum_of_price	avg_price	oyear	percentage_increase_avgprice	percentage_increase_orders
1	50864	6155806.98...	121.024830...	2017	null	null
2	61416	7386050.80...	120.262648...	2018	0.633764822419464	17.181190569232776

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

```
SELECT ceil(avg(price)) AS average_price, ceil(sum(price)) AS sum_of_price,
ceil(avg(freight_value)) AS average_freight_value, ceil(sum(freight_value)) AS sum_of_freight_value, c.cust
omer_state
FROM target-368217.Target.order_items AS i JOIN target-368217.Target.orders AS o ON i.order_id=o.order_id
JOIN target-368217.Target.customers AS c ON c.customer_id=o.customer_id GROUP BY c.customer_state;
```

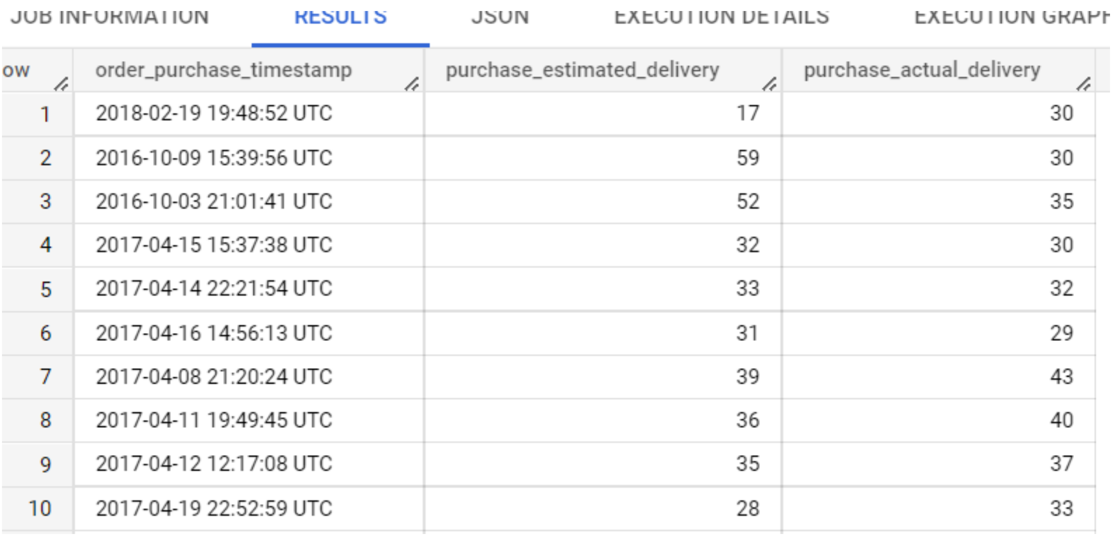


Row	average_price	sum_of_price	average_freight	sum_of_freight	customer_state
1	149.0	156454.0	29.0	29716.0	MT
2	146.0	119649.0	39.0	31524.0	MA
3	181.0	80315.0	36.0	15915.0	AL
4	110.0	5202956.0	16.0	718724.0	SP
5	121.0	1585309.0	21.0	270854.0	MG
6	146.0	262789.0	33.0	59450.0	PE
7	126.0	1824093.0	21.0	305590.0	RJ
8	126.0	302604.0	22.0	50626.0	DF
9	121.0	750305.0	22.0	135523.0	RS
10	154.0	58921.0	37.0	14112.0	SE

5.Analysis on sales, freight and delivery time

1) Calculate days between purchasing, delivering and estimated delivery

```
        SELECT order_purchase_timestamp,
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) AS
purchase_estimated_delivery,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day) AS
purchase_actual_delivery
FROM target-368217.Target.orders WHERE order_delivered_customer_date IS NOT NULL;
```

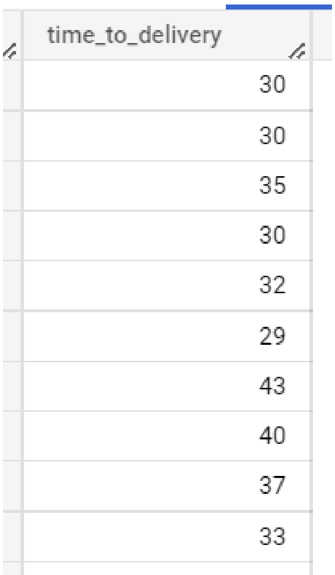


ow	order_purchase_timestamp	purchase_estimated_delivery	purchase_actual_delivery
1	2018-02-19 19:48:52 UTC	17	30
2	2016-10-09 15:39:56 UTC	59	30
3	2016-10-03 21:01:41 UTC	52	35
4	2017-04-15 15:37:38 UTC	32	30
5	2017-04-14 22:21:54 UTC	33	32
6	2017-04-16 14:56:13 UTC	31	29
7	2017-04-08 21:20:24 UTC	39	43
8	2017-04-11 19:49:45 UTC	36	40
9	2017-04-12 12:17:08 UTC	35	37
10	2017-04-19 22:52:59 UTC	28	33

2) Create columns:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

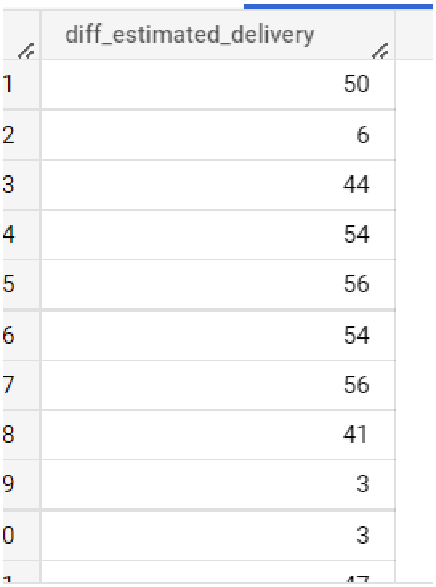
```
SELECT
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day) AS time_to_delivery
FROM target-368217.Target.orders WHERE order_delivered_customer_date IS NOT NULL;
```



time_to_delivery
30
30
35
30
32
29
43
40
37
33

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

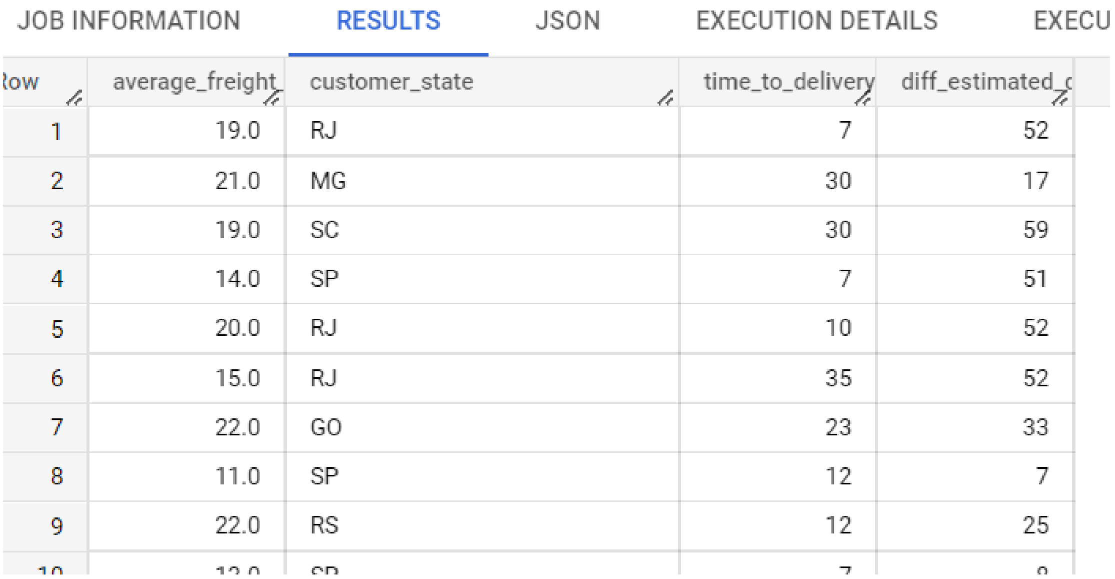
```
SELECT
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) AS diff_estimated_delivery
FROM target-368217.Target.orders;
```



	diff_estimated_delivery
1	50
2	6
3	44
4	54
5	56
6	54
7	56
8	41
9	3
0	3

3) Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery -

```
SELECT
ceil(avg(freight_value)) AS average_freight_value, c.customer_state,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day) AS time_to_delivery,
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) AS diff_estimated_delivery
FROM target-368217.Target.order_items AS i JOIN target-368217.Target.orders AS o ON i.order_id=o.order_id
JOIN target-368217.Target.customers AS c ON c.customer_id=o.customer_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state, DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day), DATE_DIF
F(order_estimated_delivery_date,order_purchase_timestamp, day);
```



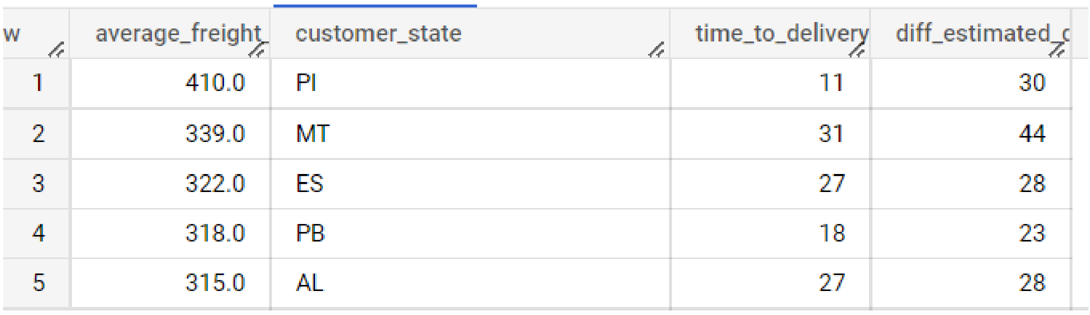
Row	average_freight	customer_state	time_to_delivery	diff_estimated_c
1	19.0	RJ	7	52
2	21.0	MG	30	17
3	19.0	SC	30	59
4	14.0	SP	7	51
5	20.0	RJ	10	52
6	15.0	RJ	35	52
7	22.0	GO	23	33
8	11.0	SP	12	7
9	22.0	RS	12	25

4) Sort the data to get the following:

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

Top 5 states with highest average freight value

```
SELECT * FROM (SELECT
ceil(avg(freight_value)) AS average_freight_value, c.customer_state,
DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day) AS time_to_delivery,
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) AS diff_estimated_delivery
FROM target-368217.Target.order_items AS i JOIN target-368217.Target.orders AS o ON i.order_id=o.order_id
JOIN target-368217.Target.customers AS c ON c.customer_id=o.customer_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state, DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day), DATE_DIF
F(order_estimated_delivery_date,order_purchase_timestamp, day)) as x order by x.average_freight_value desc
limit 5;
```

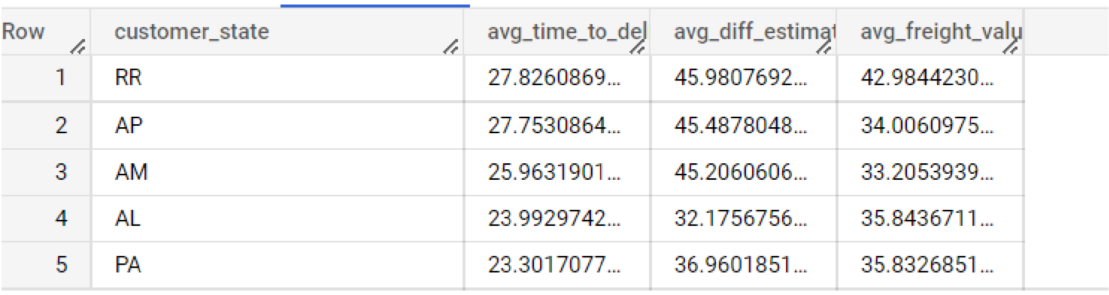


w	average_freight_	customer_state	time_to_delivery	diff_estimated_c
1	410.0	PI	11	30
2	339.0	MT	31	44
3	322.0	ES	27	28
4	318.0	PB	18	23
5	315.0	AL	27	28

Top 5 states with highest/lowest average time to delivery

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

```
SELECT c.customer_state, avg(x.time_to_delivery)  AS avg_time_to_delivery, avg(x.diff_estimated_delivery) A
S avg_diff_estimated_delivery, avg(oi.freight_value) AS avg_freight_value
FROM (SELECT customer_id, order_id, DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, day)
AS time_to_delivery,
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp, day) AS diff_estimated_delivery FROM targ
et-368217.Target.orders
WHERE order_estimated_delivery_date IS not null AND
order_purchase_timestamp is not null AND
order_estimated_delivery_date IS not null)as x JOIN target-
368217.Target.customers as c ON x.customer_id = c.customer_id JOIN target-
368217.Target.order_items oi ON x.order_id = oi.order_id group by c.customer_state
ORDER BY avg_time_to_delivery desc, avg_diff_estimated_delivery desc limit 5
```

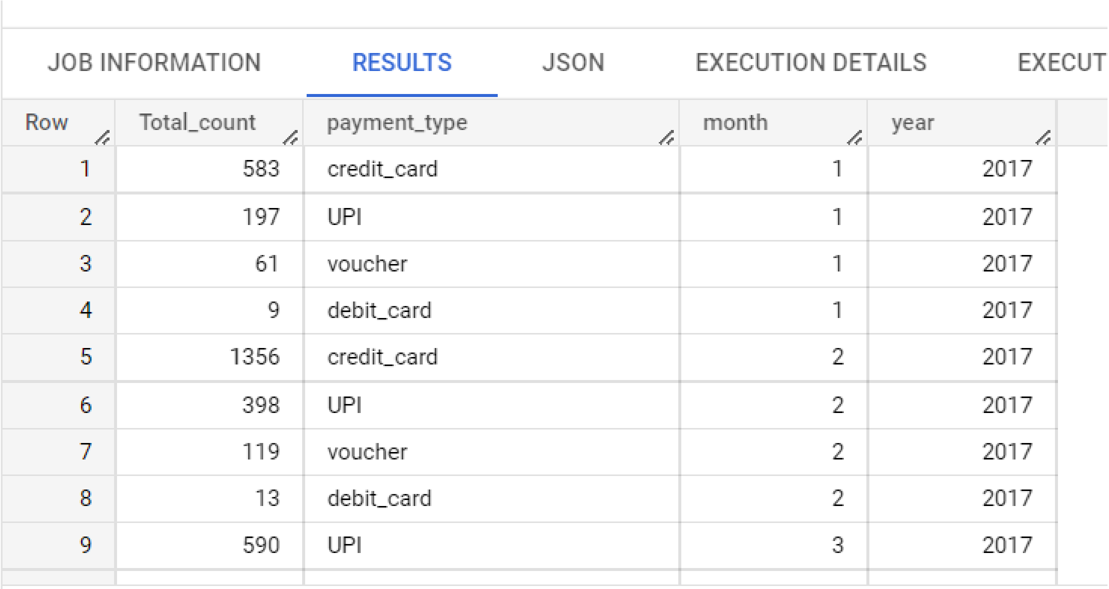


Row	customer_state	avg_time_to_del	avg_diff_estimat	avg_freight_valu
1	RR	27.8260869...	45.9807692...	42.9844230...
2	AP	27.7530864...	45.4878048...	34.0060975...
3	AM	25.9631901...	45.2060606...	33.2053939...
4	AL	23.9929742...	32.1756756...	35.8436711...
5	PA	23.3017077...	36.9601851...	35.8326851...

6. Payment type analysis:

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

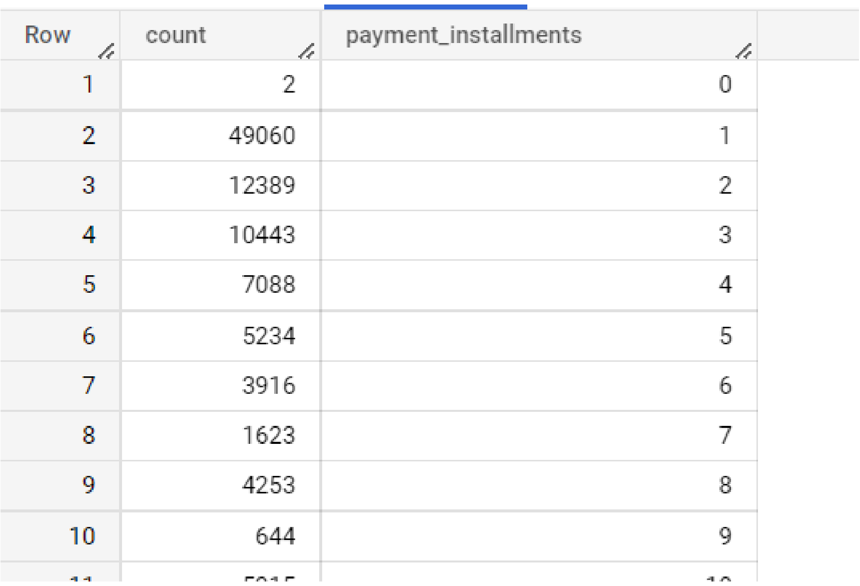
```
        SELECT * FROM (SELECT count(*) AS Total_count,p.payment_type, EXTRACT (month FROM o.order_purchase
_timestamp) AS month,EXTRACT (year FROM o.order_purchase_timestamp) AS year
FROM target-368217.Target.orders AS o JOIN target-
368217.Target.customers c ON c.customer_id = o.customer_id JOIN target-
368217.Target.payments p ON o.order_id = p.order_id WHERE o.order_purchase_timestamp  BETWEEN '2017-01-
01' AND '2018-08-
31' GROUP BY p.payment_type, EXTRACT (month FROM o.order_purchase_timestamp),  EXTRACT (year FROM o.order_p
urchase_timestamp)) AS x order by x.year, x.month;
```



Row	Total_count	payment_type	month	year
1	583	credit_card	1	2017
2	197	UPI	1	2017
3	61	voucher	1	2017
4	9	debit_card	1	2017
5	1356	credit_card	2	2017
6	398	UPI	2	2017
7	119	voucher	2	2017
8	13	debit_card	2	2017
9	590	UPI	3	2017

2. Distribution of payment installments and count of orders

```
SELECT count(DISTINCT(order_id)) AS count, payment_installments FROM target-
368217.Target.payments GROUP BY payment_installments;
```



Row	count	payment_installments
1	2	0
2	49060	1
3	12389	2
4	10443	3
5	7088	4
6	5234	5
7	3916	6
8	1623	7
9	4253	8
10	644	9
11	5015	10