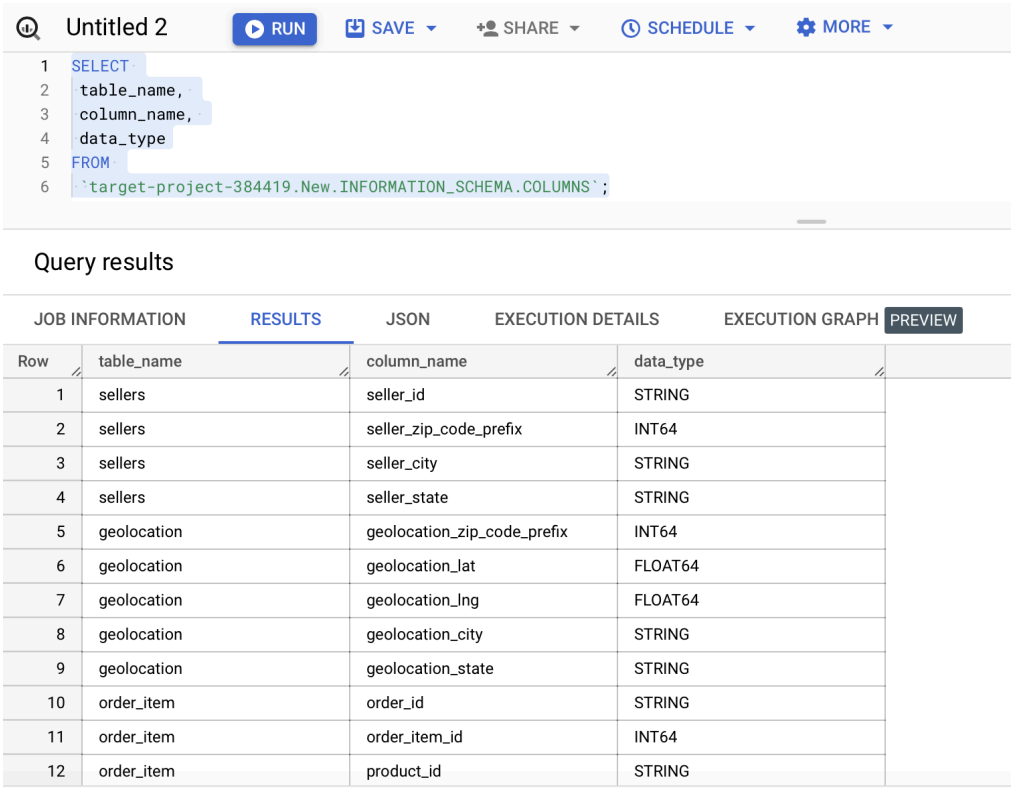
1. **Importing the dataset and conducting the usual exploratory analysis steps like checking the structure & characteristics of the dataset**

   Data type of columns in a table:

   Query:

```
SELECT
 table_name,
 column_name,
 data_type
FROM
 `target-project-384419.New.INFORMATION_SCHEMA.COLUMNS`;
```

   Result:



Untitled 2 RUN SAVE SHARE SCHEDULE MORE

```
SELECT
 table_name,
 column_name,
 data_type
FROM
 `target-project-384419.New.INFORMATION_SCHEMA.COLUMNS`;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

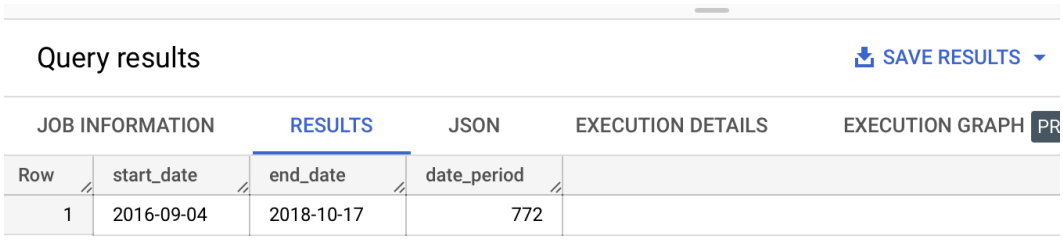
Row	table_name	column_name	data_type
1	sellers	seller_id	STRING
2	sellers	seller_zip_code_prefix	INT64
3	sellers	seller_city	STRING
4	sellers	seller_state	STRING
5	geolocation	geolocation_zip_code_prefix	INT64
6	geolocation	geolocation_lat	FLOAT64
7	geolocation	geolocation_lng	FLOAT64
8	geolocation	geolocation_city	STRING
9	geolocation	geolocation_state	STRING
10	order_item	order_id	STRING
11	order_item	order_item_id	INT64
12	order_item	product_id	STRING

   Time period for which the data is given:

Query:

```
select
min(DATE(order_purchase_timestamp)) as start_date,
max(DATE(order_purchase_timestamp)) as end_date,
date_diff(max(order_purchase_timestamp), min(order_purchase_timestamp), day) as date_period
from `New.orders`;
```

Result:



Cities and States of customers ordered during the given period

Query:

```
SELECT distinct customer_city, customer_state FROM `New.customers`
where customer_id in (select customer_id from `New.orders`)
order by customer_state asc, customer_city asc;
```

Or

```
SELECT distinct customer_city, customer_state FROM `New.customers`
group by customer_state, customer_city
order by customer_state asc, customer_city asc;
```

Both the above queries give the same output:

JOB INFORMATION | RESULTS | JSON | EX

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

---

1. In-depth Exploration:

2.1. 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?

- See how the sales are growing month on month, quarter, year. Also which region wise
- Based on the above and inference
- See which months in every year/season has high sales (aggregate months)

Query:

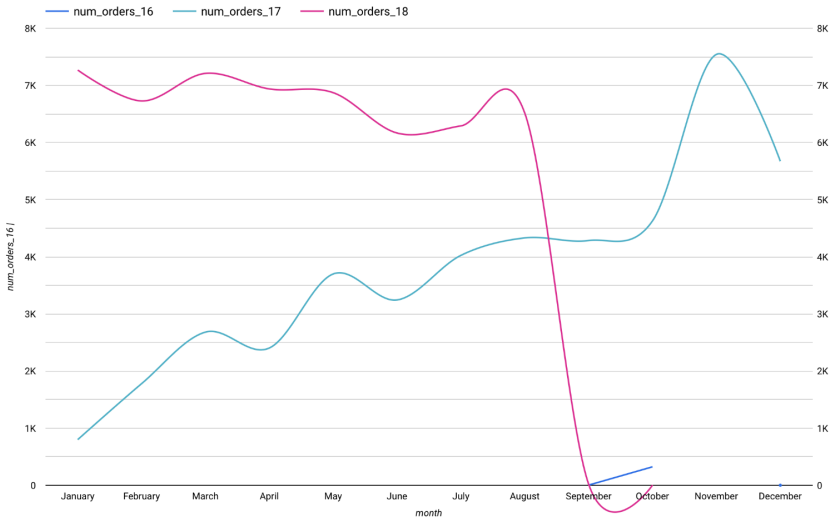
```
with n as
(select extract(month from order_purchase_timestamp) as month, count(order_id) as num_orders_16 from
`New.orders`
where extract(year from order_purchase_timestamp) = 2016
group by month
order by month asc),
n1 as
(select extract(month from order_purchase_timestamp) as month, count(order_id) as num_orders_17 from
`New.orders`
where extract(year from order_purchase_timestamp) = 2017
group by month
order by month asc),
n2 as
(select extract(month from order_purchase_timestamp) as month, count(order_id) as num_orders_18 from
`New.orders`
where extract(year from order_purchase_timestamp) = 2018
group by month
order by month asc)
select n1.month, n.num_orders_16, n1.num_orders_17, n2.num_orders_18 from n1
full join n2 on
n1.month = n2.month
left join n on
n1.month=n.month
order by n1.month;
```

Output:

Row	month	num_orders_16	num_orders_17	num_orders_18
1	1	*null*	800	7269
2	2	*null*	1780	6728
3	3	*null*	2682	7211
4	4	*null*	2404	6939
5	5	*null*	3700	6873
6	6	*null*	3245	6167
7	7	*null*	4026	6292
8	8	*null*	4331	6512
9	9	4	4285	16
10	10	324	4631	4
11	11	*null*	7544	*null*
12	12	1	5673	*null*

Seasonality with peaks at specific months

Seasonality with peaks at specific months



Insights: Due to lack of data in the year 2016 and 2018 I have considered the month on month count of orders for the year 2017.

We can infer that there is

- steady growth in the total number of orders from Jan 2017 to Jul 2017
- The growth is nearly stagnated between July to Oct 2017
- In the month of November the number of orders increased drastically during thanksgiving and before the holiday season and decreased again in December.
- But in the year 2018, we can see very minimal growth in the number of orders between Jan and August.

Avg value of order every month:

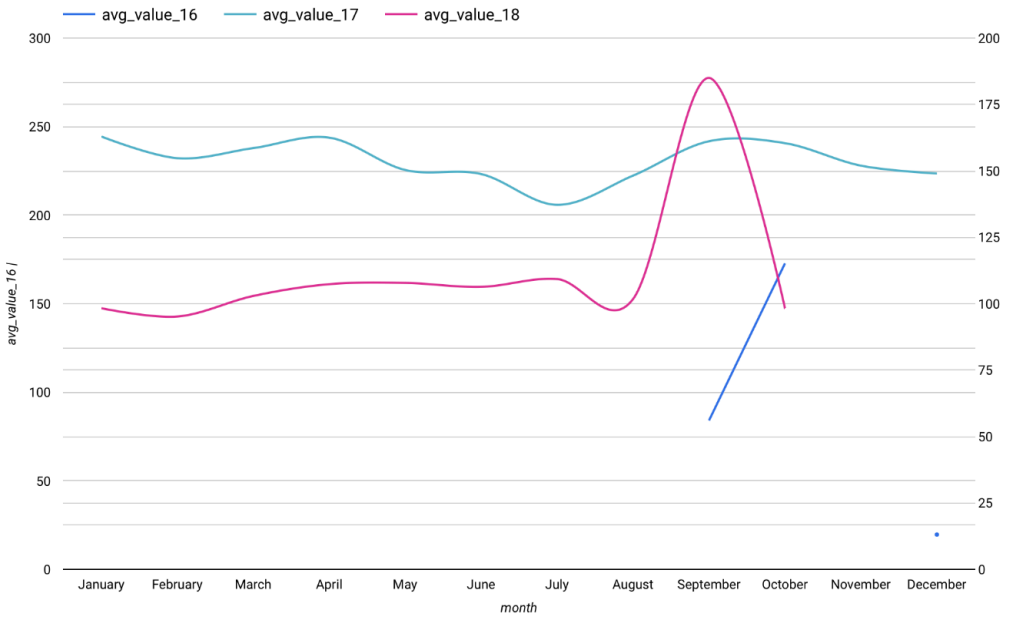
Query:

```
with n as
(select extract(month from o.order_purchase_timestamp) as month, avg(p.payment_value) as avg_value_16 from
`New.orders` o
join `New.payments`p on
p.order_id = o.order_id
where extract(year from o.order_purchase_timestamp) = 2016
group by month
order by month asc),
n1 as
(select extract(month from o.order_purchase_timestamp) as month, avg(p.payment_value) as avg_value_17 from
`New.orders` o
join `New.payments`p on
p.order_id = o.order_id
where extract(year from o.order_purchase_timestamp) = 2017
```

```
group by month
order by month asc),
n2 as
(select extract(month from o.order_purchase_timestamp) as month, avg(p.payment_value) as avg_value_18 from
`New.orders` o
join `New.payments`p on
p.order_id = o.order_id
where extract(year from o.order_purchase_timestamp) = 2018
group by month
order by month asc)
select n1.month, n.avg_value_16, n1.avg_value_17, n2.avg_value_18 from n1
full join n2 on
n1.month = n2.month
left join n on
n1.month=n.month
order by n1.month;
```

Output:

Row	month	avg_value_16	avg_value_17	avg_value_18
1	1	*null*	162.927105...	147.428821...
2	2	*null*	154.776251...	142.759398...
3	3	*null*	158.570179...	154.373285...
4	4	*null*	162.500206...	161.018931...
5	5	*null*	150.334386...	161.735409...
6	6	*null*	148.799877...	159.507789...
7	7	*null*	137.220968...	163.906677...
8	8	*null*	148.218971...	152.646360...
9	9	84.08	161.152003...	277.47125
10	10	172.779181...	160.427547...	147.417500...
11	11	*null*	151.962711...	*null*



Insights: From the above output and line chart we can see that:

- Average order value per month changes slightly throughout the year 2017
- Average order value per month also changes slightly throughout the year 2018 except in September. This is due to the drop in number of orders in the month of Sep.

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

Explanation:

Dawn: 12am to 6am

Morning 6am to 12 noon

Afternoon 12 noon to 6pm

Night 6pm to 12 midnight

Query:

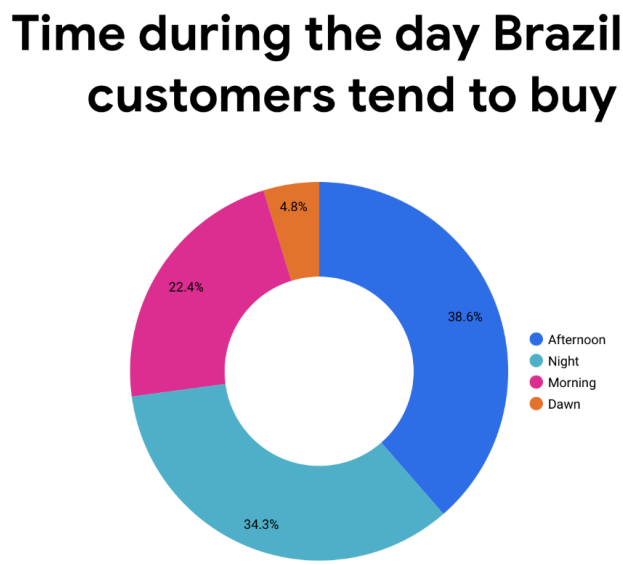
```
with n as
(select order_id, customer_id, extract(hour from order_purchase_timestamp) as hour, extract(minute from
order_purchase_timestamp) as minute, extract(second from order_purchase_timestamp) as second from
`New.orders`)
select case
when n.hour between 00 and 05 then 'Dawn'
when n.hour between 06 and 11 then 'Morning'
```

```
when n.hour between 12 and 17 then 'Afternoon'
when n.hour between 18 and 23 then 'Night'
end as time_during_day,
count(n.customer_id) as count_of_orders,
dense_rank() over(order by count(customer_id) desc) as cust_pref
from n
group by time_during_day
order by cust_pref asc;
```

Output:

Row	time_during_day	count_of_orders	cust_pref
1	Afternoon	38361	1
2	Night	34100	2
3	Morning	22240	3
4	Dawn	4740	4

Chart:



Insights:

From the above graph we can notice that the majority of Brazilian customers tend to place an order during afternoon and night time of the day. It constitutes 72.9% of the total.

---

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

Query:

```
with n as
(select c.customer_state, extract(month from o.order_purchase_timestamp) as month,
count(o.order_id) as order_count from `New.orders` o
join `New.customers` c on
c.customer_id = o.customer_id
where extract(year from o.order_purchase_timestamp) = 2017
group by c.customer_state, month
order by c.customer_state, month asc),
n1 as
(select customer_state, month, order_count,
lag(order_count, 1) over(partition by customer_state order by customer_state, month
asc) as prev_month_ord_count
from n
order by customer_state, month asc)
select customer_state, month, order_count, prev_month_ord_count,
round((order_count-prev_month_ord_count)*100/prev_month_ord_count, 2) as growth_rate
from n1;
```

Output:

Row	customer_state	month	order_count	prev_month_ord_count	growth_rate
1	AC	1	2	*null*	*null*
2	AC	2	3	2	50.0
3	AC	3	2	3	-33.33
4	AC	4	5	2	150.0
5	AC	5	8	5	60.0
6	AC	6	4	8	-50.0
7	AC	7	5	4	25.0
8	AC	8	4	5	-20.0
9	AC	9	5	4	25.0
10	AC	10	6	5	20.0
11	AC	11	5	6	-16.67
12	AC	12	5	5	0.0
13	AL	1	2	*null*	*null*

Chart:

Month on month orders by states

Top 10 - customer_state / order_count / growth_rate										
	SP		RJ		MG		RS		PR	
month	order_count	growth_rate	order_count	growth_rate	order_count	growth_rate	order_count	growth_rate	order_count	growth_rate
January	299	-	97	-	108	-	54	-	65	-
February	654	118.73	254	161.86	259	139.81	105	94.44	118	81.54
March	1,010	54.43	395	55.51	358	38.22	151	43.81	127	7.63
April	908	-10.1	338	-14.43	275	-23.18	139	-7.95	114	-10.24
May	1,425	56.94	488	44.38	428	55.64	208	49.64	213	86.84
June	1,331	-6.6	412	-15.57	363	-15.19	221	6.25	170	-20.19
July	1,604	20.51	571	38.59	453	24.79	249	12.67	203	19.41
August	1,729	7.79	562	-1.58	469	3.53	299	20.08	223	9.85
September	1,638	-5.26	609	8.36	507	8.1	278	-7.02	183	-17.94
October	1,793	9.46	668	9.69	560	10.45	252	-9.35	206	12.57
November	3,012	67.99	1,048	56.89	943	68.39	422	67.46	378	83.5
December	2,357	-21.75	783	-25.29	691	-26.72	283	-32.94	270	-28.57

3.2. Distribution of customers across the states in Brazil

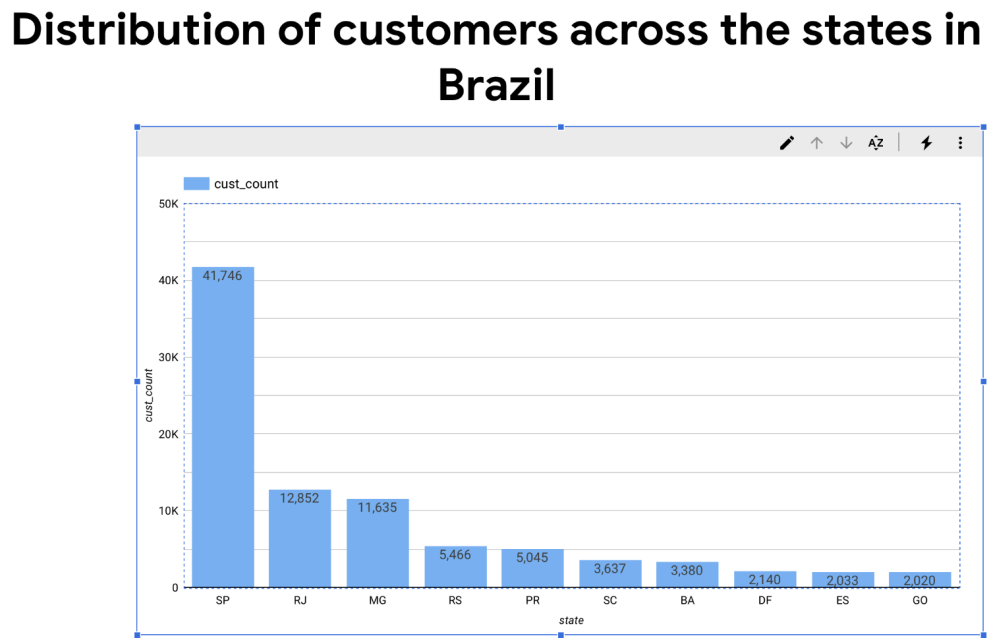
Query:

```
select distinct customer_state as state, count(distinct customer_id) as cust_count,
rank() over(order by count(distinct customer_id) desc) as rank from `New.customers`
group by customer_state
order by rank asc;
```

Output:

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

Chart:



Insights:

From the above we can infer that:

- SP state constitutes the highest number of customers

- There is a high rate of difference between the number of customers in SP and the rest of the other states.

---

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

4.1. 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

4.1.A. Query:

```
with n as
(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year,
round(sum(p.payment_value), 2) as value17
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2017
group by year
),
n1 as
(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year,
round(sum(p.payment_value), 2) as value18
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2018
group by year
)
select n.value17, n1.value18, round((n1.value18-n.value17)*100/n.value17, 2) as percent_cost_change
from n
join n1 on
n.row = n1.row;
```

Row	value17	value18	percent_cost_change
1	3669022.12	8694733.84	136.98

Insights:

From the above we can state that there is 136.98% change in the cost of orders from 2017 to 2018.

OR

4.1.B

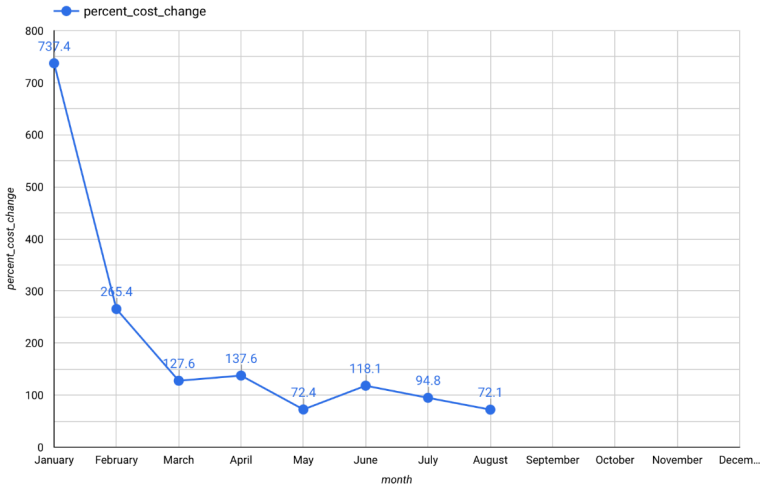
```
with n as

(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year, extract(month
from o.order_purchase_timestamp) as month, round(sum(p.payment_value), 2) as value17
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2017
group by year, month
),
n1 as
(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year, extract(month
from o.order_purchase_timestamp) as month, round(sum(p.payment_value), 2) as value18
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2018
group by year, month
)
select n.month, n.value17, n1.value18, round((n1.value18-n.value17)*100/n.value17, 2) as
percent_cost_change
from n
join n1 on
n.row = n1.row
order by n.month asc;
```

Output:

Row	month	value17	value18	percent_cost_ch
1	1	138488.04	1022425.32	638.28
2	2	291908.01	1160785.48	297.65
3	3	449863.6	1023880.5	127.6
4	4	417788.03	992463.34	137.55
5	5	592918.82	1115004.18	88.05
6	6	511276.38	1153982.15	125.71
7	7	592382.92	1159652.12	95.76
8	8	674396.32	1066540.75	58.15

% cost change in cost of orders from 2017 to 2018



Insights:

- There is a high percent of change in the cost of orders in Jan.
- The change in the month on month percentage of cost of orders is positive throughout the year from Jan to August.
- But the growth decreased drastically from Jan to Feb. went downhill hence there.

Using avg:

```
with n as
(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year, extract(month
from o.order_purchase_timestamp) as month, round(avg(p.payment_value), 2) as value17
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2017
group by year, month
),
n1 as
(
 select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year, extract(month
from o.order_purchase_timestamp) as month, round(avg(p.payment_value), 2) as value18
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2018
group by year, month
```

```
)
select n.month, n.value17, n1.value18, round((n1.value18-n.value17)*100/n.value17, 2) as
percent_avgcost_change
from n
join n1 on
n.row = n1.row
order by n.month asc;
```

Output:

Row	month	value17	value18	percent_avgcost
1	1	162.93	163.91	0.6
2	2	154.78	142.76	-7.77
3	3	158.57	159.51	0.59
4	4	162.5	161.74	-0.47
5	5	150.33	147.43	-1.93
6	6	148.8	161.02	8.21
7	7	137.22	154.37	12.5
8	8	148.22	152.65	2.99

OR

4.1.D

```
with n as
(
select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year,
round(avg(p.payment_value), 2) as value17
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2017
group by year
),
n1 as
(
select row_number() over() as row, extract(year from o.order_purchase_timestamp) as year,
round(avg(p.payment_value), 2) as value18
from `New.orders` o
join `New.payments` p on
o.order_id = p.order_id
where (extract(month from order_purchase_timestamp) between 1 and 8) and extract(year from
order_purchase_timestamp) = 2018
group by year
)
select n.value17, n1.value18, round((n1.value18-n.value17)*100/n.value17, 2) as percent_cost_change
from n
join n1 on
n.row = n1.row;
```

Output:

Row	value17	value18	percent_cost_ch
1	150.43	155.28	3.22

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

Query:

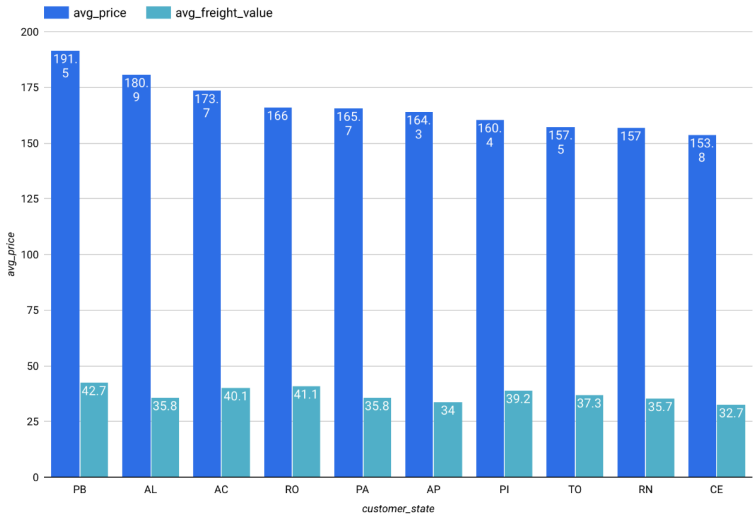
```
select c.customer_state, round(avg(o.price), 2) as avg_price, round(sum(o.price), 2) as sum_price,
round(avg(o.freight_value), 2) as avg_freight_value, round(sum(o.freight_value), 2) as sum_freight_value
from `New.order_items` o
join `New.orders` o1 on
o1.order_id = o.order_id
join `New.customers` c on
c.customer_id = o1.customer_id
group by c.customer_state
order by avg_price desc, sum_price desc, avg_freight_value desc, sum_freight_value desc;
```

Output:

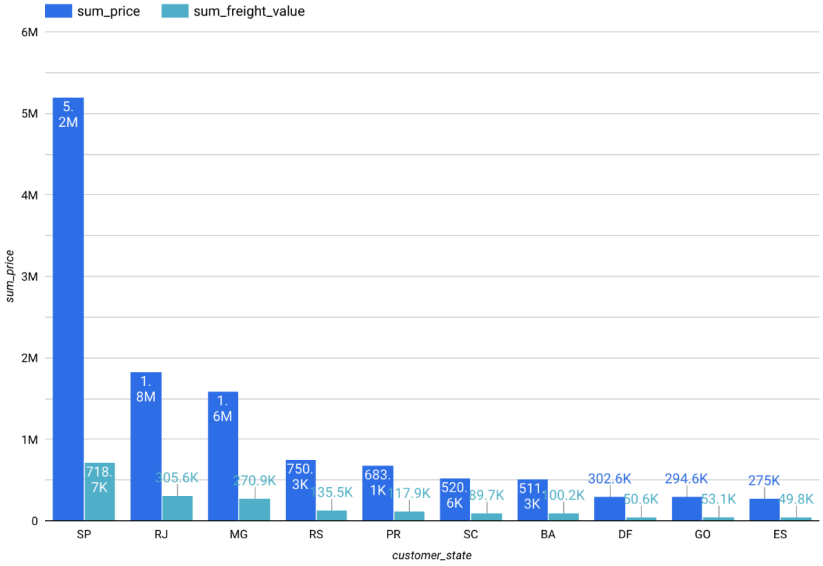
Row	customer_state	avg_price	sum_price	avg_freight_valu	sum_freight_val
1	PB	191.48	115268.08	42.72	25719.73
2	AL	180.89	80314.81	35.84	15914.59
3	AC	173.73	15982.95	40.07	3686.75
4	RO	165.97	46140.64	41.07	11417.38
5	PA	165.69	178947.81	35.83	38699.3
6	AP	164.32	13474.3	34.01	2788.5
7	PI	160.36	86914.08	39.15	21218.2
8	TO	157.53	49621.74	37.25	11732.68
9	RN	156.97	83034.98	35.65	18860.1
10	CE	153.76	227254.71	32.71	48351.59
11	SE	153.04	58920.85	36.65	14111.47
12	RR	150.57	7829.43	42.98	2235.19

Chart:

Average price and freight value for all states



Sum price and freight value for all states



---

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

    Query:

```
select customer_state, round(avg(freight_value),2) as avg_freight_value, round(avg(time_to_delivery), 2) as
avg_time_to_delivery, round(avg(diff_estimated_delivery), 2) as avg_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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state;
```

    Output:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	customer_state	avg_freight_value	avg_time_to_delivery	avg_diff_estimated_delivery
1	GO	22.56	14.95	11.37
2	SP	15.12	8.26	10.26
3	RS	21.61	14.71	13.2
4	BA	26.49	18.77	10.12
5	MG	20.63	11.51	12.4
6	MT	28.0	17.51	13.64
7	RJ	20.91	14.69	11.14
8	SC	21.51	14.52	10.66
9	SE	36.57	20.98	9.17
10	PE	32.69	17.79	12.55

4. Sort the data to get the following:
5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

5.5.a. Top 5 states with highest average freight value:

Query:

```
select customer_state, round(avg(freight_value),2) as avg_freight_value
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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_freight_value desc
limit 5;
```

Output:

Row	customer_state	avg_freight_value
1	RR	43.09
2	PB	43.09
3	RO	41.33
4	AC	40.05
5	PI	39.12

5.5.b. Top 5 states with lowest average freight value

Query:

```
select customer_state, round(avg(freight_value),2) as avg_freight_value
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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_freight_value asc
limit 5;
```

Output:

Row	customer_state	avg_freight_valu
1	SP	15.12
2	PR	20.47
3	MG	20.63
4	RJ	20.91
5	DF	21.07

5.6.a. Top 5 states with highest average time to delivery

Query:

```
select customer_state, round(avg(time_to_delivery), 2) as avg_time_to_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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_time_to_delivery desc
limit 5;
```

Output:

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

5.6.b. Top 5 states with lowest average time to delivery

Query:

```
select customer_state, round(avg(time_to_delivery), 2) as avg_time_to_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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_time_to_delivery asc
limit 5;
```

Output:

Row	customer_state	avg_time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.51
4	DF	12.5
5	SC	14.52

5.7.a. Top 5 states where delivery is really fast/ not so fast compared to estimated dat

Query:

```
select customer_state, round(avg(diff_estimated_delivery), 2) as avg_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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_diff_estimated_delivery desc
limit 5;
```

Output:

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

5.7.b. Top 5 states where delivery is not so fast compared to estimated date

Query:

```
select customer_state, round(avg(diff_estimated_delivery), 2) as avg_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,
o1.freight_value, c.customer_state
from `New.orders` o
join `New.order_items` o1 on
o.order_id = o1.order_id
join `New.customers` c on
c.customer_id = o.customer_id
WHERE o.order_status = "delivered")
group by customer_state
order by avg_diff_estimated_delivery asc
limit 5;
```

Output:

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

---

6. Payment type analysis:

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

Have considered only the year 2017 due to missing data for few months from year 2016 and 2018

Query:

```sql
with n as
(select p.payment_type, extract(month from o.order_purchase_timestamp) as month, count(distinct p.order_id) as
order_count from `New.orders`o
join `New.payments` p on
p.order_id = o.order_id
where extract(year from o.order_purchase_timestamp) = 2017
group by p.payment_type, month
order by p.payment_type asc, month asc),
n1 as
(
 select payment_type , month, order_count,
 lag(order_count, 1) over(partition by payment_type order by payment_type, month asc) as prev_month_ord_count
 from n
 order by payment_type, month asc
)
select payment_type, month, order_count, prev_month_ord_count,
round((order_count-prev_month_ord_count)*100/prev_month_ord_count, 2) as growth_rate
from n1
```

Output:

Row	payment_type	month	order_count	prev_month_ord_count	growth_rate
1	UPI	1	197	*null*	*null*
2	UPI	2	398	197	102.03
3	UPI	3	590	398	48.24
4	UPI	4	496	590	-15.93
5	UPI	5	772	496	55.65
6	UPI	6	707	772	-8.42
7	UPI	7	845	707	19.52
8	UPI	8	938	845	11.01
9	UPI	9	903	938	-3.73
10	UPI	10	993	903	9.97
11	UPI	11	1509	993	51.96
12	UPI	12	1160	1509	-23.13
13	credit_card	1	582	*null*	*null*
14	credit_card	2	1347	582	131.44

Chart:

Month over Month count of orders for different payment types

	payment_type / growth_rate / order_count							
	credit_card		debit_card		voucher		UPI	
month	growth_rate	order_count	growth_rate	order_count	growth_rate	order_count	growth_rate	order_count
January	-	582	-	9	-	33	-	197
February	131.44	1,347	44.44	13	109.09	69	102.03	398
March	49.07	2,008	138.46	31	78.26	123	48.24	590
April	-8.62	1,835	-12.9	27	-6.5	115	-15.93	496
May	54.39	2,833	11.11	30	48.7	171	55.65	772
June	-13.45	2,452	-10	27	-16.96	142	-8.42	707
July	25.29	3,072	-18.52	22	44.37	205	19.52	845
August	6.51	3,272	54.55	34	-3.41	198	11.01	938
September	0.06	3,274	26.47	43	-12.12	174	-3.73	903
October	7.21	3,510	20.93	52	19.54	208	9.97	993
November	67.15	5,867	34.62	70	28.37	267	51.96	1,509
December	-25.63	4,363	-8.57	64	-17.6	220	-23.13	1,160

Insights:

- Growth rate is highest for credit card payment in the month of February and order count is highest in the month November.
- Growth rate is highest for debit card payment in the month of March and order count is highest in the month November.
- Growth rate is highest for voucher payment in the month of February and order count is highest in the month November.
- Growth rate is highest for UPI payment in the month February and order count is highest in the month November.

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

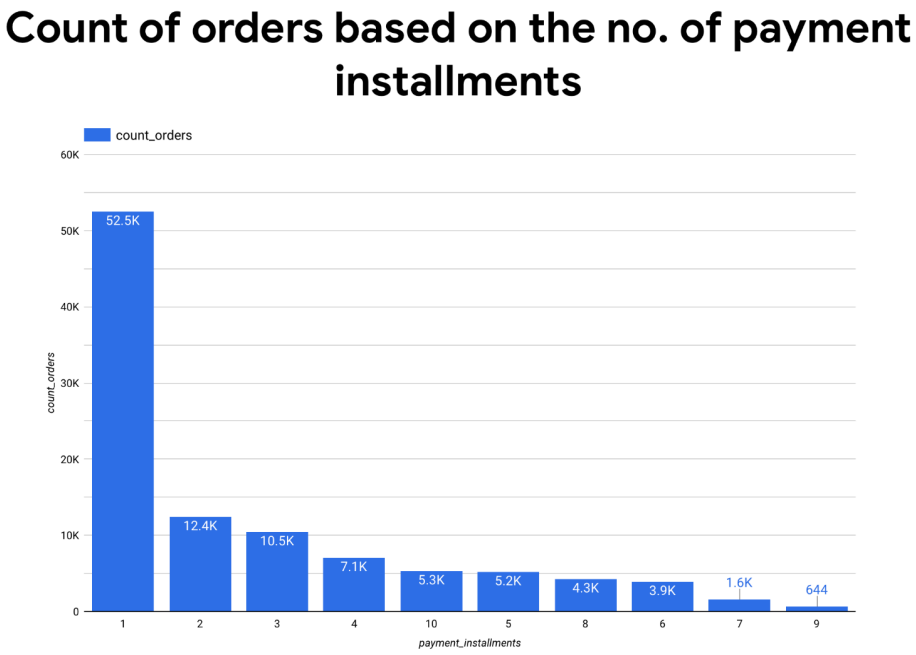
Query:

```
select payment_installments, count(order_id) as count_orders from `New.payments`
group by payment_installments
order by count_orders desc;
```

Output:

Row	payment_installments	count_orders
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

Chart:



Insights:

A large number of orders are paid through 1 installment payments. Followed by 2 and 3.

Actionable insights:

1. Seasonal Patterns: The analysis reveals a significant increase in the number of orders during the holiday season in November, followed by a decrease in December. Target can

leverage this trend by implementing targeted marketing campaigns, attractive promotions, and special offers to capitalize on the increased customer demand during these months.

2. Focus on Growth Opportunities: While there was steady growth in the number of orders from January to July 2017, growth stagnated between July and October. To reignite growth, Target can identify and focus on specific segments or product categories that have the potential for expansion. By understanding customer preferences and market trends, Target can tailor its offerings to capture new customers and increase sales.
3. Time of Day: The analysis shows that the majority of Brazilian customers tend to place orders during the afternoon and night. Target can optimize its operations during these peak hours by ensuring sufficient staffing, efficient order processing, and timely delivery. Additionally, Target can consider offering exclusive deals or promotions during these time periods to further incentivize customers to make purchases.
4. Cost of Orders: The analysis indicates a significant change (136.98%) in the cost of orders from 2017 to 2018. Target should closely monitor and analyze the factors contributing to this change. If the increase is primarily driven by higher product costs or operational expenses, Target could explore ways to optimize its supply chain, negotiate better pricing with suppliers, or identify cost-saving measures to maintain profitability.
5. Installment Payments: A large number of orders are paid through 1 installment payments, followed by 2 and 3. Target can introduce flexible payment options and financing plans to accommodate customer preferences. This could include partnerships with financial institutions to offer attractive installment-based payment solutions, promoting these options to customers during the checkout process.
6. Customer Experience: Target's commitment to providing an exceptional guest experience should remain a top priority. Continuously enhancing customer service, streamlining the online ordering process, ensuring accurate and timely deliveries, and actively addressing customer feedback and reviews will help foster customer loyalty and drive repeat business.
7. Data-Driven Decision Making: Target should continue to leverage data analysis and insights to make informed business decisions. By regularly monitoring sales trends, customer preferences, and market dynamics, Target can adapt its strategies, product offerings, and marketing campaigns to meet evolving customer needs and stay ahead of the competition.