

## TARGET SQL

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

TABLE NAME : **customers**

customers

QUERY

SHARE

COPY

SNAPSHOT

DELETE

REFRESH

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
<input type="checkbox"/>	customer_id	STRING	NULLABLE				
<input type="checkbox"/>	customer_unique_id	STRING	NULLABLE				
<input type="checkbox"/>	customer_zip_code_prefix	INTEGER	NULLABLE				
<input type="checkbox"/>	customer_city	STRING	NULLABLE				
<input type="checkbox"/>	customer_state	STRING	NULLABLE				

customers

\*Editor

sellers

order\_items

geolocation

products

orc

RUN

SAVE

SHARE

SCHEDULE

MORE

This script will process 24.75 MB when run.

```
1 SELECT * FROM `target_data.customers`;
2 SELECT COUNT(*) FROM `target_data.customers`;--99441
3 SELECT COUNT(DISTINCT customer_id) FROM `target_data.customers`;--99441
4 SELECT COUNT(DISTINCT customer_unique_id) FROM `target_data.customers`;--96096
5 SELECT COUNT(*) FROM (SELECT customer_unique_id,ROW_NUMBER() OVER(PARTITION BY customer_unique_id) AS ROWNUM
6 FROM `target_data.customers` ) AS E WHERE ROWNUM > 1;--3345 (96096+3345 = 99441)
7 SELECT * FROM (SELECT customer_unique_id,ROW_NUMBER() OVER(PARTITION BY customer_unique_id) AS ROWNUM
8 FROM `target_data.customers` ) AS E WHERE ROWNUM > 1;
9
10 SELECT DISTINCT CUSTOMER_CITY FROM `target_data.customers`;
11 SELECT COUNT(DISTINCT CUSTOMER_CITY) FROM `target_data.customers`;--4119
12 SELECT DISTINCT CUSTOMER_STATE FROM `target_data.customers`;
13 SELECT COUNT(DISTINCT CUSTOMER_STATE) FROM `target_data.customers`;--27
14
15
16
```

TABLE NAME : **sellers**

sellers

QUERY

SHARE

COPY

SNAPSHOT

DELETE

EXPORT

REFRESH

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

	Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
<input type="checkbox"/>	seller_id	STRING	NULLABLE				
<input type="checkbox"/>	seller_zip_code_prefix	INTEGER	NULLABLE				
<input type="checkbox"/>	seller_city	STRING	NULLABLE				
<input type="checkbox"/>	seller_state	STRING	NULLABLE				

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

customers

\*Editor

sellers

order\_items

geolocation

products

orc

RUN

SAVE

SHARE

SCHEDULE

MORE

Processing location: asia-south2

Press Alt+F1 for Accessibility Options.

```

1 SELECT * FROM `target_data.sellers`;
2 SELECT COUNT(*) FROM `target_data.sellers`;--3095
3 SELECT COUNT(DISTINCT SELLER_ID) FROM `target_data.sellers`;--3095
4 SELECT COUNT(*) FROM (SELECT SELLER_ID,ROW_NUMBER() OVER(PARTITION BY SELLER_ID) AS ROWNUM
5 FROM `target_data.sellers`) AS E WHERE ROWNUM > 1;-- No duplicates

```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	f0_
1	0

TABLE NAME : **order\_items**

customers

\*Editor

sellers

order\_items

geolocation

products

order\_items

QUERY

SHARE

COPY

SNAPSHOT

DELETE

REFRESH

SCHEMA

DETAILS

PREVIEW

LINEAGE

<input type="checkbox"/>	<a href="#">order_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">order_item_id</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">seller_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">shipping_limit_date</a>	TIMESTAMP	NULLABLE
<input type="checkbox"/>	<a href="#">price</a>	FLOAT	NULLABLE
<input type="checkbox"/>	<a href="#">freight_value</a>	FLOAT	NULLABLE

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

PERSONAL HISTORY

PROJECT HISTORY

REFRESH

customers

\*Editor

sellers

order\_items

geolocation

products

RUN

SAVE

SHARE

SCHEDULE

MORE

Syntax error: Expected end of input but got keyword SELECT

```

1 select * from `target_data.order_items`;
2 select count(*) from `target_data.order_items`;--112650
3 select count(distinct order_id),count(distinct order_item_id),
4 | count(distinct product_id),count(distinct seller_id) from `target_data.order_items`;--98666,(quantity)21,32951,
5 | sellers - 3095
6 SELECT * FROM (SELECT order_id,ROW_NUMBER() OVER(PARTITION BY order_id) AS ROWNUM
7 FROM `target_data.order_items`) AS E WHERE ROWNUM > 1;
8 SELECT * FROM `target_data.order_items` where order_id= '261f725152296e3e8d5041687181d836'

```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	f0_	f1_	f2_	f3_
1	98666	21	32951	3095

TABLE NAME : **geolocations**

geolocation

QUERY

SHARE

COPY

SNAPSHOT

DELETE

REFRESH

SCHEMA

DETAILS

PREVIEW

LINEAGE

Filter

Enter property name or value

Field name

Type

Mode

Collation

Default Value

Policy Tags

Description

geolocation\_zip\_code\_prefix

INTEGER

NULLABLE

geolocation\_lat

FLOAT

NULLABLE

geolocation\_lng

FLOAT

NULLABLE

geolocation\_city

STRING

NULLABLE

geolocation\_state

STRING

NULLABLE

EDIT SCHEMA

VIEW ROW ACCESS POLICIES

customers

\*Editor

sellers

order\_items

geolocation

products

+

RUN

SAVE

SHARE

SCHEDULE

MORE

Query completed.

```
1 select * from `target_data.geolocation`;  
2 select count(distinct geolocation_city) from `target_data.geolocation`;--8011  
3 select count(distinct geolocation_state) from `target_data.geolocation`;--27  
4  
5  
6
```

Processing location: asia-south2

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row

f0\_

1

8011

TABLE NAME : **payments**

payments

QUERY

SHARE

COPY

SNAPSHOT

DELETE

REFRESH

SCHEMA

DETAILS

PREVIEW

LINEAGE

Field name

Type

Mode

Collation

Default Value

Policy Tags

Description

order\_id

STRING

NULLABLE

payment\_sequential

INTEGER

NULLABLE

payment\_type

STRING

NULLABLE

payment\_installments

INTEGER

NULLABLE

payment\_value

FLOAT

NULLABLE

EDIT SCHEMA

VIEW ROW ACCESS POLICIES



review\_comment\_message

<input type="checkbox"/>	<a href="#">review_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">order_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">review_score</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">review_comment_title</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">review_creation_date</a>	TIMESTAMP	NULLABLE
<input type="checkbox"/>	<a href="#">review_answer_timestamp</a>	TIMESTAMP	NULLABLE

```
1 select * from `target_data.order_reviews`;
2 select count(*) from `target_data.order_reviews`;--99224
3 select count(distinct order_id) from `target_data.order_reviews`;--98673
4 select count(distinct review_id) from `target_data.order_reviews`;--98410
5 SELECT * FROM (SELECT review_id,ROW_NUMBER() OVER(PARTITION BY review_id) AS ROWNUM
6 FROM `target_data.order_reviews`) AS E WHERE ROWNUM > 1;
7
8 select * from `target_data.order_reviews` where review_id = '3415c9f764e478409e8e0660ae816dd2';
```

## products

<input type="checkbox"/>	<a href="#">product_id</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">product_category</a>	STRING	NULLABLE
<input type="checkbox"/>	<a href="#">product_name_length</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_description_length</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_photos_qty</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_weight_g</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_length_cm</a>	INTEGER	NULLABLE
<input type="checkbox"/>	<a href="#">product_height_cm</a>	INTEGER	NULLABLE

Processing location: asia-south2

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	product_id	product_category	product_name	product_description	product_photos	product_weight	product_length
1	a0ab96e461d74537772b8495...	climatization	41	717	1	1050	
2	4d7585daba2f8b3ed7f874479...	fixed telephony	53	897	2	300	

2. Time period for which the data is given (2016-09-04 TO 2018-10-17)

Processing location: asia-south2

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	f0_
1	2018-10-17

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

Processing location: asia-south2

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	customer_state	customer_city
1	RN	acu
2	CE	ico
3	RS	ine

## 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?

1. Yearly number of sales was increasing

```
with cte as
(select
  o.order_id,
  FORMAT_DATE('%Y', DATE(o.order_purchase_timestamp)) as year,
  price
 from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
)

select year,COUNT(order_id) as cnt from cte
group by year
order by cnt desc;
```

The screenshot shows a SQL IDE interface with a query editor and a results table. The query editor contains the following SQL code:

```
1 with cte as
2 (select
3   o.order_id,
4   FORMAT_DATE('%Y', DATE(o.order_purchase_timestamp)) as year,
5   price
6 from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
7 )
8
9 select year,COUNT(order_id) as cnt from cte
10 group by year
11 order by cnt desc;
```

The results table, titled "Query results", shows the following data:

Row	year	cnt
1	2018	61652
2	2017	51386
3	2016	387

2. In 2017 sales started increasing, at NOV -17 it reached highest number of sales

```
with cte as
(select
  o.order_id,
  FORMAT_DATETIME("%B, %Y",o.order_purchase_timestamp) as m_year,
  price
from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
)

select m_year,round(sum(price),0) as sum_yr from cte
where m_year like '%2017%'
group by m_year
order by sum_yr desc;
```

Untitled 3

RUN

SAVE

SHARE

SCHEDULE

MORE

Query completed

```
1 with cte as
2 (select
3   o.order_id,
4   FORMAT_DATETIME("%B, %Y",o.order_purchase_timestamp) as m_year,
5   price
6 from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
7 )
8
9 select m_year,round(sum(price),0) as sum_yr from cte
10 where m_year like '%2017%'
11
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	m_year	sum_yr
1	November, 2017	1010271.0
2	December, 2017	743914.0



3. In 2018 – Initially the sales is in Peak, JAN-18, MAR-18 the sales is in peak, later sales maintained above 6k

```
with cte as
(select
  o.order_id,
  FORMAT_DATETIME("%B, %Y",o.order_purchase_timestamp) as m_year,
  price
from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
)
select m_year,round(sum(price),0) as sum_yr from cte
where m_year like '%2018%'
group by m_year
order by sum_yr desc;
```

Query completed.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	m_year	sum_yr
1	May, 2018	990018.0
3	March, 2018	983213.0
4	January, 2018	950030.0

4. Lowest sales were in before DEC-16, SEP-18, OCT-18

```
with cte as
(select
  o.order_id,
  FORMAT_DATETIME("%B, %Y",o.order_purchase_timestamp) as month,
  price
from `target_data.orders` o left join `target_data.order_items` oi on o.order_id= oi.order_id
)
select month,round(sum(price),0) as price from cte
--where month in ('September, 2018','October, 2018','December, 2016')
group by month
order by price;
```

⌕ \*Untitled 2 × ⌕ \*Untitled 4 × ⌕ \*Untitled 5 × ⌕ \*Untitled 3 ×

⌕ Untitled 3

RUN

SAVE

SHARE

SCHEDULE

MORE

```
1 with cte as
2 (select
3   o.order_id,
4   FORMAT_DATETIME("%B, %Y", o.order_purchase_timestamp) as month,
5   price
6   from `target_data.orders` o left join `target_data.order_items` oi on o.order_id = oi.order_id
7 )
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	month	price
1	October, 2018	null
2	December, 2016	11.0
3	September, 2018	145.0
4	September, 2016	267.0
5	October, 2016	49508.0

⌕ \*Untitled 2 × ⌕ \*Untitled 4 × ⌕ \*Untitled 5 × ⌕ \*Untitled 3 ×

⌕ Untitled 3

RUN

SAVE

SHARE

SCHEDULE

MORE

Query completed

```
1 with cte as
2 (select
3   o.order_id,
4   FORMAT_DATETIME("%B, %Y", o.order_purchase_timestamp) as month,
5   price
6   from `target_data.orders` o left join `target_data.order_items` oi on o.order_id = oi.order_id
7 )
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	month	price
1	April, 2018	996648.0
2	May, 2018	996518.0
3	March, 2018	983213.0
4	January, 2018	950030.0
5	July, 2018	895507.0

⌕ \*Unsaved query 2 × payments × order\_items × orders ×

⌕ Unsaved query 2

RUN

SAVE

SHARE

SCHEDULE

MORE

This script will process 41.97 MB when run.

```
20
21 SELECT count(order_id),
22        FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) AS year_name
23 FROM `target_data.orders`
24 group by year_name order by year_name;
```

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

This BigQuery table is being used for Change Data Capture(CDC), and this preview was last updated at Mon Apr 03 2023 18:54:12 GMT+0530 (India Standard Time). To pull real-time data, please query the table.

Row	f0_	year_name
1	329	2016
2	45101	2017
3	54011	2018

```
18 select * from `target_data.payments`;
19
20
21 SELECT order_purchase_timestamp, FORMAT_DATE('%B', DATE(order_purchase_timestamp)) AS month_name,
22 FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) AS year_name
23 FROM `target_data.orders`;
24
25 SELECT order_purchase_timestamp,concat(FORMAT_DATE('%B', DATE(order_purchase_timestamp)), ' ',FORMAT_DATE('%Y', DATE
26 (order_purchase_timestamp))) as month_year
27 FROM `target_data.orders`;
28
29 select count(order_id),month_year from
30 (
31 SELECT order_id,concat(FORMAT_DATE('%B', DATE(order_purchase_timestamp)), ' ',FORMAT_DATE('%Y', DATE
32 (order_purchase_timestamp))) as month_year
33 FROM `target_data.orders`
34 ) as a
35 group by month_year
36 order by substr(month_year,-2,4);
```

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	f0_	month_year
1	324	October 2016
2	4	September 2016
3	1	December 2016
4	7544	November 2017
5	5673	December 2017
6	2404	April 2017

1. Most orders placed in 9 to 22 (Morning and Afternoon)
2. Maximum orders received 4 o'clock (Afternoon)
3. Minimum numbers of orders received between 1 and 6 o'clock (Morning)

```
19
20
21 SELECT COUNT(ORDER_ID),
22        EXTRACT(HOUR FROM TIMESTAMPTO_TIMESTAMP(order_purchase_timestamp)) AS hour_ord
23 FROM `target_data.orders`
24 --WHERE EXTRACT(YEAR FROM TIMESTAMPTO_TIMESTAMP(order_purchase_timestamp))= 2018
25 GROUP BY hour_ord
26 ORDER BY hour_ord;
--
```

Query results

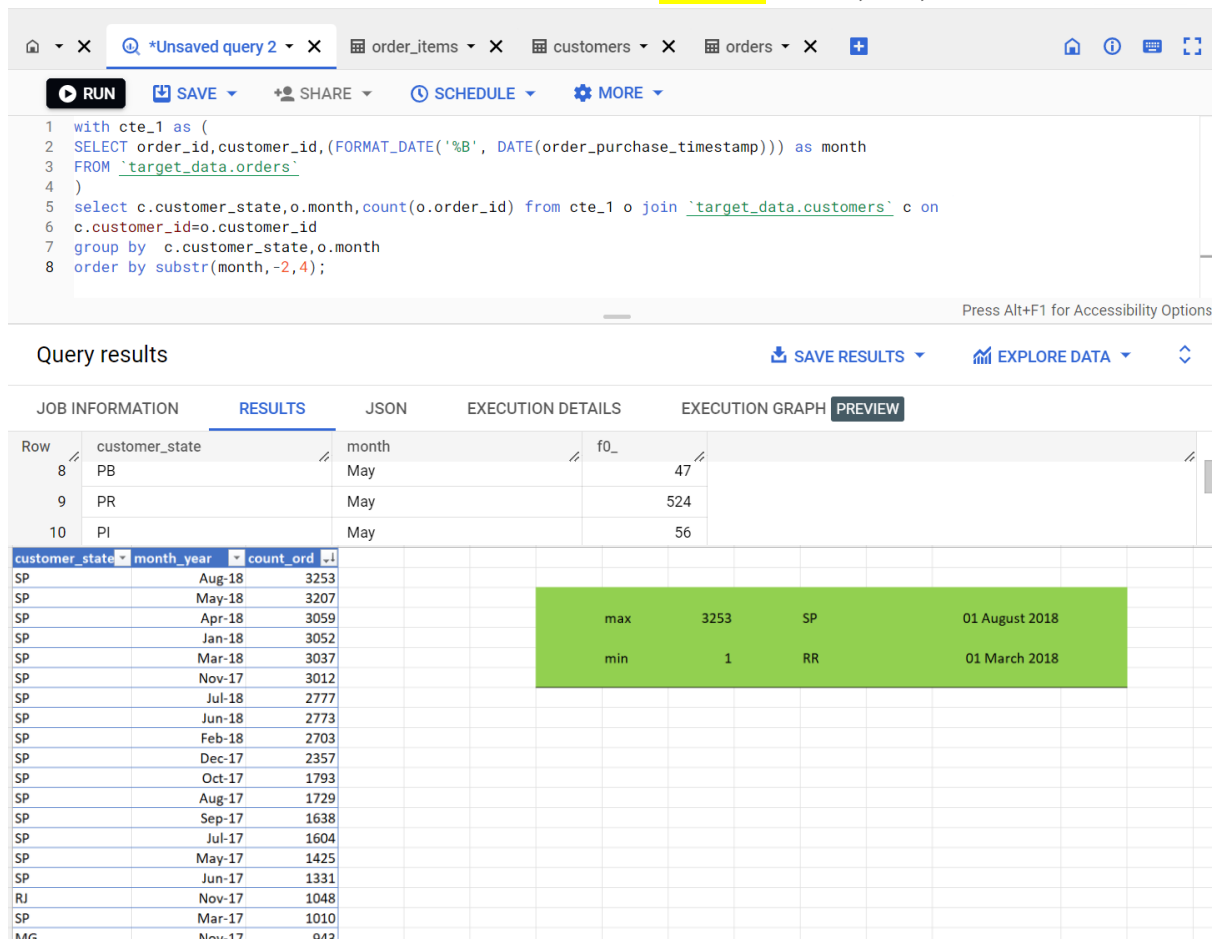
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	f0_	hour_ord				
1	2394	0				
2	1170	1				
3	510	2				

order_Count	hour
2394	0
1170	1
510	2
272	3
206	4
188	5
502	6
1231	7
2967	8
4785	9
6177	10
6578	11
5995	12
6518	13
6569	14
6454	15
6675	16
6150	17
5769	18
5982	19
6193	20
6217	21
5816	22
4123	23

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

#### 1. Get month on month orders by states

- Maximum orders placed in the state of **SP -São Paulo** in the month of **AUG-18** and also many orders were placed in the month of **MAR-2018 to MAY -2018** in the state of São Paulo
- Minimum ordered state are RR - **Roraima** in SEP,OCT,DEC months



#### 2. Distribution of customers across the states in Brazil

- In Brazil maximum customers are in state **SP -São Paulo**
- Minimum customers are in state of RR - **Roraima**



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

```

with cte as
(select
  o.order_id,FORMAT_DATE('%B', DATE(order_purchase_timestamp)) as month,
  FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) as year,
  price
 from `target_data.orders` o join `target_data.order_items` oi on o.order_id= oi.order_id
 where FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) = '2017'),
cte2 as
(select
  o.order_id,FORMAT_DATE('%B', DATE(order_purchase_timestamp)) as month,
  FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) as year,
  price
 from `target_data.orders` o join `target_data.order_items` oi on o.order_id= oi.order_id
 where FORMAT_DATE('%Y', DATE(order_purchase_timestamp)) = '2018')
select a.month,a.c1,b.c2,(b.c2-a.c1) as diff,round(((b.c2-a.c1)/a.c1)*100,2)as percent_incr from
(select cte.month, round(sum(cte.price),0) as c1 from cte group by cte.month ) as a
join
(select cte2.month, round(sum(cte2.price),0) as c2 from cte2 group by cte2.month) as b
on a.month = b.month
and a.month not in ('September','October','November','December')
order by percent_incr desc;

```

Query completed.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	month	year_2017	year_2018	diff	percent_incr	
1	January	120313.0	950030.0	829717.0	689.63	
2	February	247303.0	844179.0	596876.0	241.35	
3	April	359927.0	996648.0	636721.0	176.9	
4	March	374344.0	983213.0	608869.0	162.65	
5	June	123020.0	865124.0	742104.0	602.48	

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

1. Maximum Mean price value of customer state is PB – Paraíba
2. Minimum mean price of customer state is SP- São Paulo
3. Maximum mean freight value of customer state is – RR – Roraima
4. Minimum mean freight value of customer state is – SP- São Paulo
5. Maximum total price of customer state is - is SP- São Paulo
6. Minimum total price of customer state is - RR – Roraima

```
6
7 select c.customer_state,sum(oi.price),sum(oi.freight_value),avg(oi.price),avg(oi.freight_value)
8 from `target_data.customers` c join `target_data.orders` o on c.customer_id=o.customer_id
9 join `target_data.order_items` oi on o.order_id=oi.order_id
10 group by c.customer_state
11 order by avg(oi.price)desc;
```

Press Alt+F1 for Accessibility Option

### Query results

[SAVE RESULTS](#) ▾

[EXPLORE DATA](#) ▾



JOB INFORMATION

**RESULTS**

JSON

EXECUTION DETAILS

EXECUTION GRAPH

**PREVIEW**

Row	customer_state	f0_	f1_	f2_	f3_
1	PB	115268.080...	25719.7299...	191.475215...	42.7238039...
2	AL	80314.8100...	15914.5899...	180.889211...	35.8436711...
3	AC	15982.9499...	3686.74999...	173.727717...	40.0733695...
4	RO	46140.6400...	11417.38	165.973525...	41.0697122...



## 5. Analysis on sales, freight and delivery time

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

Query editor interface showing a SQL query to calculate delivery metrics:

```
1 select order_id,
2 order_purchase_timestamp,
3 order_delivered_customer_date,
4 order_estimated_delivery_date,
5 date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as cus_delivery,
6 date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as estimate_delivery
7 from `target_data.orders`
8 where order_delivered_customer_date is not null;
```

Query results table:

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	cus_delivery	estimate_delivery
4	eb9547ae1a...	2016-10-08 20:17:50 UTC	2016-10-19 18:47:43 UTC	2016-11-30 00:00:00 UTC	10	52
5	dc42f66542...	2016-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	52
6	334a486ec9e...	2017-03-17 15:56:47 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	20	61

### 2. Find time\_to\_delivery & diff\_estimated\_delivery. Formula for the same given below:

Query editor interface showing a SQL query to calculate delivery metrics:

```
1 select order_id,
2 order_purchase_timestamp,
3 order_delivered_customer_date,
4 order_estimated_delivery_date,
5 date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery,
6 date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as diff_estimated_delivery
7 from `target_data.orders`
8 where order_delivered_customer_date is not null;
```

Query results table:

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_delivery
41	d28eec0b54...	2018-05-04 15:21:06 UTC	2018-06-18 12:19:28 UTC	2018-06-06 00:00:00 UTC	44	-12
42	37715f55306...	2018-05-15 21:21:58 UTC	2018-06-20 21:32:54 UTC	2018-06-06 00:00:00 UTC	36	-14
43	624b7f6222...	2017-10-24 15:15:50 UTC	2017-12-06 10:41:24 UTC	2017-11-10 00:00:00 UTC	43	-26

### 3. Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery

Untitled 3 ▶ RUN ▶ SAVE ▶ SHARE ▶ SCHEDULE ▶ MORE ✓ This query will process 16.82 MB ...

```

12 order by c.customer_id)
13
14 select
15     customer_state,
16     round(avg(freight_value),2) mean_freight_value,
17     round(avg(diff_estimated_delivery),2) as mean_diff_estimated_delivery,
18     round(avg(time_to_delivery),2) as mean_time_to_delivery
19 from cte
20 group by customer_state;

```

Press Alt+F1 for Accessibility Options

### Query results

▶ SAVE RESULTS

▶ EXPLORE DATA



JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	mean_freight_value	mean_diff_estimated	mean_time_to_delive		
21	MA	38.49	21.20375000000...	9.109999999999...		
22	DF	21.07	12.50148619957...	11.27473460721...		
23	PI	39.12	18.93116634799	10.68260038240		

4. Sort the data to get the following:

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

```

with cte as(
select
    c.customer_id,
    o.order_id,
    c.customer_state,
    oi.freight_value,
    date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day) as diff_estimated_delivery,
    date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day) as time_to_delivery
from `target-382407.target_data.customers` c join `target-382407.target_data.orders` o on c.customer_id= o.customer_id
join `target-382407.target_data.order_items` oi on o.order_id= oi.order_id
where o.order_delivered_customer_date is not null
order by c.customer_id)

select
    customer_state,
    round(avg(freight_value),2) mean_freight_value,
    round(avg(diff_estimated_delivery),2) as mean_diff_estimated_delivery,
    round(avg(time_to_delivery),2) as mean_time_to_delivery
from cte
group by customer_state;

```

order\_items ▶ \*Unsaved query 3 ▶

▶ RUN ▶ SAVE ▶ SHARE ▶ SCHEDULE ▶ MORE

```

1 select c.customer_state,avg(oi.freight_value) as avg_2
2 from `target_data.customers`c
3 join `target_data.orders`o on o.customer_id=c.customer_id
4 join `target_data.order_items`oi on oi.order_id= o.order_id
5 group by c.customer_state
6 order by avg_2
7 limit 5;

```

Press Alt+F1 for Accessibility Options

### Query results

▶ SAVE RESULTS

▶ EXPLORE DATA



JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_2				
1	SP	15.1472753...				
2	PR	20.5316515...				
3	MG	20.6301668...				
4	RJ	20.9609239...				
5	DF	21.0413549...				

order\_items X \*Unsaved query 3 X

RUN SAVE SHARE SCHEDULE MORE

```

1 select c.customer_state,avg(oi.freight_value) as avg_2
2 from `target_data.customers` c
3 join `target_data.orders` o on o.customer_id=c.customer_id
4 join `target_data.order_items` oi on oi.order_id= o.order_id
5 group by c.customer_state
6 order by avg_2 desc
7 limit 5;

```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	avg_2				
1	RR	42.9844230...				
2	PB	42.7238039...				
3	RO	41.0697122...				
4	AC	40.0733695...				
5	PI	39.1479704...				

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

```

with cte as(
select
  c.customer_id,
  o.order_id,
  c.customer_state,
  oi.freight_value,
  date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day) as diff_estimated_delivery,
  date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day) as time_to_delivery
from `target-382407.target_data.customers` c join `target-382407.target_data.orders` o on c.customer_id= o.customer_id
join `target-382407.target_data.order_items` oi on o.order_id= oi.order_id
where o.order_delivered_customer_date is not null
order by c.customer_id)

select
  customer_state,
  round(avg(time_to_delivery),2) as time_to_delivery
from
  cte
group by customer_state
order by time_to_delivery;

```

\*Untitled 2 X \*Untitled 3 X \*Untitled 4 X

Untitled 4 RUN SAVE SHARE SCHEDULE MORE

```

1 with cte as(
2 select
3   c.customer_id,
4   o.order_id,
5   c.customer_state,

```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state	time_to_delivery				
1	AL	7.98				
2	MA	9.11				
3	SE	9.17				
4	ES	9.77				
5	BA	10.12				

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

```

with cte as(
select
  c.customer_id,
  o.order_id,
  c.customer_state,
  oi.freight_value,
  date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day) as diff_estimated_delivery,
  date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,day) as time_to_delivery
from `target-382407.target_data.customers` c join `target-382407.target_data.orders` o on c.customer_id= o.customer_id
join `target-382407.target_data.order_items` oi on o.order_id= oi.order_id
where o.order_delivered_customer_date is not null
order by c.customer_id)

select * from
(select
  customer_state,
  round(avg(time_to_delivery),2) as time_to_delivery,
  round(avg(diff_estimated_delivery),2) as estimated_delivery
from
  cte
group by customer_state
order by time_to_delivery)as a
where time_to_delivery > estimated_delivery;

```

Untitled 4 ▶ RUN 📄 SAVE ▼ 👤 SHARE ▼ 🕒 SCHEDULE ▼ ⚙️ MORE ▼

```

17 round(avg(time_to_delivery),2) as time_to_delivery,
18 round(avg(diff_estimated_delivery),2) as estimated_delivery
19 from
20 | cte
21 group by customer_state
22 order by time_to_delivery)as a
23 where time_to_delivery > estimated_delivery;

```

Press Alt+F1 for Accessibility Options.

## Query results

📄 SAVE RESULTS ▼ 📊 EXPLORE DATA ▼ ↕

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	customer_state ▼	time_to_delivery ▼	estimated_delivery			
1	SP	10.27	8.26			
2	MG	12.4	11.52			
3	PR	12.53	11.48			

## 6. Payment type analysis:

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

```
with cte as
(
select
o.order_id,
FORMAT_DATETIME("%B, %Y",o.order_purchase_timestamp) as month,
payment_type
from `target-382407.target_data.orders` o join `target_data.payments` p on o.order_id= p.order_id
)

select month,payment_type,count(order_id)
from cte
group by month,payment_type
order by month,payment_type;
```

Press Alt+F1 for Accessibility Options.

Query results

[SAVE RESULTS](#) [EXPLORE DATA](#)

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	EXECUTION GRAPH	PREVIEW
Row	month	payment_type	fo_			
1	April, 2017	UPI	496			
2	April, 2017	credit_card	1846			
3	April, 2017	debit_card	27			
4	April, 2017	voucher	202			
5	April, 2018	UPI	1287			
6	April, 2018	credit_card	5455			

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

🏠 × 🔍 \*Untitled 2 × 🔍 \*Untitled 4 × 🔍 \*Untitled 5 × 📄 payments × 🔍 \*Untitled 3 × +

🔍 **Untitled 3** ▶️ RUN 📄 SAVE ▾ 👤 SHARE ▾ ⌚ SCHEDULE ▾ ⚙️ MORE ▾ ✅ Query completed.

```
1 SELECT payment_installments, count(order_id)
2 FROM `target-382407.target_data.payments`
3 group by payment_installments
4 order by count(order_id) desc;
```

Press Alt+F1 for Accessibility Options.

Query results

📄 SAVE RESULTS ▾ 📊 EXPLORE DATA ▾ ⬆️

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	payment_installment	f0_
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328

```
with cte as
(
select
o.order_id,
payment_type,
payment_installments
from `target-382407.target_data.orders` o join `target_data.payments` p on o.order_id= p.order_id
)
select payment_type, payment_installments, count(order_id)
from cte
group by payment_type, payment_installments;
```

🏠 × 🔍 \*Untitled 2 × 🔍 \*Untitled 4 × 🔍 \*Untitled 5 × 📄 payments × 🔍 \*Untitled 3 × +

🔍 **Untitled 5** ▶️ RUN 📄 SAVE ▾ 👤 SHARE ▾ ⌚ SCHEDULE ▾ ⚙️ MORE ▾

```
1 with cte as
2 (
3 select
4 o.order_id,
5 payment_type,
6 payment_installments
```

Press Alt+F1 for Accessibility Options.

Query results

📄 SAVE RESULTS ▾ 📊 EXPLORE DATA ▾ ⬆️

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

EXECUTION GRAPH

PREVIEW

Row	payment_type	payment_installment	f0_
1	UPI	1	19784
2	credit_card	7	1626
3	credit_card	10	5328
4	credit_card	6	3920
5	voucher	1	5775

## 7. Actionable Insights

1. People of buying in the single instalment is more around 52546 so much of wealthiest people living in SP state, we can provide more stores so that they No of customers got increased
2. From the state RR of a smaller number of people buying items there will give more offers so that will attract more customers it leads to improve the sales over there
3. May and August having most sales, will minimize the freight value on that month let the customers will purchase more

## 8. Recommendations

1. mean fright value varies between 15 to 45
2. Finding the better place or any good storage places nearby each state it indirectly reduces the fight value
3. It will more affordable customer can purchase more items more often
4. September and October month having minimum amount of sales happen, Focus more advertisement on that month and Give more offers to customers
5. Conduct Big billion days kinds of sales and offers can improve the customers strength
6. AP and RR state was having more number of days to take delivered, To improve the time to deliver and minimize the amount of days to deliver in these state, Implementing the local goods storage better improve the time to deliver