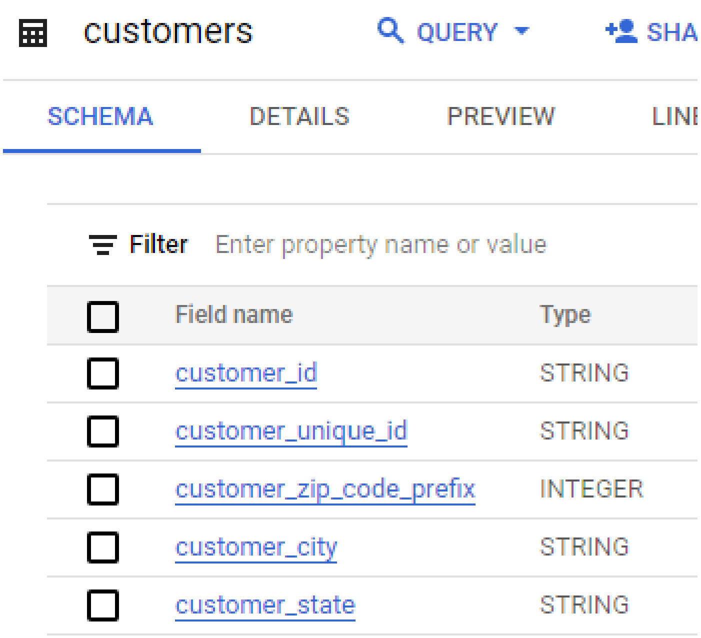
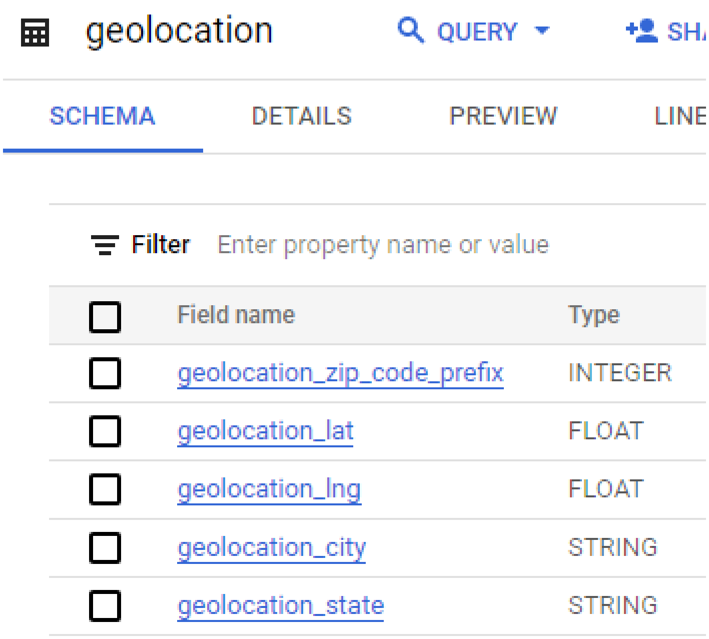
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

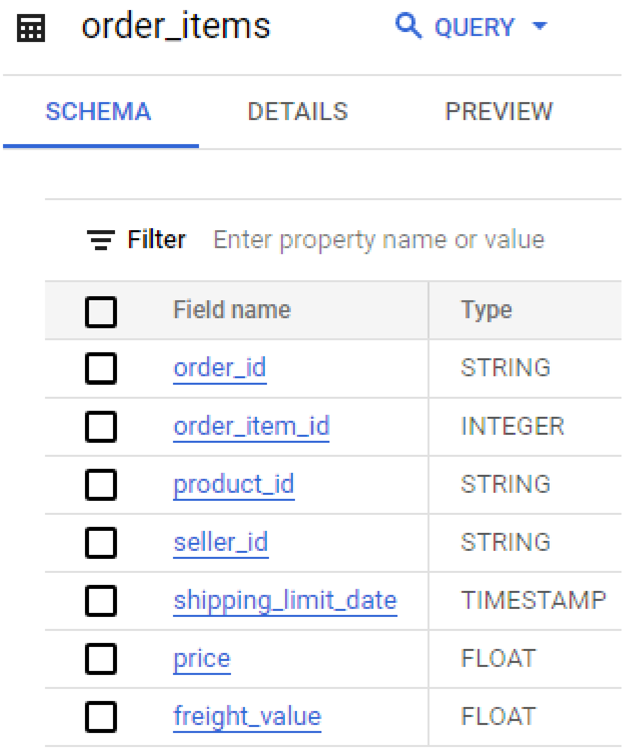
1. **Data type of columns in a table**



□	Field name	Type
□	customer_id	STRING
□	customer_unique_id	STRING
□	customer_zip_code_prefix	INTEGER
□	customer_city	STRING
□	customer_state	STRING

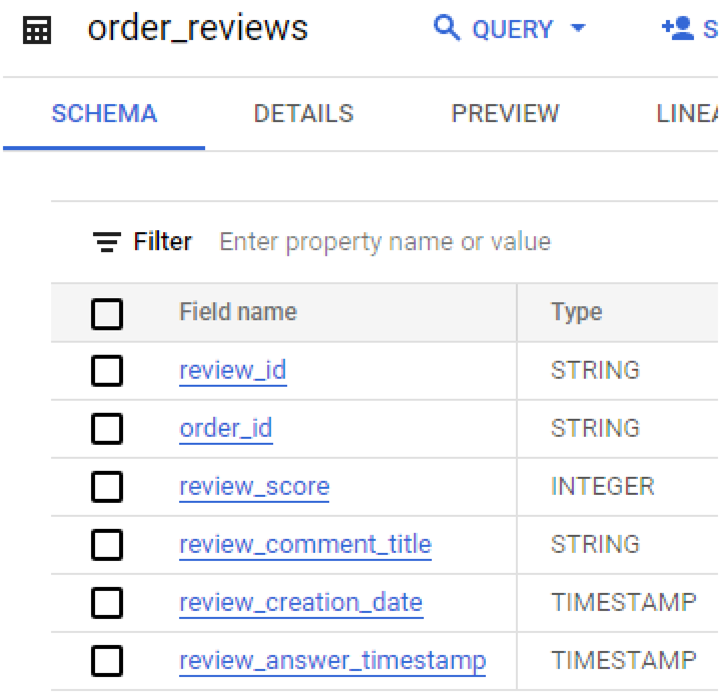


□	Field name	Type
□	geolocation_zip_code_prefix	INTEGER
□	geolocation_lat	FLOAT
□	geolocation_lng	FLOAT
□	geolocation_city	STRING
□	geolocation_state	STRING



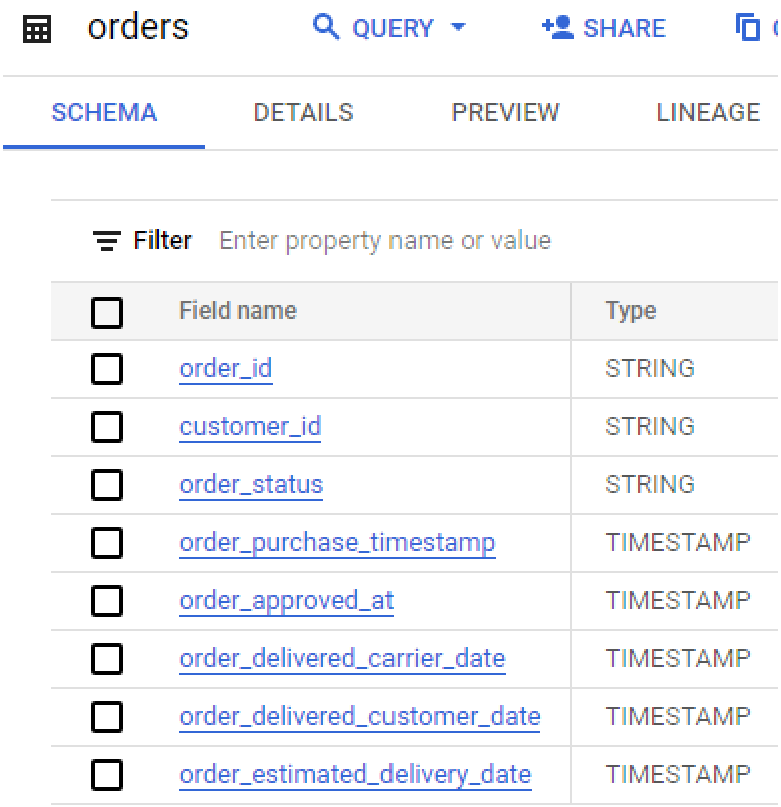
order_items

□	Field name	Type
□	order_id	STRING
□	order_item_id	INTEGER
□	product_id	STRING
□	seller_id	STRING
□	shipping_limit_date	TIMESTAMP
□	price	FLOAT
□	freight_value	FLOAT

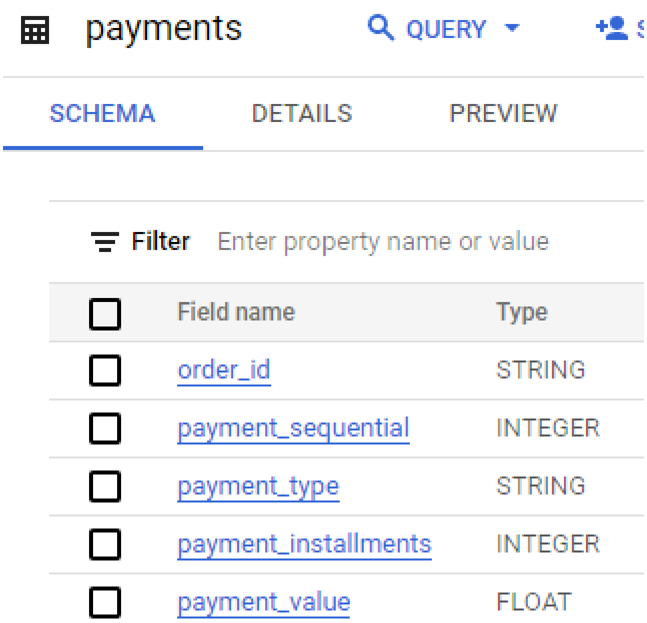


order_reviews

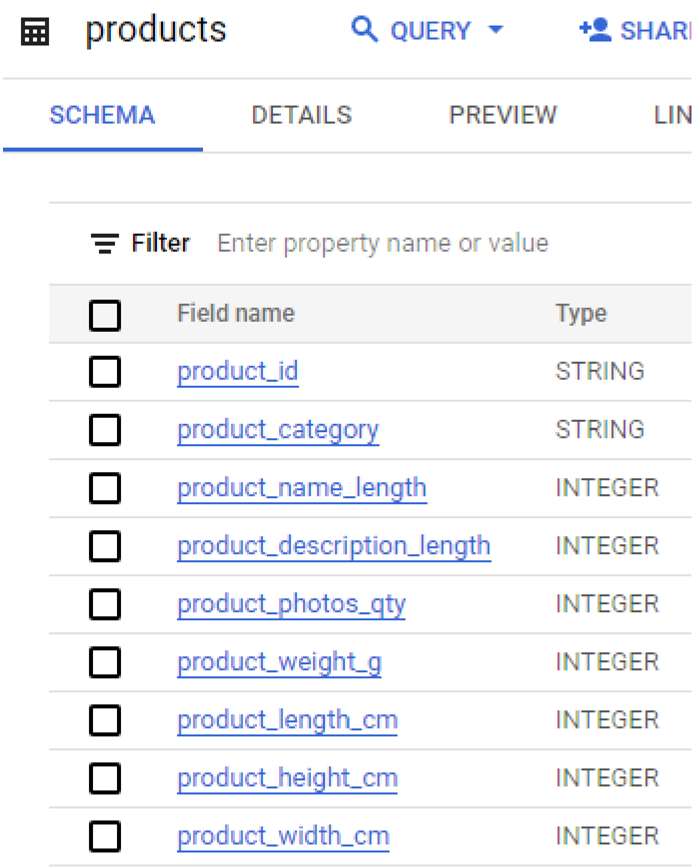
□	Field name	Type
□	review_id	STRING
□	order_id	STRING
□	review_score	INTEGER
□	review_comment_title	STRING
□	review_creation_date	TIMESTAMP
□	review_answer_timestamp	TIMESTAMP



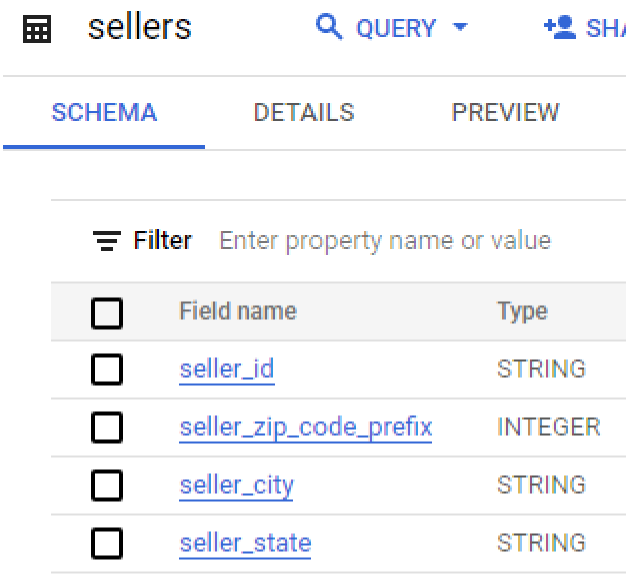
□	Field name	Type
□	order_id	STRING
□	customer_id	STRING
□	order_status	STRING
□	order_purchase_timestamp	TIMESTAMP
□	order_approved_at	TIMESTAMP
□	order_delivered_carrier_date	TIMESTAMP
□	order_delivered_customer_date	TIMESTAMP
□	order_estimated_delivery_date	TIMESTAMP



□	Field name	Type
□	order_id	STRING
□	payment_sequential	INTEGER
□	payment_type	STRING
□	payment_installments	INTEGER
□	payment_value	FLOAT



□	Field name	Type
□	product_id	STRING
□	product_category	STRING
□	product_name_length	INTEGER
□	product_description_length	INTEGER
□	product_photos_qty	INTEGER
□	product_weight_g	INTEGER
□	product_length_cm	INTEGER
□	product_height_cm	INTEGER
□	product_width_cm	INTEGER



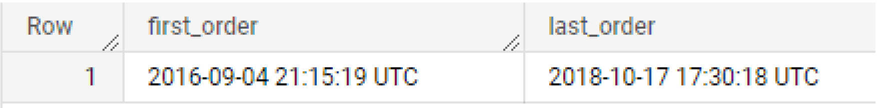
□	Field name	Type
□	seller_id	STRING
□	seller_zip_code_prefix	INTEGER
□	seller_city	STRING
□	seller_state	STRING

2. Time period for which the data is given

Query:

```
select min(order_purchase_timestamp) as first_order,
       max(order_purchase_timestamp) as last_order

from `sql_project.orders`
```

Result:



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

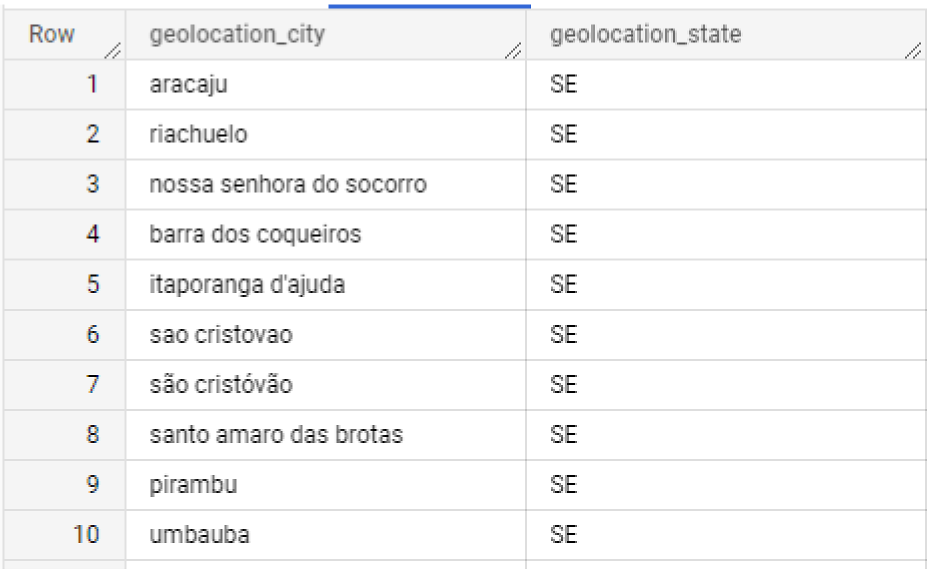
The Time period for which the data is given is between **2016-09-04 21:15:19 UTC** and **2018-10-17 17:30:18 UTC**

3. Cities and States of customers ordered during the given period

Query:

```
select distinct g.geolocation_city,
        g.geolocation_state

from `sql_project.customers` c
inner join `sql_project.geolocation`g
on g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
inner join `sql_project.orders` o
on o.customer_id = c.customer_id
```

Result:



Row	geolocation_city	geolocation_state
1	aracaju	SE
2	riachuelo	SE
3	nossa senhora do socorro	SE
4	barra dos coqueiros	SE
5	itaporanga d'ajuda	SE
6	sao cristovao	SE
7	são cristóvão	SE
8	santo amaro das brotas	SE
9	pirambu	SE
10	umbauba	SE

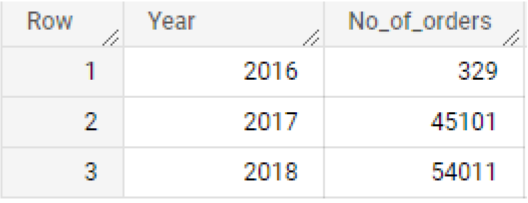
2. In-depth Exploration:

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?

Query:

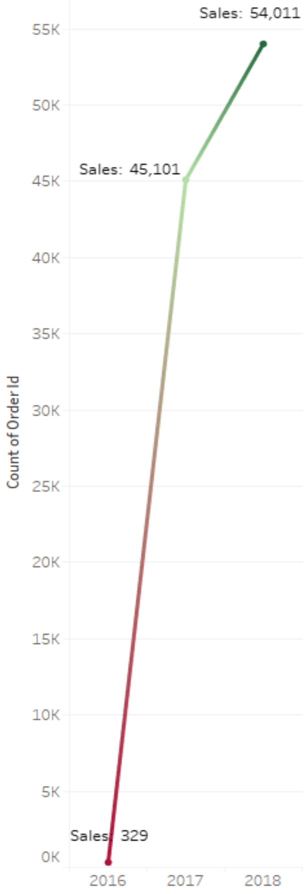
```
select extract( year from order_purchase_timestamp) as Year,
        count(order_id) as No_of_orders

from `sql_project.orders`

group by 1

order by 1
```

Result :



Row	Year	No_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Chart:

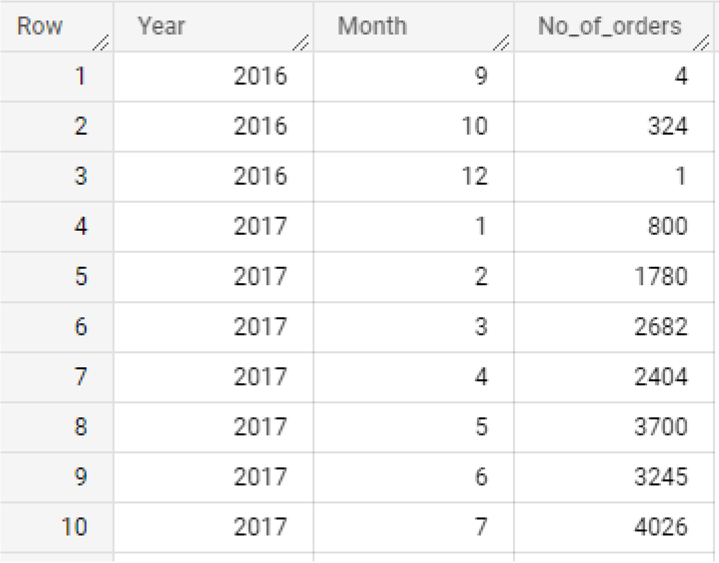


As you can see the sales for each year are increasing, so it is apparent that there is a growing trend in e-commerce in Brazil.

Query:

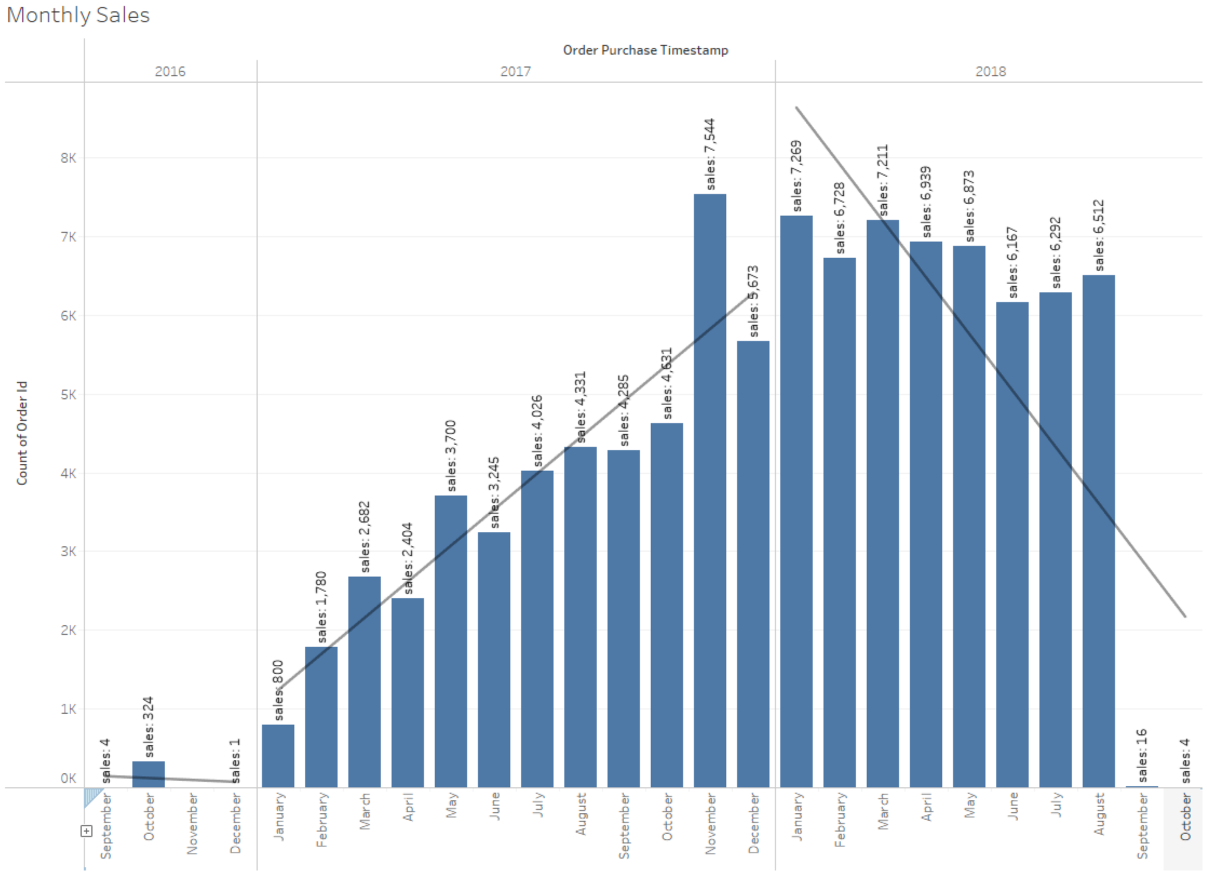
```
select  extract( year from order_purchase_timestamp) as Year,
        extract( month from order_purchase_timestamp) as Month,
        count( order_id) as No_of_orders

from `sql_project.orders`

group by 1,2

order by 1,2
```

Result:



Row	Year	Month	No_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Chart:



Actionable Insights:

As you can see the sales for each year are increasing, so it is apparent that there is a growing trend in e-commerce in Brazil. Sales have increased exponentially during 2016-2017 from 329 to 45101 and sales growth was slow during 2017-2018.

The monthly chart indicates the same, Sales gradually grew and peaked at the end of 2017 (November) and decreased the next month. then again increased by a lot in 2018(January). Seasonality peaks are during the month of November and January. From there the sales were on a downtrend.

The yearly sales number is on an uptrend, but the monthly sales number gradually increased till the end of 2017 and from there Sales are on a steady decline. The sales dropped by a lot on September 2018

Recommendations:

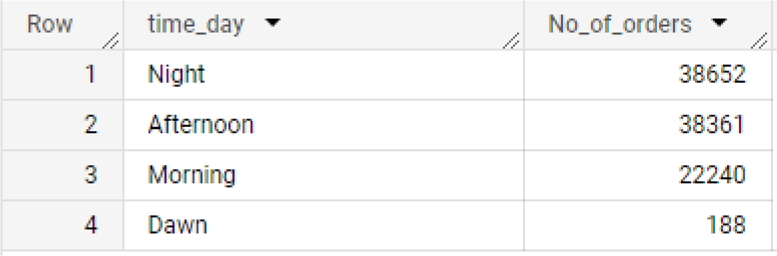
Even though there is a growing trend in e-commerce in Brazil, Sales growth slowed down during 2017-2018 compared to 2016-2017. The monthly sales chart shows the peak seasons trends during the month of November and January and also, during the months when sales are low. **We can provide discounts/offers to customers during the non-peak season to attract customers.**

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

Query:

```
select
        case when extract(time from o.order_purchase_timestamp) between '05:00:00' and
'05:59:59' then 'Dawn'
        when extract(time from o.order_purchase_timestamp) between '06:00:00' and
'11:59:59' then 'Morning'
        when extract(time from o.order_purchase_timestamp) between '12:00:00' and
'17:59:59' then 'Afternoon'
        when extract(time from o.order_purchase_timestamp) between '18:00:00' and
'23:59:59' or extract(time from o.order_purchase_timestamp) between '00:00:00' and
'04:59:59' then 'Night'  end as time_day,
        count(order_id) as No_of_orders

from `sql_project.customers` c
inner join `sql_project.orders` o
on o.customer_id = c.customer_id
group by time_day
order by 2 desc
```

Result:



Row	time_day	No_of_orders
1	Night	38652
2	Afternoon	38361
3	Morning	22240
4	Dawn	188

Actionable Insights:

Brazilian customers tend to buy more during the **Night** which is between the time period of **18:00:00** to **23:59:59** and also between **00:00:00** to **04:59:59** with **38652** orders. Dawn is when customer purchases are low.

Recommendations:

We can provide offers or discounts to customers who purchase during dawn so that customer purchases increase during dawn too.

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

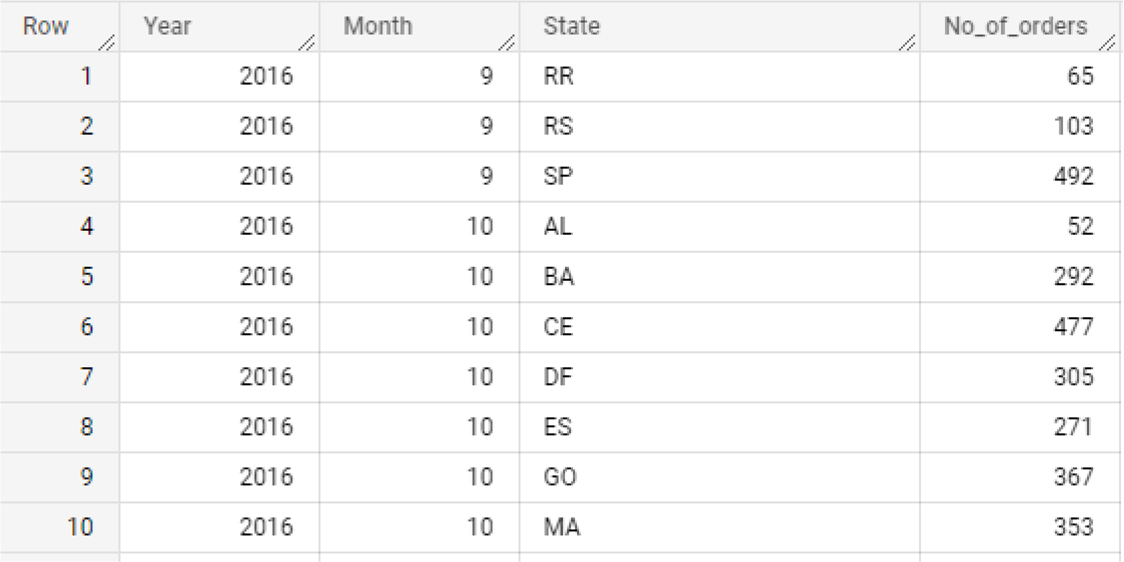
1. Get month on month orders by states

Query:

```
select extract(year from o.order_purchase_timestamp) as Year,
       extract(month from o.order_purchase_timestamp) as Month,
       g.geolocation_state as State,
       count(o.order_id) as No_of_orders

from `sql_project.orders` o
inner join `sql_project.customers`c
```

```
on o.customer_id = c.customer_id
inner join `sql_project.geolocation` g
on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix

group by 1,2,3

order by 1,2,3
```

Result:



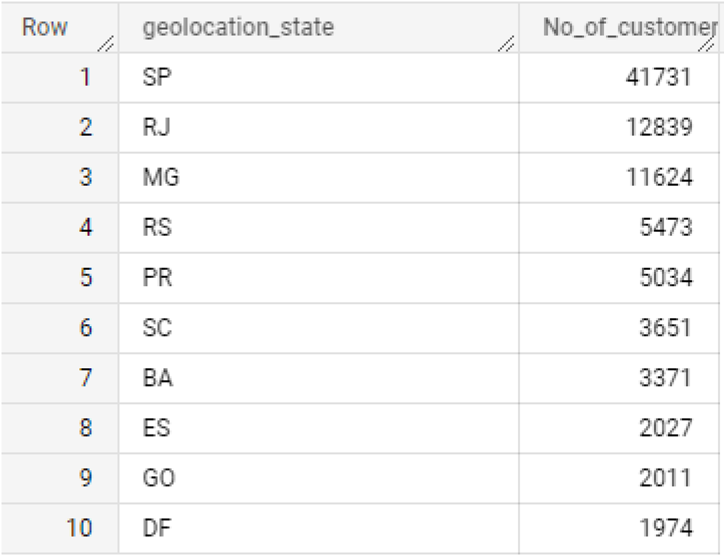
Row	Year	Month	State	No_of_orders
1	2016	9	RR	65
2	2016	9	RS	103
3	2016	9	SP	492
4	2016	10	AL	52
5	2016	10	BA	292
6	2016	10	CE	477
7	2016	10	DF	305
8	2016	10	ES	271
9	2016	10	GO	367
10	2016	10	MA	353

2. Distribution of customers across the states in Brazil

Query:

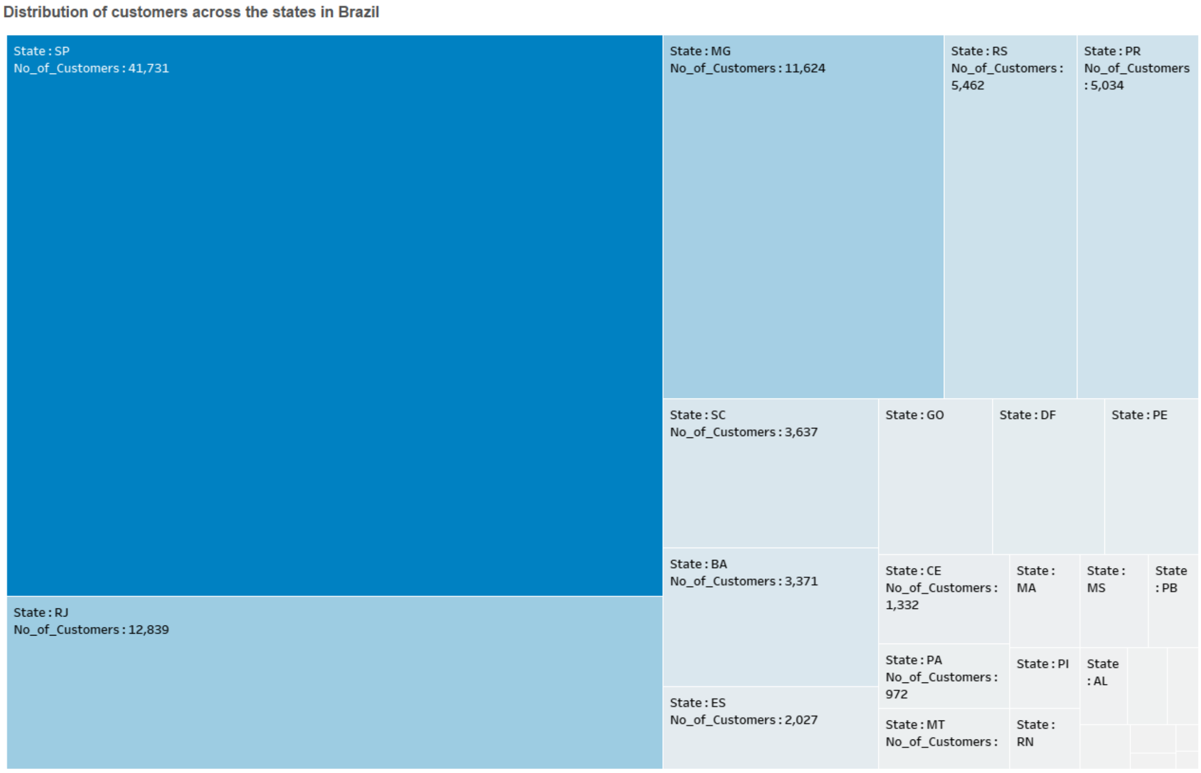
```
select g.geolocation_state,
    count( distinct c.customer_id) as No_of_customers

from `sql_project.customers` c
inner join `sql_project.geolocation` g
on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix

group by 1

order by 2 desc
```

Result:



Row	geolocation_state	No_of_customer
1	SP	41731
2	RJ	12839
3	MG	11624
4	RS	5473
5	PR	5034
6	SC	3651
7	BA	3371
8	ES	2027
9	GO	2011
10	DF	1974

Chart:



Actionable Insights:

Sau Paulo(SP) has the highest number of customers, all the other states have less than 13000 customers

Recommendations:

We can offer discounts and incentives for only new customers in the states with fewer customers to attract more customers

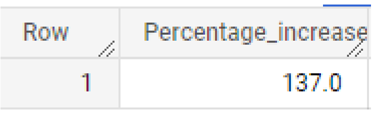
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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

Query:

```
SELECT
ROUND(((ty_sales - ly_sales) / ly_sales) * 100) AS Percentage_increase
FROM (
SELECT
SUM(CASE WHEN Year = 2018 AND Month BETWEEN 1 AND 8 THEN payment_value END) AS ty_sales,
SUM(CASE WHEN Year = 2017 AND Month BETWEEN 1 AND 8 THEN payment_value END) AS ly_sales
FROM (
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS Month,
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS Year,
p.payment_value
FROM
`sql_project.payments` p
INNER JOIN `sql_project.orders` o ON o.order_id = p.order_id
)
) AS subquery_alias;
```

Result:



Row	Percentage_increase
1	137.0

Actionable Insights:

The % increase in cost of orders from 2017 to 2018 including months between Jan to Aug only is 137%

Recommendations:

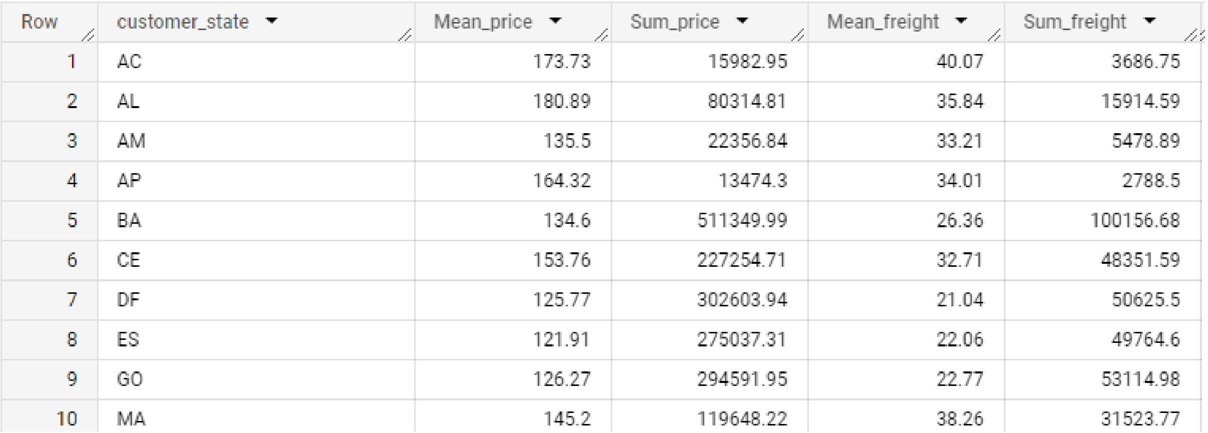
By examining the percentage increase in the cost of orders across different product categories, you can identify which categories are driving higher profit margins. Focus on promoting and expanding your offerings within these profitable categories to maximize profitability.

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

Query:

```
select c.customer_state,
       round(avg(ot.price),2) as Mean_price,
       round(sum(ot.price),2)  as Sum_price,
       round(avg(ot.freight_value),2) as Mean_freight,
       round(sum(ot.freight_value),2) as Sum_freight

from `sql_project.order_items` ot
inner join `sql_project.orders` o
on ot.order_id = o.order_id
inner join `sql_project.customers` c
on o.customer_id = c.customer_id

group by 1
order by 1
```

Result:



Row	customer_state	Mean_price	Sum_price	Mean_freight	Sum_freight
1	AC	173.73	15982.95	40.07	3686.75
2	AL	180.89	80314.81	35.84	15914.59
3	AM	135.5	22356.84	33.21	5478.89
4	AP	164.32	13474.3	34.01	2788.5
5	BA	134.6	511349.99	26.36	100156.68
6	CE	153.76	227254.71	32.71	48351.59
7	DF	125.77	302603.94	21.04	50625.5
8	ES	121.91	275037.31	22.06	49764.6
9	GO	126.27	294591.95	22.77	53114.98
10	MA	145.2	119648.22	38.26	31523.77

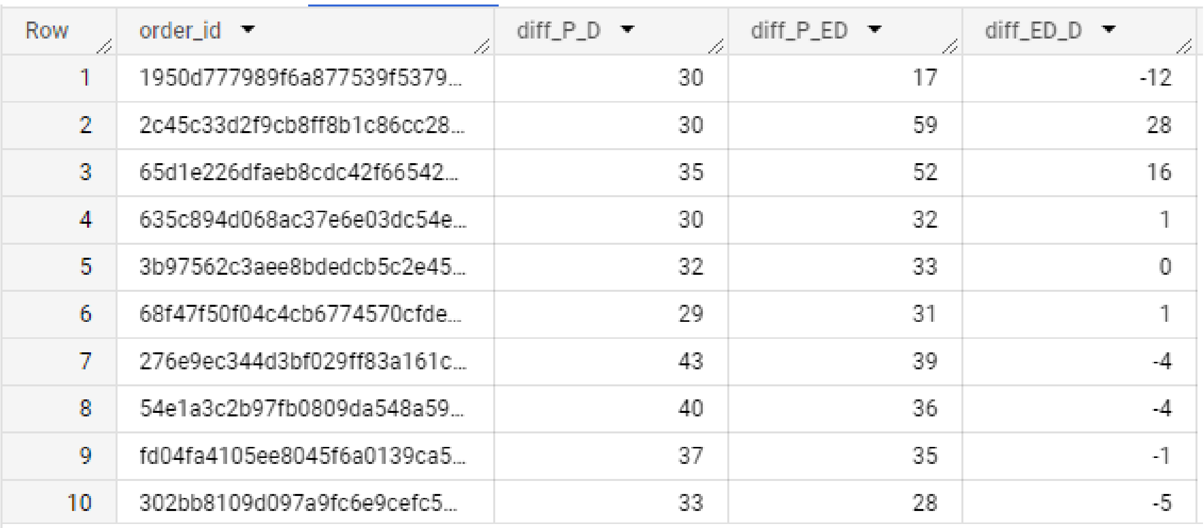
5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Query:

```
select distinct order_id,
        date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
diff_P_D,
        date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as
diff_P_ED,
        date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as
diff_ED_D

from `sql_project.orders`

where order_delivered_customer_date is not null
```

Result:



Row	order_id	diff_P_D	diff_P_ED	diff_ED_D
1	1950d777989f6a877539f5379...	30	17	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	59	28
3	65d1e226dfaeb8cdc42f66542...	35	52	16
4	635c894d068ac37e6e03dc54e...	30	32	1
5	3b97562c3aee8bdedcb5c2e45...	32	33	0
6	68f47f50f04c4cb6774570cfde...	29	31	1
7	276e9ec344d3bf029ff83a161c...	43	39	-4
8	54e1a3c2b97fb0809da548a59...	40	36	-4
9	fd04fa4105ee8045f6a0139ca5...	37	35	-1
10	302bb8109d097a9fc6e9cefc5...	33	28	-5

Actionable Insights:

Positive values in difference between estimated delivery and actual delivery (diff_ED_D) denote that the order was delivered before the estimated time. negative value denotes that the order was delivered after the estimated delivery time. If the value is zero it denotes that the order was delivered on the estimated delivery date.

Recommendations:

We can filter out the order_id where diff_ED_D is negative to find the orders where delivery was late and see if there are any supply chain issues and fix it

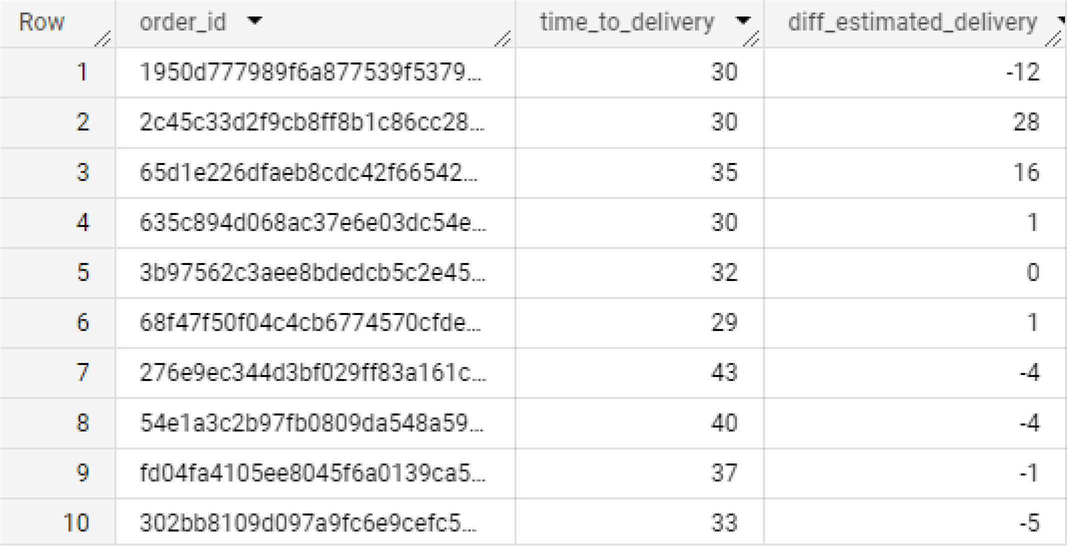
2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- time_to_delivery = order_delivered_customer_date-order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

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 `sql_project.orders`

where order_delivered_customer_date is not null
```

Result:



Row	order_id	time_to_delivery	diff_estimated_delivery
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

Actionable Insights:

diff_estimated_delivery indicates if a delivery is made before or after the estimated delivery date. A positive value denotes delivery before the estimated delivery and a negative value denotes delivery after the estimated delivery. If the value is zero it denotes that the order was delivered on the estimated delivery date.

Recommendations:

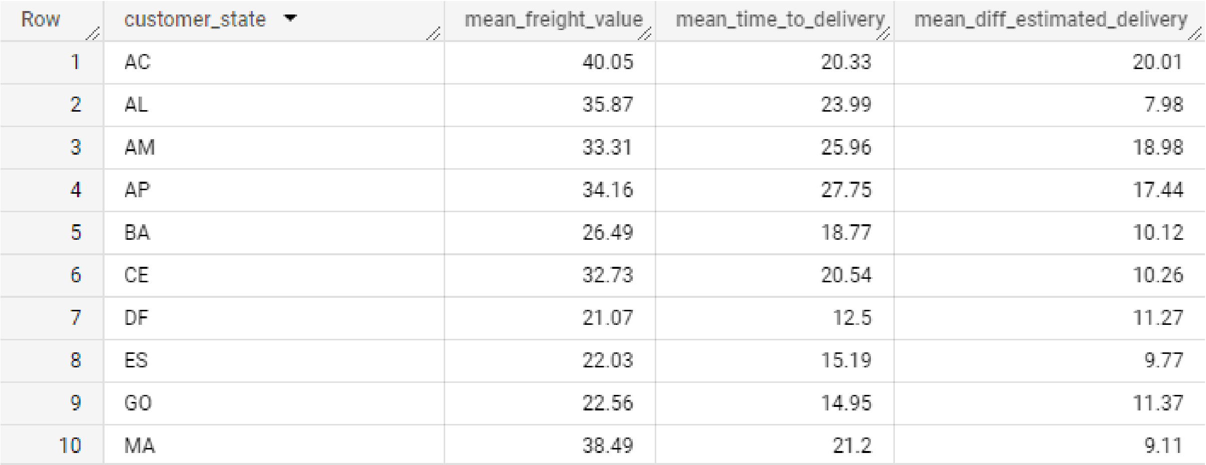
We can filter out the order_id where diff_estimated_delivery is negative to find the orders where delivery was late and find out what is the problem and fix it.

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

Query:

```
select c.customer_state,
        round(avg(oi.freight_value),2) as mean_freight_value,
        round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp,
day)),2) as  mean_time_to_delivery,
        round(avg(date_diff(order_estimated_delivery_date,
order_delivered_customer_date,day)),2) as mean_diff_estimated_delivery

from `sql_project.orders` o
inner join `sql_project.customers` c
on o.customer_id = c.customer_id
inner join `sql_project.order_items` oi
on oi.order_id = o.order_id

where order_delivered_customer_date is not null
group by 1
order by 1
```

Result:



Row	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AC	40.05	20.33	20.01
2	AL	35.87	23.99	7.98
3	AM	33.31	25.96	18.98
4	AP	34.16	27.75	17.44
5	BA	26.49	18.77	10.12
6	CE	32.73	20.54	10.26
7	DF	21.07	12.5	11.27
8	ES	22.03	15.19	9.77
9	GO	22.56	14.95	11.37
10	MA	38.49	21.2	9.11

Actionable Insights:

For each state in Brazil we have obtained the follows:

mean_freight_value is the average amount paid to a carrier company for the transportation of goods from the point of origin to an agreed location for each state in Brazil

mean_time_to_delivery is the average days taken to deliver an item from the time of the purchase to delivery for each state in Brazil

mean_diff_estimated_delivery is the average date difference from the estimated delivery to the actual delivery date for each state in Brazil

Recommendations:

We can filter out states where mean_freight_value is high and try to reduce the cost so that the profit is increased

We can filter out states where mean_time_to_delivery is high and try to reduce the wait time for the customer. More customers will place orders if delivery is quick.

We can filter out states where mean_diff_estimated_delivery is high and try to reduce it by delivering the orders closer to the estimated delivery date.

4.Sort the data to get the following:

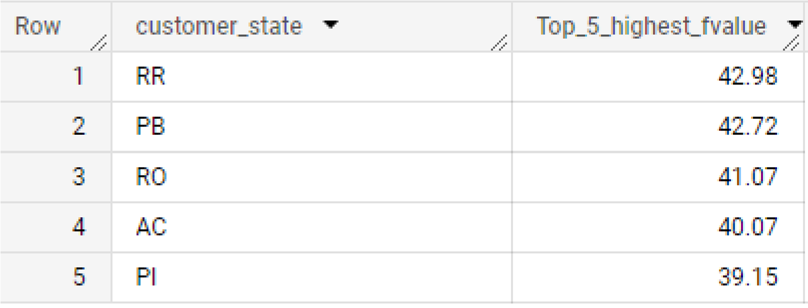
5.Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Highest

Query:

```
select c.customer_state,
        round(avg(oi.freight_value),2) as Top_5_highest_fvalue,

from `sql_project.orders` o
inner join `sql_project.customers` c
on o.customer_id = c.customer_id
inner join `sql_project.order_items` oi
on oi.order_id = o.order_id
group by 1
order by 2 desc
limit 5
```

Result:



Row	customer_state	Top_5_highest_fvalue
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Actionable Insights:

Top 5 states with highest freight value are RR, PB,RO,AC and PI

Recommendations:

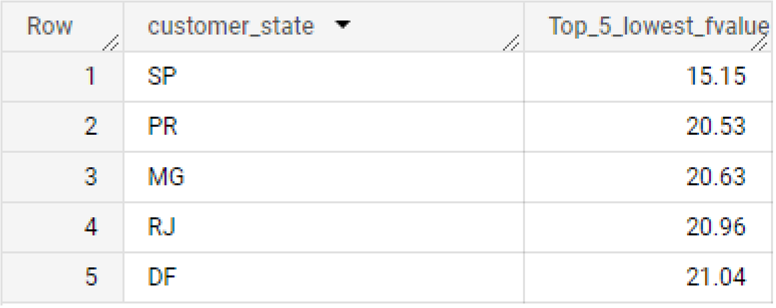
We can find out why the average freight value is high for the state and find ways to reduce it which will increase the profit.

Lowest

Query:

```
select c.customer_state,
        round(avg(oi.freight_value),2) as Top_5_lowest_fvalue,

from `sql_project.orders` o
inner join `sql_project.customers` c
on o.customer_id = c.customer_id
inner join `sql_project.order_items` oi
on oi.order_id = o.order_id
group by 1
order by 2
limit 5
```

Result:



Row	customer_state	Top_5_lowest_fvalue
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

Actionable Insights:

Top 5 states with the lowest freight values are SP, PR, MG, RJ and DF

Recommendations:

We can find ways to reduce the freight value even more if possible as it will help increase the profit.

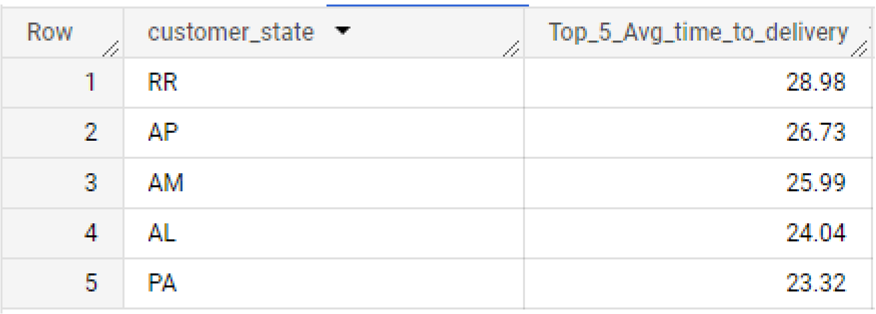
6. Top 5 states with highest/lowest average time to delivery

Highest

Query:

```
select  c.customer_state,

          round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp,
day)),2) as  Top_5_Avg_time_to_delivery

from `sql_project.customers` c
inner join `sql_project.orders` o
on c.customer_id = o.customer_id
where order_delivered_customer_date is not null
group by 1
order by 2 desc
limit 5
```

Result:



Row	customer_state	Top_5_Avg_time_to_delivery
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Actionable Insights:

Top 5 states with the highest average time to delivery are RR, AP, AM, AL and PA

Recommendations:

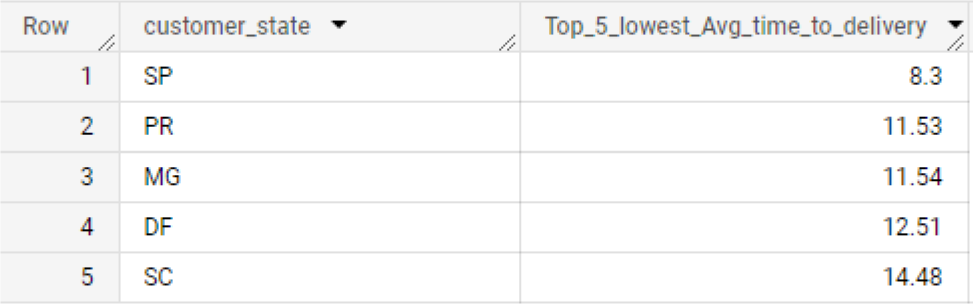
We can find ways to deliver orders faster in these states so that customers will make more purchases

Lowest

Query:

```
select  c.customer_state,

          round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, 
day)),2) as  Top_5_lowest_Avg_time_to_delivery

from `sql_project.customers` c
inner join `sql_project.orders` o
on c.customer_id = o.customer_id
where order_delivered_customer_date is not null
group by 1
order by 2
limit 5
```

Result:



Row	customer_state	Top_5_lowest_Avg_time_to_delivery
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Actionable Insights:

Top 5 lowest average time to delivery are SP, PR, MG, DF and SC

Recommendations:

We can find ways to deliver orders even fasters in these states which will attract customers to make more purchases

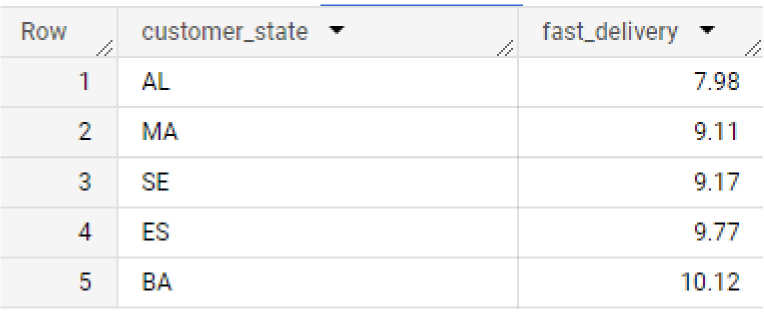
7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Top 5 states where delivery is really fast

Query:

```
select c.customer_state,
        round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date,day)),2) as fast_delivery


from `sql_project.orders` o
inner join `sql_project.customers` c
on o.customer_id = c.customer_id
inner join `sql_project.order_items` oi
on oi.order_id = o.order_id
where o.order_delivered_customer_date is not null
group by 1
order by 2
limit 5
```

Result:



Row	customer_state	fast_delivery
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

Actionable Insights:

Top 5 states where delivery is fast compared to the estimated delivery are Al, MA, SE, ES and BA

Recommendations:

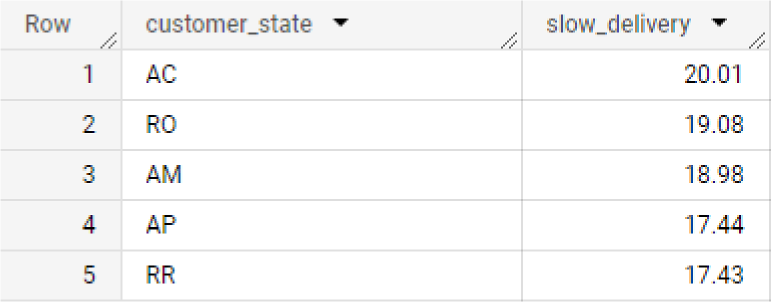
Delivery is faster in these states so we can leverage this faster delivery in ads to attract more customers

Top 5 states where delivery is not so fast

Query:

```
select c.customer_state,
        round(avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date,day)),2) as slow_delivery


from `sql_project.orders` o
inner join `sql_project.customers` c
on o.customer_id = c.customer_id
inner join `sql_project.order_items` oi
on oi.order_id = o.order_id
where o.order_delivered_customer_date is not null
group by 1
order by 2 desc
limit 5
```

Result:



Row	customer_state	slow_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

Actionable Insights:

Top 5 states where delivery is slow compared to estimated delivery are AC, RO, AM, AP and RR

Recommendations:

Delivery is slow in these states, we can find ways to deliver faster in these states which will help retain customers and also bring in new customers

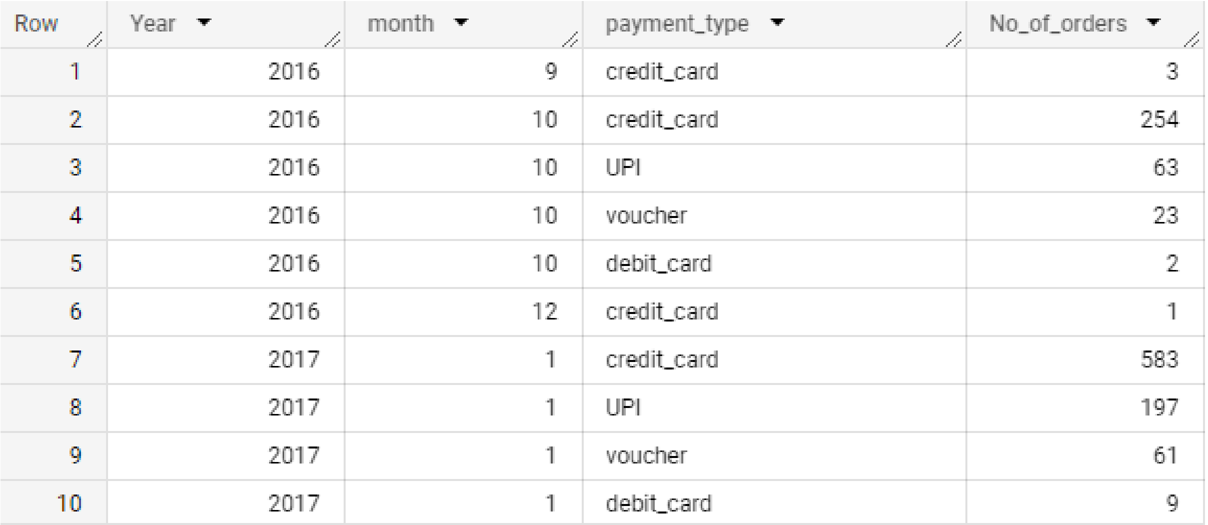
6. Payment type analysis:

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

Query:

```
select extract(year from o.order_purchase_timestamp) as Year,
        extract(month from o.order_purchase_timestamp) as month,
        p.payment_type,
        count(o.order_id) as No_of_orders

from `sql_project.orders` o
inner join `sql_project.payments` p
on o.order_id = p.order_id
group by 1,2,3
order by 1,2
```

Result:



Row	Year	month	payment_type	No_of_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9

Actionable Insights:

Over every month the number of orders placed with each payment type such as credit card, debit card, voucher and UPI.

Recommendations:

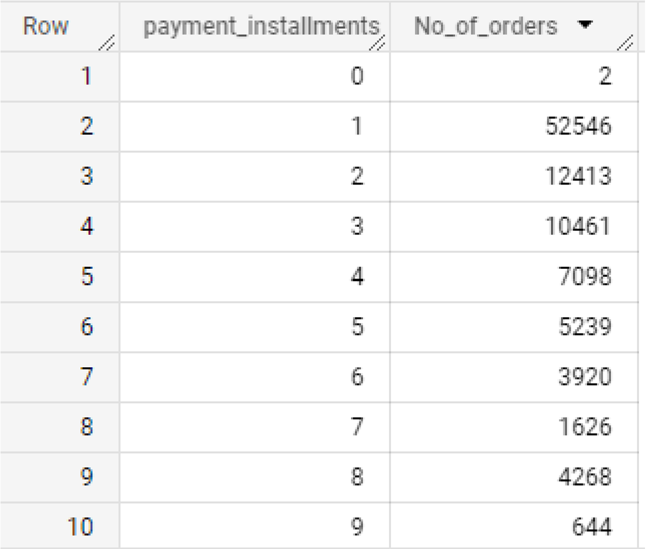
We can leverage payment information to set up subscribing and membership programs which can provide recurring revenue and improve customer retention, resulting in increased profitability over time. We can also run targeted marketing campaigns for selected payment types to increase. This can improve customer engagement, increase repeat purchases, and drive higher sales volumes.

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

Query:

```
select payment_installments, count(order_id) as No_of_orders

from `sql_project.payments`

group by 1
```

Result:



Row	payment_installments	No_of_orders ▼
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

Actionable Insights:

Total number of orders for each payment instalment. Lot of orders placed with one time payment

Recommendations:

We can promote instalment payment options to customers. They will have the flexibility to spread their payments over multiple instalments, thereby reducing the immediate financial burden. Display instalment pricing prominently on product pages and during the checkout process to encourage customers to choose this payment method.