

Project_name: my-targetsql-project

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 – Orders

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar displays the project structure under 'my-targetsql-project'. The 'Orders' table is selected in the main pane. The 'SCHEMA' tab is active, showing the following columns:

Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
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				

At the bottom of the schema view, there are buttons for 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'.

2. Time period for which the data is given

```
select
MIN(order_purchase_timestamp) AS from_date,
MAX(order_purchase_timestamp) AS till_date
from `my-targetsql-project.Case_Target_sql.Orders`
limit 10;
```

The screenshot shows the Google Cloud BigQuery interface with the results of the executed SQL query. The query is:

```
select
MIN(order_purchase_timestamp) AS from_date,
MAX(order_purchase_timestamp) AS till_date
from `my-targetsql-project.Case_Target_sql.Orders`
limit 10;
```

The results table shows one row:

from_date	till_date
2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

```
SELECT DISTINCT Cus.customer_city,Cus.customer_state
FROM `my-targetsql-project.Case_Target_sql.Orders` AS Ord
INNER JOIN `my-targetsql-project.Case_Target_sql.Customers` AS Cus
ON Ord.customer_id=Cus.customer_id
LIMIT 10;
```

The screenshot shows the Google BigQuery web interface. In the top navigation bar, it says "Query results - BigQuery - My-TargetSQL-Project". Below the navigation, there's a search bar and a "RUN" button. The main area is titled "Query results" and displays a table of data. The table has columns "customer_city" and "customer_state". The data is as follows:

customer_city	customer_state
acu	RN
ico	CE
ipe	RS
ipu	CE
ita	SC
itu	SP
jau	SP
luz	MG
poa	SP
uba	MG

2. In-depth Exploration:

- 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?

```
SELECT DISTINCT
COUNT(order_id) AS sales_count,Month,Year
FROM
(SELECT *,
extract(month from order_purchase_timestamp) AS Month,
extract(year from order_purchase_timestamp) AS Year
FROM `my-targetsql-project.Case_Target_sql.Orders` ) temp
GROUP BY Month,Year
ORDER BY Year,Month
LIMIT 10;
```

The screenshot shows the Google BigQuery web interface. The top navigation bar includes 'Google Cloud' and 'My-TargetSQL-Project'. The main area has tabs for 'Sandbox' and 'Explorer'. A sidebar on the left lists datasets like 'my-targetsql-project' and its tables, including 'Orders' which is currently selected. The central workspace contains an 'Unsaved query' editor with the SQL code provided above, and a 'Query results' section below it. The 'RESULTS' tab is active, displaying a table with 10 rows of data:

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

>>> Initially in **October 2016** there is increase in order purchases/sales which again declines for next 2 months. However, during **year 2017** there was upward trend in order purchases/sales for whole year. Also observed some sudden spike during Month of **November 2017** as its month with **Black Friday sale** and it can be considered seasonal. Also, **1st quarter of the year 2018** shows high sales which slowly declines in next quarter and sharply declines further from the end of **3rd quarter**. Overall, we can say there is growing trend on e-commerce in Brazil since 2016 till 2018 as it has grown quite a lot than initial stage.

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

```
SELECT
COUNT(order_id) AS sales_count,time_phase
FROM
(SELECT *,
CASE

WHEN TIME(order_purchase_timestamp) BETWEEN '03:00:00' AND '05:00:00' THEN 'Dawn'
WHEN TIME(order_purchase_timestamp) BETWEEN '05:00:00' AND '12:00:00' THEN 'Morning'
WHEN TIME(order_purchase_timestamp) BETWEEN '12:00:00' AND '18:00:00' THEN 'Afternoon'
ELSE 'Night'
END as time_phase
FROM `my-targetsql-project.Case_Target_sql.Orders` ) temp
GROUP BY time_phase
ORDER BY sales_count DESC;
```

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the project structure under 'my-targetsql-project'. The main area shows the query code in the 'Orders' tab, with the results table below it. The results table has four rows:

Row	sales_count	time_phase
1	38365	Afternoon
2	38170	Night
3	22428	Morning
4	478	Dawn

>>>Overall customers are active during daytime and maximum purchases are made during afternoon time following nighttime at second place. Morning time purchases are still made at commendable rate however dawn phase is the least to have activity from customers.

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

1. Get month on month orders by states

```
SELECT DISTINCT
COUNT(order_id) AS sales_count,Month,Year,temp.customer_state
FROM
(SELECT ord.order_id,cus.customer_state,
extract(month from order_purchase_timestamp) AS Month,
extract(year from order_purchase_timestamp) AS Year
FROM `my-targetsql-project.Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Customers` AS cus
ON ord.customer_id=cus.customer_id
INNER JOIN `Case_Target_sql.Geolocation` as geo
ON geo.geolocation_zip_code_prefix=cus.customer_zip_code_prefix) temp
GROUP BY Month,Year,temp.customer_state
ORDER BY Year,Month,sales_count;
```

The screenshot shows the Google Cloud BigQuery interface. The top navigation bar includes 'Google Cloud' and 'My-TargetSQL-Project'. The main area has tabs for 'Explorer', '+ ADD DATA', and 'Editor'. The 'Editor' tab is active, displaying the SQL query. The 'RUN' button is highlighted. Below the editor, the 'Query results' section shows the processed data. The data is presented in a table with columns: Row, sales_count, Month, Year, and customer_state. The results are as follows:

Row	sales_count	Month	Year	customer_state
1	65	9	2016	RR
2	103	9	2016	RS
3	492	9	2016	SP
4	16	10	2016	PB
5	52	10	2016	AL
6	56	10	2016	PI
7	65	10	2016	SE
8	65	10	2016	RR
9	220	10	2016	RN
10	271	10	2016	ES

>>> Here data shows state wise and month on month sales count is highlighted.

2. Distribution of customers across the states in Brazil

```
SELECT DISTINCT
COUNT(cust.customer_id) AS customer_count,cust.customer_state
FROM `Case_Target_sql.Customers` cust
INNER JOIN `Case_Target_sql.Orders` ord
ON ord.customer_id=cust.customer_id
GROUP BY cust.customer_state
ORDER BY customer_count DESC ;
```

The screenshot shows the Google BigQuery interface. On the left, the sidebar displays the project 'my-targetsql-project' and the dataset 'Case_Target_sql'. The 'Orders' table is selected. In the main area, a query is being run:

```
1 SELECT DISTINCT
2 COUNT(cust.customer_id) AS customer_count,cust.customer_state
3 FROM `Case_Target_sql.Customers` cust
4 INNER JOIN `Case_Target_sql.Orders` ord
5 ON ord.customer_id=cust.customer_id
6 GROUP BY cust.customer_state
7 ORDER BY customer_count DESC;
```

The 'Query results' section shows the following data:

Row	customer_count	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF
9	2033	ES
10	2020	GO

>>> Out of 27 Brazilian states Sao Paulo leads with the number of maximum customers purchases of around 40% of the total orders. Rio de Janeiro and Minas Gerais following Sao Paulo where together 3 of these states have customers of around 66% out of total. SP and MG are the two major states with highest number of populations.

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

- 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

```

SELECT Total_monthly_order_cost,Total_yearly_order_cost,
ROUND((Total_monthly_order_cost*100)/Total_yearly_order_cost,2) as Cost_of_ord_percentage,Month,Year
FROM
(SELECT DISTINCT
Round(SUM(payment_value) OVER (PARTITION BY Month,Year),2) as Total_monthly_order_cost,
ROUND(SUM (payment_value) OVER (PARTITION BY Year),2) as Total_yearly_order_cost,
Month,Year
FROM
(SELECT order_purchase_timestamp,payment_value,
extract(month from order_purchase_timestamp) AS Month,
extract(year from order_purchase_timestamp) AS Year
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Payments` pay
ON ord.order_id=pay.order_id
) temp
WHERE Month < 9 and Year between 2016 and 2019
ORDER BY Year,Month) temp2
    
```

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar includes sections for Analysis, SQL workspace, Data transfers, Scheduled queries, Analytics Hub, Dataform, Migration, SQL translation, Administration, Monitoring, Capacity management, BI Engine, and Policy tags. The main area displays an 'Unsaved query' with the following SQL code:

```

1 SELECT Total_monthly_order_cost,Total_yearly_order_cost,
2 ROUND((Total_monthly_order_cost*100)/Total_yearly_order_cost,2) as Cost_of_ord_percentage,Month,Year
3 FROM
4 (SELECT DISTINCT
5 Round(SUM(payment_value) OVER (PARTITION BY Month,Year),2) as Total_monthly_order_cost,
6 ROUND(SUM (payment_value) OVER (PARTITION BY Year),2) as Total_yearly_order_cost,
7 Month,Year
8 FROM
9 (SELECT order.purchase_timestamp,payment_value,
10 extract(month from order.purchase_timestamp) AS Month,
11 extract(year from order.purchase_timestamp) AS Year
12 FROM `Case_Target_sql.Orders` ord
13 INNER JOIN `Case_Target_sql.Payments` pay
14 ON ord.order_id=pay.order_id
15 ) temp
16 WHERE Month < 9 and Year between 2016 and 2019
17 ORDER BY Year,Month) temp2
18
19
    
```

The 'Query results' section shows a table with 10 rows of data:

Row	Total_monthly_order_cost	Total_yearly_order_cost	Cost_of_ord_percentage	Month	Year
1	138488.04	3669022.12	3.77	1	2017
2	291908.01	3669022.12	7.96	2	2017
3	449863.6	3669022.12	12.26	3	2017
4	417786.03	3669022.12	11.39	4	2017
5	592918.82	3669022.12	16.16	5	2017
6	511276.38	3669022.12	13.93	6	2017
7	592382.92	3669022.12	16.15	7	2017
8	674396.32	3669022.12	18.38	8	2017
9	1115004.18	8694733.84	12.82	1	2018
10	992463.34	8694733.84	11.41	2	2018

>>> Total cost of orders between month Jan to Aug for years 2017 & 2018. Shows significant increase after first two months where it reached maximum of 18% of total cost during August 2017. For an entire year 2018 it is quite steady trend compared to year 2017 with an average 12.5% of total cost of orders.

- Mean & Sum of price and freight value by customer state

```

SELECT cust.customer_state,
ROUND(SUM(freight_value),2) as total_freight_value,
    
```

```

ROUND(AVG(freight_value),2) as avg_freight_value,
ROUND(SUM(price),2) as total_price_value,
ROUND(AVG(price),2) as avg_price_value
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Order_items` oit
ON ord.order_id=oit.order_id
INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY avg_freight_value desc;

```

5.

;

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar includes sections for Analysis, Data transfers, Scheduled queries, Analytics Hub, Dataform, Migration, SQL translation, Administration, Monitoring, Capacity management, BI Engine, and Policy tags. The main area displays a query editor with the following code:

```

1. SELECT cust.customer_state,
2. ROUND(SUM(freight_value),2) as total_freight_value,
3. ROUND(AVG(freight_value),2) as avg_freight_value,
4. ROUND(SUM(price),2) as total_price_value,
5. ROUND(AVG(price),2) as avg_price_value
6. FROM `Case_Target_sql.Orders` ord
7. INNER JOIN `Case_Target_sql.Order_items` oit
8. ON ord.order_id=oit.order_id
9. INNER JOIN `Case_Target_sql.Customers` cust
10. ON cust.customer_id=ord.customer_id
11. GROUP BY cust.customer_state
12. ORDER BY avg_freight_value desc;
13

```

Below the code, the "Query results" section shows a table with the following data:

customer_state	total_freight_value	avg_freight_value	total_price_value	avg_price_value
RR	2235.19	42.98	7829.43	150.57
PB	25719.73	42.72	115268.08	191.48
RO	11417.38	41.07	46140.64	165.97
AC	3686.75	40.07	15982.95	173.73
PI	21218.2	39.15	86914.08	160.36
MA	31523.77	38.26	119648.22	145.2
TO	11732.68	37.25	49621.74	157.53
SE	14111.47	36.65	58920.85	153.04
AL	15914.59	35.84	80314.81	180.89
PA	38699.3	35.83	178947.81	165.69

>>> Here just a view for the states where average freight cost is higher than rest of the states. Where the more populated states have lower freight cost. Average Freight value in the states where it is highest like RR, PB, RO & AC is twice more than compared to states with lower freight cost.

5. Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT order_id,order_date,delivery_date,estimated_del_date,
```

```

date_diff(delivery_date,order_date,day) as delivered_in_days,
date_diff(estimated_del_date,order_date,day) as estimated_days,
date_diff(delivery_date,estimated_del_date,day) as days_delta_with_estimated
FROM
(SELECT order_id,
extract(date from order_purchase_timestamp) as order_date,
extract(date from order_delivered_customer_date) as delivery_date,
extract(date from order_estimated_delivery_date) as estimated_del_date
FROM `Case_Target_sql.Orders`  

) temp
WHERE temp.delivery_date IS NOT NULL;

```

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the project structure under 'my-targetsql-project' with 'Case_Target_sql' selected. The main area shows an unsaved query with the following SQL code:

```

1 SELECT order_id,order_date,delivery_date,estimated_del_date,
2 date_diff(delivery_date,order_date,day) as delivered_in_days,
3 date_diff(estimated_del_date,order_date,day) as estimated_days,
4 date_diff(delivery_date,estimated_del_date,day) as days_delta_with_estimated
5 FROM
6 (SELECT order_id,
7 extract(date from order_purchase_timestamp) as order_date,
8 extract(date from order_delivered_customer_date) as delivery_date,
9 extract(date from order_estimated_delivery_date) as estimated_del_date
10 FROM `Case_Target_sql.Orders`  

11 ) temp
12 WHERE temp.delivery_date IS NOT NULL;
13

```

The 'RESULTS' tab is selected in the query results pane, displaying the following data:

Row	order_id	order_date	delivery_date	estimated_del_date	delivered_in_days	estimated_days	days_delta_with_estimated
1	770d331c84e5b214bd9dc70a1...	2016-10-07	2016-10-14	2016-11-29	7	53	-46
2	2e45c33d29cb8ff8b1c86cc28...	2016-10-09	2016-11-09	2016-12-08	31	60	-29
3	dabf2b0e35b423f94618bf965f...	2016-10-09	2016-10-16	2016-11-30	7	52	-45
4	8beb59392e21af5eb9547ae1a...	2016-10-08	2016-10-19	2016-11-30	11	53	-42
5	65d1e226dfaeb8cdc42f66542...	2016-10-03	2016-11-08	2016-11-25	36	53	-17
6	cecf8f5f7a13e5ah934a48hee9e...	2017-03-17	2017-04-07	2017-05-18	21	62	-41

Below the table, there are navigation controls for results per page (50), page 1-50 of 96476, and refresh buttons.

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

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

SELECT *

```

FROM
(SELECT order_id,
extract(date from order_purchase_timestamp) as order_purchase_date,
extract(date from order_delivered_customer_date) as order_delivered_cdate,
extract(date from order_estimated_delivery_date) as est_del_date,
date_diff( order_delivered_customer_date,order_purchase_timestamp, day) as time_to_delivery,
date_diff(order_delivered_customer_date,order_estimated_delivery_date, day) as diff_estimated_delivery
FROM `Case_Target_sql.Orders`  

) temp
WHERE order_delivered_cdate IS NOT NULL
ORDER BY diff_estimated_delivery DESC;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar includes sections for Analysis, SQL workspace, Data transfers, Scheduled queries, Analytics Hub, Dataform, Migration, SQL translation, Administration, Monitoring, Capacity management, BI Engine, and Policy tags. The main area shows a query editor with the following code:

```

1 SELECT *
2 FROM
3 (SELECT order_id,
4 extract(date from order_purchase_timestamp) as order_purchase_date,
5 extract(date from order_delivered_customer_date) as order_delivered_cdate,
6 extract(date from order_estimated_delivery_date) as est_del_date,
7 date_diff( order_delivered_customer_date,order_purchase_timestamp, day) as time_to_delivery,
8 date_diff(order_delivered_customer_date,order_estimated_delivery_date, day) as diff_estimated_delivery
9 FROM `Case_Target_sql.Orders`  

10 ) temp
11 WHERE order_delivered_cdate IS NOT NULL
12 ORDER BY diff_estimated_delivery DESC;
13

```

The status bar at the top right indicates "This query will process 5.48 MB when run." Below the code, the "Query results" section displays a table with 10 rows of data. The columns are: Row, order_id, order_purchase_date, order_delivered_cdate, est_del_date, time_to_delivery, and diff_estimated_delivery. The data is as follows:

Row	order_id	order_purchase_date	order_delivered_cdate	est_del_date	time_to_delivery	diff_estimated_delivery
1	1b3190b2dfa9d789e1f14c05b...	2016-02-23	2018-09-19	2018-03-15	208	188
2	ca07593549f1816d26a572e06...	2017-02-21	2017-09-19	2017-03-22	209	181
3	47f404249edcc3ae9199792...	2016-01-03	2018-07-13	2018-01-19	191	175
4	2f6324feff907e3ea3f2a9e9550...	2017-09-13	2017-09-19	2017-04-05	189	167
5	285b9426d6982034523a855f...	2017-09-08	2017-09-19	2017-04-06	194	166
6	4400d017af552815d15a9e41a...	2017-03-07	2017-09-19	2017-04-07	195	165
7	c27815f7e3d00926b5855262...	2017-03-15	2017-09-19	2017-04-10	187	162
8	0f4519c5f1c541ddc9f21b3d...	2017-03-09	2017-09-19	2017-04-11	194	161
9	d24e6541120cea179a11a517...	2017-06-12	2017-12-04	2017-06-26	175	161
10	2d7561026d542c8dbd8f0dea...	2017-03-15	2017-09-19	2017-04-13	188	159

At the bottom, there are buttons for "SAVE RESULTS", "EXPLORE DATA", and "REFRESH".

>>>Here required date with just added with some order to showcase what is the maximum diff_estimated_delivery.

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

```

SELECT cust.customer_state,
ROUND(AVG(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_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Order_items` oit
ON ord.order_id=oit.order_id
INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY mean_time_to_delivery desc;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar includes sections for Analysis, Migration, Administration, and BI Engine. The main area displays a query editor with the following code:

```

1 SELECT cust.customer_state,
2 ROUND(AVG(freight_value),2) as mean_freight_value,
3 ROUND(AVG(date_diff( order_delivered_customer_date,order_purchase_timestamp, day)),2) as mean_time_to_delivery,
4 ROUND(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
5 FROM `Case_Target_sql.Orders` ord
6 INNER JOIN `Case_Target_sql.Order_items` oit
7 ON ord.order_id=oit.order_id
8 INNER JOIN `Case_Target_sql.Customers` cust
9 ON cust.customer_id=ord.customer_id
10 GROUP BY cust.customer_state
11 ORDER BY mean_time_to_delivery desc;
12

```

The results section shows a table with the following data:

customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
RR	42.98	27.83	-17.43
AP	34.01	27.75	-17.44
AM	33.21	25.96	-18.98
AL	35.84	23.99	-7.98
PA	35.83	23.3	-13.37
MA	38.26	21.2	-9.11
SE	36.65	20.98	-9.17
CE	32.71	20.54	-10.26
AC	40.07	20.33	-20.01
PB	42.72	20.12	-12.15

>>> Average time for the delivery around state like RR, AP, AM looks higher, also average freight value looks higher in these states.

4. Sort the data to get the following:

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

```

SELECT cust.customer_state,
ROUND(AVG(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_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Order_items` oit
ON ord.order_id=oit.order_id
INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY mean_freight_value ASC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays the project structure under 'my-targetsql-project' with various datasets like Case_Target_sql, Order_items, and Order_reviews. The main area shows a query editor with the following SQL code:

```

1 SELECT cust.customer_state,
2 ROUND(AVG(freight_value),2) as mean_freight_value,
3 ROUND(AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp, day)),2) as mean_time_to_delivery,
4 ROUND(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
5 FROM `Case_Target_sql.Orders` ord
6 INNER JOIN `Case_Target_sql.Order_items` oit
7 ON ord.order_id=oit.order_id
8 INNER JOIN `Case_Target_sql.Customers` cust
9 ON cust.customer_id=ord.customer_id
10 GROUP BY cust.customer_state
11 ORDER BY mean_freight_value ASC
12 LIMIT 5;
13

```

The 'RESULTS' tab is selected in the 'Query results' section, showing the following data:

customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
SP	15.15	8.26	-10.27
PR	20.53	11.48	-12.53
MG	20.63	11.52	-12.4
RJ	20.96	14.69	-11.14
DF	21.04	12.5	-11.27

>>>The states SP, PR, MG & RJ with less average freight value are one of the most populated states where most customers are located in.

```

SELECT cust.customer_state,
ROUND(AVG(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_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Order_items` oit

```

```

ON ord.order_id=oit.order_id
INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY mean_freight_value DESC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists resources under 'my-targetsql-project', including 'Case_Target_sql' which contains 'Customers' and 'Order_Items'. The main area displays a query editor with the following SQL code:

```

1 SELECT cust.customer_state,
2 ROUND(AVG(freight_value),2) as mean_freight_value,
3 ROUND(AVG(date_diff( order_delivered_customer_date,order_purchase_timestamp, day)),2) as mean_time_to_delivery,
4 ROUND(AVG(date_diff( order_delivered_customer_date,order_estimated_delivery_date, day)),2) as mean_diff_estimated_delivery
5 FROM `Case_Target_sql.Orders` ord
6 INNER JOIN `Case_Target_sql.Order_items` oit
7 ON ord.order_id=oit.order_id
8 INNER JOIN `Case_Target_sql.Customers` cust
9 ON cust.customer_id=ord.customer_id
10 GROUP BY cust.customer_state
11 ORDER BY mean_freight_value DESC
12 LIMIT 5;
13

```

Below the code, the 'Query results' section shows a table with the following data:

customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
RR	42.98	27.83	-17.43
PB	42.72	20.12	-12.15
RO	41.07	19.28	-19.08
AC	40.07	20.33	-20.01
PI	39.15	18.93	-10.68

>>>The state like RR,PB,RO & AC with most average freight cost is having twice more than the states with lowest freight cost. Similar strategies like states with low cost, shall be planned to reduce the freight cost.

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

```

SELECT cust.customer_state,
ROUND(AVG(date_diff( order_delivered_customer_date,order_purchase_timestamp, day)),2) as avg_time_to_delivery
_days,
FROM `Case_Target_sql.Orders` ord

```

```

INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY avg_time_to_delivery_days DESC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. The left sidebar is titled 'Explorer' and lists resources under 'my-targetsql-project' and 'Case_Target_sql'. The 'Customers' table is selected. The main area displays a query in the 'Query Editor' and its results in the 'Query results' table.

Query Editor:

```

1 SELECT cust.customer_state,
2 ROUND(AVG(date_diff( order_delivered_customer_date,order_purchase_timestamp, day)),2) as avg_time_to_delivery_days,
3 FROM `Case_Target_sql.Orders` ord
4 INNER JOIN `Case_Target_sql.Customers` cust
5 ON cust.customer_id=ord.customer_id
6 GROUP BY cust.customer_state
7 ORDER BY avg_time_to_delivery_days DESC
8 LIMIT 5;
9

```

Query results:

customer_state	avg_time_to_delivery_days
RR	28.98
AP	26.73
AM	25.99
AL	24.04
PA	23.32

>>>States RR, AP, AM, AL & PA are the topmost states where average delivery time is highest up to 29 days.

```

SELECT cust.customer_state,
ROUND(AVG(date_diff( order_delivered_customer_date,order_purchase_timestamp, day)),2) as avg_time_to_delivery
_days,
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id

```

```

GROUP BY cust.customer_state
ORDER BY avg_time_to_delivery_days ASC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists resources under 'my-targetsql-project', including 'Case_Target_sql' which contains 'Customers', 'Geolocation', 'Order_items', 'Order_reviews', 'Orders', 'Payments', 'Products', and 'Sellers'. The 'Customers' table is selected. In the main area, a query editor window displays the following SQL code:

```

1 SELECT cust.customer_state,
2    ROUND(AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp, day)),2) as avg_time_to_delivery_days,
3   FROM `Case_Target_sql.Orders` ord
4  INNER JOIN `Case_Target_sql.Customers` cust
5    ON cust.customer_id=ord.customer_id
6   GROUP BY cust.customer_state
7   ORDER BY avg_time_to_delivery_days ASC
8   LIMIT 5;
9

```

Below the code, the 'Query results' section shows a table with the following data:

customer_state	avg_time_to_delivery_days
SP	8.3
PR	11.53
MG	11.54
DF	12.51
SC	14.48

>>>These states are those where the avg delivery time is less than compared to rest of the states. State Sao Paulo with lowest average delivery time of 1 week only.

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

```

SELECT cust.customer_state,
ROUND(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date, day)),2) as avg_diff_estimated_delivery
FROM `Case_Target_sql.Orders` ord

```

```

INNER JOIN `Case_Target_sql.Customers` cust
ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY avg_diff_estimated_delivery ASC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists resources under 'my-targetsql-project' and 'Case_Target_sql'. The 'Customers' table is selected. In the main area, a query is displayed:

```

1 SELECT cust.customer_state,
2 ROUND(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date, day)),2) as avg_diff_estimated_delivery
3 FROM `Case_Target_sql.Orders` ord
4 INNER JOIN `Case_Target_sql.Customers` cust
5 ON cust.customer_id=ord.customer_id
6 GROUP BY cust.customer_state
7 ORDER BY avg_diff_estimated_delivery ASC
8 LIMIT 5;
9

```

The 'Query results' section shows the following data:

customer_state	avg_diff_estimated_delivery
AC	-19.76
RO	-19.13
AP	-18.73
AM	-18.61
RR	-16.41

>>> Here these states with average diff estimated delivery shows us that how faster the order was delivered compared to the estimated day of deliver. Here we can see some states like AC & RO have where order was delivered earlier with average value of 19 days.

```

SELECT cust.customer_state,
ROUND(AVG(date_diff(order_delivered_customer_date,order_estimated_delivery_date, day)),2) as avg_diff_estimated_delivery
FROM `Case_Target_sql.Orders` ord
INNER JOIN `Case_Target_sql.Customers` cust

```

```

ON cust.customer_id=ord.customer_id
GROUP BY cust.customer_state
ORDER BY avg_diff_estimated_delivery DESC
LIMIT 5;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists resources under 'my-targetsql-project' and 'Case_Target_sql'. In the center, a query editor window displays a SQL query to find states where delivery is delayed by more than 7 days. The 'RESULTS' tab is selected, showing a table with five rows of data. The table has columns 'customer_state' and 'avg_diff_estimated_delivery'. The data is as follows:

customer_state	avg_diff_estimated_delivery
AL	-7.95
MA	-8.77
SE	-9.17
ES	-9.62
BA	-9.93

>>>Here states like AL, MA & other 3 where we can say order is delivered just 1 week before.

Overall delivery estimated difference is always > 7days in every states.

6. Payment type analysis:

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

```

SELECT COUNT(ord.order_id) as order_count,payment_type,
extract(month from ord.order_purchase_timestamp) AS Month,
extract(year from ord.order_purchase_timestamp) AS Year
FROM `Case_Target_sql.Payments` pay
INNER JOIN `Case_Target_sql.Orders` ord

```

```

ON ord.order_id=pay.order_id
GROUP BY Month,Year,payment_type
ORDER BY Year,Month,payment_type
;

```

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar displays the project 'my-targetsql-project' and its datasets, including 'Case_Target_sql'. Under 'Case_Target_sql', several tables are listed: Customers, Geolocation, Order_items, Order_reviews, Orders, Payments, Products, and Sellers. The 'Orders' table is currently selected. The main area shows the query results for the following SQL code:

```

1 SELECT COUNT(ord.order_id) as order_count,payment_type,
2 extract(month from ord.order.purchase_timestamp) AS Month,
3 extract(year from ord.order.purchase_timestamp) AS Year
4 FROM `Case_Target_sql.Payments` pay
5 INNER JOIN `Case_Target_sql.Orders` ord
6 ON ord.order_id=pay.order_id
7 GROUP BY Month,Year,payment_type
8 ORDER BY Year,Month,payment_type
9 ;

```

The results table has columns: Row, order_count, payment_type, Month, and Year. The data is as follows:

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

>>> Payment type Credit Card is by far the most used by customer to purchase an order.

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

```
SELECT COUNT(order_id) as order_count,payment_installments
FROM `Case_Target_sql.Payments`
GROUP BY payment_installments;
```

The screenshot shows the Google BigQuery interface. On the left, the 'Explorer' sidebar lists resources under 'my-targetsql-project'. A query is running in the main area:

```
1 SELECT COUNT(order_id) as order_count,payment_installments
2 FROM `Case_Target_sql.Payments`
3 GROUP BY payment_installments
4 ;
```

The 'Query results' section displays the following data:

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

>>> Maximum customers opt for the payment installment number between 2 to 4 of about nearly 75% of the total payment types. While only 2 orders with full payment are observed.

Actionable Insights:	Recommendations:
<p>Initially in October 2016 there is increase in order purchases/sales which again declines for next 2 months. However, during year 2017 there was upward trend in order purchases/sales for whole year. Also observed some sudden spike during Month of November 2017 as its month with Black Friday sale and to be considered</p>	<p>As we can see there is increase in the demand during the specific seasonal sale like black Friday, strategies should be defined accordingly. More count of products shall be made available. Staff shall be increased during such peak season. Inventory shall be well maintained during season. Server maintenance shall be done prior to seasonal peak.</p>

seasonal. Also, 1st quarter of the year 2018 shows high sales which slowly declines in next quarter and sharply declines further from the end of 3rd quarter. Overall, we can say there is growing trend on e-commerce in Brazil since 2016 till 2018 as it has grown quite a lot than initial stage.	Quality check or audits shall be conducted (internal or external) before season to ensure best products and services given to customer. Overall looking at the growing trend considering percentage growth in previous years; accordingly, varied product category, services and support can be increased.
By looking at time of purchases we can say customers are active during daytime and maximum purchases are made during afternoon time following nighttime at second place. Morning time purchases are still made at commendable rate however dawn phase is the least to have activity from customers.	More support shall be provided during peak hours of purchases. Customer support shall be planned accordingly with more resources during such hours. Minor maintenance activities can be performed/downtime window when less activity of customer. E.g., Dawn phase. Staff can also get some breathing period during this time.
Out of 27 Brazilian states Sao Paulo leads with the number of maximum customers purchases of around 40% of the total orders. Rio de Jannero and Minas Gerais following Sao Paulo where together 3 of these states have customers of around 66% out of total. SP and MG are the two major states with highest number of populations.	As most populated states have a greater number of customers. Product category, inventory, customer support and quality service shall be on priority in these states. Whereas in states, where there are less customers placing orders can be focused with more marketing of the products and brand to attract new customer.
Total cost of orders between month Jan to Aug for years 2017 & 2018. Shows significant increase after first two months where it reached maximum of 18% of total cost during August 2017. For an entire year 2018 it is quite steady trend compared to year 2017 with an average 12.5% of total cost of orders.	As it is observed that there is usual growing trend/pattern during some specific months, business strategy can be aligned in way to increase the profit margin during peak time. It is also advised to encourage customer during other time through various marketing strategies when it is not season.
There are some states where average freight cost is higher than rest of the states. Where the more populated states have lower freight cost. Average Freight value in the states where it is highest like RR, PB, RO & AC is twice more than compared to states with lower freight cost.	Freight cost can be reduced in such states where it is very high compared to more popular states. Here delivery time also is major factor where along with freight cost delivery time is also high. In such states where courier companies with slow services can be dropped off and alternative can be found on to improve services or else some cost negotiation shall be done with current courier services.
Average time for the delivery around state like RR, AP, AM looks higher, also average freight value looks higher in these states.	For the delivery time in these states there can be several affecting factors which might need to be found of with further in-depth analysis and counter measures can be defined accordingly.
The states SP, PR, MG & RJ with less average freight value are one of the most populated states where most customers are located in. The state like RR, PB, RO & AC with most average freight cost is having twice more than the states with lowest freight cost.	Strategies in such states where freight cost is low can also be studied to understand how and why it is less compared to the higher ones. Similar strategies can be applied in the states where it is high to reduce freight cost.
These states are those where the average delivery time is less than compared to rest of the states. State Sao Paulo with lowest average delivery time of 1 week only.	Factors like traffic scenarios, roads, locality, number of stores shipping products & their availability can be analyzed to check how it is affecting the deliveries in these states for improvement of other states and its strategies.
States RR, AP, AM, AL & PA are the topmost states where average delivery time is highest up to 29 days.	These states can be kept as an improvement objective as in these states average delivery time is very high, and we may risk losing customer due to delay in the delivery or even long time taken for delivery.

Here these states with average diff estimated delivery shows us that how faster the order was delivered compared to the estimated day of deliver. Here we can see some states like AC & RO have where order was delivered earlier with average value of 19 days	Average diff estimated delivery can also be one of the KPI factor to be added for next year. If delivery is always faster in such states, then estimated days shall also not vary much and can be kept likewise and with less difference. It can be tested and slowly reduced to 1 week.
Here states like AL, MA & other 3 where we can say order is delivered just 1 week before. Overall delivery estimated difference is always > 7days in every states.	Here, it seems not much difference between actual delivery date and estimated date. However, it can still be kept low priority objective to reduce it to half week amount.
For Payment type maximum customers opt for the payment installment number between 2 to 4 of about nearly 75% of the total payment types. While only 2 orders with full payment are observed. Also, payment type Credit Card is by far the most used by customer to purchase an order.	For payment type, where most cost is involved then there can be more profit margin. It could be one of the potential areas where profit can be increased from by applying slightly higher interest or less discount in that range may work. Whereas to have instant cashflow we can encourage customers to go for full payment with some minor discounts.