

Target: SQL

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analyzing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

Dataset: <https://drive.google.com/drive/folders/1TGEc66YKbD443nsIRi1bWgVd238gJCnb>

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

The column description for these csv files is given below.

The customers.csv contain following features:

Features	Description
customer_id	ID of the consumer who made the purchase
customer_unique_id	Unique ID of the consumer
customer_zip_code_prefix	Zip Code of consumer's location
customer_city	Name of the City from where order is made
customer_state	State Code from where order is made (Eg. são paulo - SP)

The sellers.csv contains following features:

Features	Description
seller_id	Unique ID of the seller registered
seller_zip_code_prefix	Zip Code of the seller's location
seller_city	Name of the City of the seller
seller_state	State Code (Eg. são paulo - SP)

The order_items.csv contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
order_item_id	A Unique ID given to each item ordered in the order
product_id	A Unique ID given to each product available on the site
seller_id	Unique ID of the seller registered in Target
shipping_limit_date	The date before which the ordered product must be shipped
price	Actual price of the products ordered
freight_value	Price rate at which a product is delivered from one point to another

The geolocations.csv contain following features:

Features	Description
geolocation_zip_code_prefix	First 5 digits of Zip Code
geolocation_lat	Latitude
geolocation_lng	Longitude
geolocation_city	City
geolocation_state	State

The payments.csv contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
payment_sequential	Sequences of the payments made in case of EMI
payment_type	Mode of payment used (Eg. Credit Card)
payment_installments	Number of installments in case of EMI purchase
payment_value	Total amount paid for the purchase order

The orders.csv contain following features:

Features	Description
order_id	A Unique ID of order made by the consumers
customer_id	ID of the consumer who made the purchase
order_status	Status of the order made i.e. delivered, shipped, etc.
order_purchase_timestamp	Timestamp of the purchase
order_delivered_carrier_date	Delivery date at which carrier made the delivery
order_delivered_customer_date	Date at which customer got the product
order_estimated_delivery_date	Estimated delivery date of the products

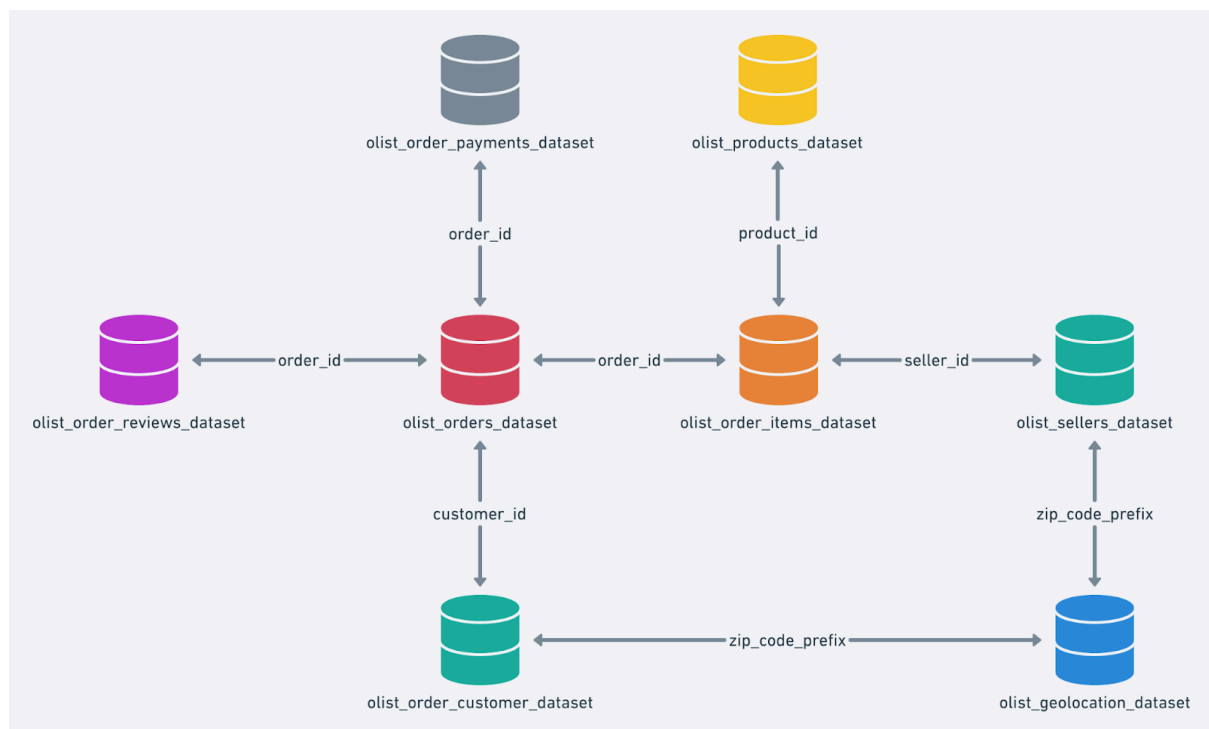
The reviews.csv contain following features:

Features	Description
review_id	ID of the review given on the product ordered by the order id
order_id	A Unique ID of order made by the consumers
review_score	Review score given by the customer for each order on a scale of 1-5
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order
review_creation_date	Timestamp of the review when it is created
review_answer_timestamp	Timestamp of the review answered

The products.csv contain following features:

Features	Description
product_id	A Unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	Length of the string which specifies the name given to the products ordered
product_description_lenght	Length of the description written for each product ordered on the site
product_photos_qty	Number of photos of each product ordered available on the shopping portal
product_weight_g	Weight of the products ordered in grams
product_length_cm	Length of the products ordered in centimeters
product_height_cm	Height of the products ordered in centimeters
product_width_cm	Width of the product ordered in centimeters

Dataset schema:



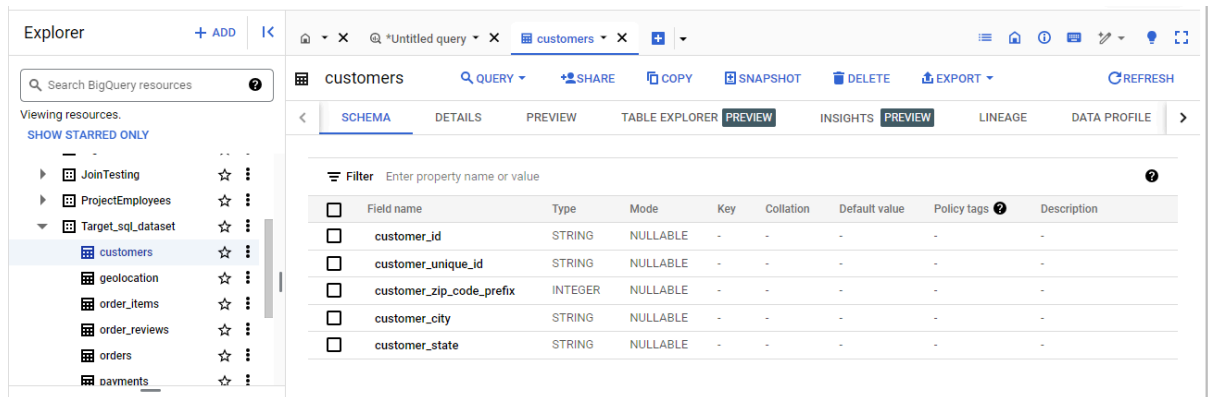
Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

What does 'good' look like?

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

1. Data type of all columns in the "customers" table.



The screenshot shows the Google BigQuery interface. On the left, the 'Explorer' pane lists resources, with 'customers' selected under 'Target_sql_dataset'. The main pane shows the 'customers' table schema with the following columns:

Field name	Type	Mode	Key	Collation	Default value	Policy tags	Description
customer_id	STRING	NULLABLE	-	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-	-

Inference:

customer_id is in STRING data type and contains 99441 unique.

customer_unique_id is in STRING data type contains duplicates.

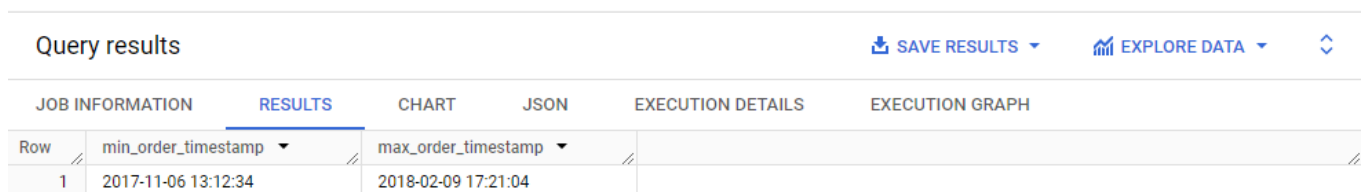
customer_zip_code_prefix is a integer data type contains duplicates.

customer_city is in STRING data type contains duplicates.

customer_state is in STRING data type contains duplicates.

2. Get the time range between which the orders were placed.

```
Select concat(min_order_date, " ", min_order_time ) AS min_order_timestamp,
concat(max_order_date, " ", max_order_time ) AS max_order_timestamp
FROM (
Select extract (date from MIN(order_purchase_timestamp)) As min_order_date,
extract (time from MIN(order_purchase_timestamp)) As min_order_time,
extract (date from MAX(order_purchase_timestamp)) As max_order_date,
extract (time from MAX(order_purchase_timestamp)) As max_order_time
from aerial-prism-427113-h2.Target_sql_dataset.orders
WHERE order_status = 'created')X
```



The screenshot shows the 'Query results' section of the Google BigQuery interface. The query results are displayed in a table with two columns: 'min_order_timestamp' and 'max_order_timestamp'.

Row	min_order_timestamp	max_order_timestamp
1	2017-11-06 13:12:34	2018-02-09 17:21:04

Inference: In order_status column in orders table there are 8 types of values are present i.e created, shipped, approved, cancelled, invoiced, delivered, processing and unavailable.

So to check the minimum and maximum timestamp in which orders were placed I filter the data based on order_status equal to "created".

Min_order_timestamp = 2017-11-06 13:12:34

Max_order_timestamp = 2018-02-09 17:21:04

Conclusion 2:

```
Select extract (date from MIN(order_purchase_timestamp)) As min_order_date,
       extract (time from MIN(order_purchase_timestamp)) As min_order_time,
       extract (date from MAX(order_purchase_timestamp)) As max_order_date,
       extract (time from MAX(order_purchase_timestamp)) As max_order_time
from aerial-prism-427113-h2.Target_sql_dataset.orders
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	min_order_date	min_order_time	max_order_date	max_order_time		
1	2016-09-04	21:15:19	2018-10-17	17:30:18		

3. Count the Cities & States of customers who ordered during the given period.

```
SELECT COUNT(distinct customer_city)total_city, COUNT(distinct
customer_state)total_states FROM (
SELECT o.customer_id, c.customer_city, c.customer_state
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers AS c USING
(customer_id)
WHERE order_purchase_timestamp
BETWEEN (SELECT MIN(order_purchase_timestamp) FROM
aerial-prism-427113-h2.Target_sql_dataset.orders)
AND (SELECT MAX(order_purchase_timestamp) FROM
aerial-prism-427113-h2.Target_sql_dataset.orders))
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	total_city	total_states				
1	4119	27				

Inference: Joining the orders and customers tables to identify the customers who placed orders during the given range, and finding the cities and states from which these orders were made.

Conclusion 2:

```
SELECT customer_city, customer_state, COUNT(*) AS order_count
FROM(
SELECT o.customer_id, c.customer_city, c.customer_state
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers AS c USING
(customer_id)
WHERE order_purchase_timestamp
BETWEEN (SELECT MIN(order_purchase_timestamp) FROM
aerial-prism-427113-h2.Target_sql_dataset.orders)
AND (SELECT MAX(order_purchase_timestamp) FROM
aerial-prism-427113-h2.Target_sql_dataset.orders))
GROUP BY customer_state, customer_city
ORDER BY order_count desc
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_city	customer_state	order_count			
1	sao paulo	SP	15540			
2	rio de janeiro	RJ	6882			
3	belo horizonte	MG	2773			
4	brasilia	DF	2131			
5	curitiba	PR	1521			
6	campinas	SP	1444			
7	porto alegre	RS	1379			
8	salvador	BA	1245			
9	guarulhos	SP	1189			
10	sao bernardo do campo	SP	938			

Inference: Most of the order are from sao paulo city which belongs to SP state.

2. In-depth Exploration:

1. Is there a growing trend in the no. of orders placed over the past years?

```
SELECT year_col, count(order_id) no_of_orders FROM(
select order_id, customer_id, EXTRACT(year from order_purchase_timestamp) year_col
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
WHERE order_status = 'created')
GROUP BY year_col
```

Query results SAVE RESULTS

JOB INFORMATIONRESULTSCHARTJSONEXECUTION DETAILSEXECUTION GRAPH

Row	year_col	no_of_orders	
1	2017	4	
2	2018	1	

INFERENCE: the number of order placed during 2017 is 4 and in 2018 is 1 so we can conclude that placing order has reduced significantly to 75%.

Conclusion2:

```
SELECT *, ROUND(((lead_col - no_of_orders)/no_of_orders)*100, 2) As per_change FROM
(
SELECT year_col, count(order_id) no_of_orders, lead(count(order_id))OVER(order by
year_col) lead_col,
FROM(
select order_id,customer_id, EXTRACT(year from order_purchase_timestamp) year_col
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
)
GROUP BY year_col
ORDER BY year_col)
```

Query results

[SAVE RESULTS](#)

JOB INFORMATIONRESULTSCHARTJSONEXECUTION DETAILSEXECUTION GRAPH

Row	year_col	no_of_orders	lead_col	per_change	
1	2016	329	45101	13608.51	
2	2017	45101	54011	19.76	
3	2018	54011	null	null	

INFERENCE:If we compare the 2017 and 2018 data the number of orders has increased by 19.76% which is a significant increase even though the 2018 year is not completed yet.

2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
SELECT year_col, month_col, COUNT(*) AS order_count
FROM (
select order_id,customer_id,EXTRACT(month from order_purchase_timestamp) month_col,
EXTRACT(year from order_purchase_timestamp) year_col
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
)X
GROUP BY year_col,month_col
order by year_col, month_col
```


Query results

JOB INFORMATION		RESULTS	CHART	JSON	EXI
Row	year_col ▼	month_col ▼	order_count ▼		
1	2016	9	4		
2	2016	10	324		
3	2016	12	1		
4	2017	1	800		
5	2017	2	1780		
6	2017	3	2682		
7	2017	4	2404		
8	2017	5	3700		
9	2017	6	3245		
10	2017	7	4026		

INFERENCE:From November to December 2016, the orders were low. However, once 2017 started, the orders began increasing, growing month by month.

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

```

SELECT sel_time, COUNT(*) as no_of_purchases
FROM (
SELECT *,
    CASE WHEN purchase_time between "00:00:00" and "06:59:59" THEN "Dawn"
         WHEN purchase_time between "07:00:00" and "12:59:59" THEN "Mornings"
         WHEN purchase_time between "13:00:00" and "18:59:59" THEN "Afternoon"
         WHEN purchase_time between "19:00:00" and "23:59:59" THEN "Night"
    END AS sel_time
FROM (
SELECT customer_id, extract(date from order_purchase_timestamp) purchase_date,
extract(hour from order_purchase_timestamp) purchase_hours,
extract(time from order_purchase_timestamp) purchase_time
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
)a)b
GROUP BY sel_time
ORDER BY no_of_purchases

```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	sel_time	no_of_purchases				
1	Dawn	5242				
2	Mornings	27733				
3	Night	28331				
4	Afternoon	38135				

INFERENCE: The most of the orders are made during the afternoon between 1PM to 6PM and the order size is 39135 and the Dawn time orders are less i.e 5242. During the span of 773 days.

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

1. Get the month on month no. of orders placed in each state.

```

SELECT customer_state,p_month, COUNT(*) as no_of_orders
FROM (
SELECT order_id, customer_id, extract(year from order_purchase_timestamp) p_year,
extract(month from order_purchase_timestamp) p_month
FROM aerial-prism-427113-h2.Target_sql_dataset.orders as o
Order by p_year, p_month)x
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers as c
USING(customer_id)
GROUP BY customer_state, p_month
ORDER BY customer_state, p_month

```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	p_month	no_of_orders			
1	AC	1	8			
2	AC	2	6			
3	AC	3	4			
4	AC	4	9			
5	AC	5	10			
6	AC	6	7			
7	AC	7	9			
8	AC	8	7			
9	AC	9	5			
10	AC	10	6			

INFERENCE: The data is for each states and and the month on month number of orders And it is seen that customer_state is equal to `SP` and has the highest month on month orders compared to other states.

2. How are the customers distributed across all the states?

```
SELECT customer_state, customer_count,
       sum(customer_count) OVER() as total_customer,
       Round((customer_count / sum(customer_count) OVER())*100,2) as
percent_of_cus
FROM (
  SELECT customer_state, COUNT(customer_id) as customer_count
  FROM aerial-prism-427113-h2.Target_sql_dataset.customers
  GROUP BY customer_state
  ORDER BY customer_count desc
)a
ORDER BY customer_count desc
```

Query results

[SAVE RESULTS](#) ▼

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state ▼	customer_count ▼	total_customer ▼	percent_of_cus ▼		
1	SP	41746	99441	41.98		
2	RJ	12852	99441	12.92		
3	MG	11635	99441	11.7		
4	RS	5466	99441	5.5		
5	PR	5045	99441	5.07		
6	SC	3637	99441	3.66		
7	BA	3380	99441	3.4		
8	DF	2140	99441	2.15		
9	ES	2033	99441	2.04		
10	GO	2020	99441	2.03		

INFERENCE: Out of 99441 customers 42% of customers are from the `SP` state and followed by RJ and MG contributes to 13% and 12% which around 69% of customers are from the three states and the rest are from other states.

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

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

You can use the "payment_value" column in the payments table to get the cost of orders.

```

SELECT month_val, percent_change
FROM(
SELECT *, lead(cost_of_order) OVER (partition by month_val order by year_val) as
lead_val,
ROUND(((lead(cost_of_order) OVER (order by month_val) -
cost_of_order)/cost_of_order)*100,2) As percent_change
FROM (
SELECT year_val,month_val, round(SUM(payment_value),2) as cost_of_order
FROM (
SELECT order_id,customer_id,order_purchase_timestamp,
EXTRACT(year from order_purchase_timestamp) year_val,
EXTRACT(month from order_purchase_timestamp) month_val,
payment_value
FROM aerial-prism-427113-h2.Target_sql_dataset.orders as o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.payments as p using(order_id)
WHERE EXTRACT(month from order_purchase_timestamp) IN (1,2,3,4,5,6,7,8)
AND EXTRACT(year from order_purchase_timestamp) IN (2017,2018)
)a
GROUP BY year_val,month_val
)b
ORDER BY month_val, year_val
)c
WHERE lead_val is not null

```

Query results

[SAVE RESULTS](#) ▾

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	month_val ▾	percent_change ▾				
1	1	705.13				
2	2	239.99				
3	3	157.78				
4	4	177.84				
5	5	94.63				
6	6	100.26				
7	7	80.04				
8	8	51.61				

INFERENCE: From the above table we can infer that the payment value has increased over the past year for all eight months and in the starting months the payment value change was significant.

2. Calculate the Total & Average value of order price for each state.

```
SELECT c.customer_state,ROUND(SUM(price),2) total_price,
ROUND(AVG(price),2) avg_price
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.order_items oi USING (order_id)
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers c USING (customer_id)
GROUP BY c.customer_state
ORDER BY total_price desc
```

Query results

[SAVE RESULTS](#) ▼

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state ▼	total_price ▼	avg_price ▼			
1	SP	5202955.05	109.65			
2	RJ	1824092.67	125.12			
3	MG	1585308.03	120.75			
4	RS	750304.02	120.34			
5	PR	683083.76	119.0			
6	SC	520553.34	124.65			
7	BA	511349.99	134.6			
8	DF	302603.94	125.77			
9	GO	294591.95	126.27			
10	ES	275037.31	121.91			

INFERENCE:From the above table we can conclude the total price that we collected is highest from the SP state but the overall the average price is not the highest compared to other states but the state BA it average price is highest.we can say somewhat that the state BA has more rich people if the available orders are same in both places.

3. Calculate the Total & Average value of order freight for each state.

```
SELECT customer_state,ROUND(SUM(freight_value),2) total_freight_cost,
ROUND(AVG(freight_value),2) avg_freight_cost
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.order_items oi USING (order_id)
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers c USING (customer_id)
GROUP BY customer_state
ORDER BY total_freight_cost desc
```

Query results

[SAVE RESULTS](#) ▾

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state ▾	total_freight_cost ▾	avg_freight_cost ▾			
1	SP	718723.07	15.15			
2	RJ	305589.31	20.96			
3	MG	270853.46	20.63			
4	RS	135522.74	21.74			
5	PR	117851.68	20.53			
6	BA	100156.68	26.36			
7	SC	89660.26	21.47			
8	PE	59449.66	32.92			
9	GO	53114.98	22.77			
10	DF	50625.5	21.04			

INFERENCE: As expected, the total freight cost will be highest for SP state but the average freight cost is lowest compared to others. We can conclude that in the SP state, orders which are having low freight charges are ordered the most compared to others.

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

1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Also, calculate the difference (in days) between the estimated & actual delivery date of an order.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- $\text{time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$

- $\text{diff_estimated_delivery} = \text{order_delivered_customer_date} - \text{order_estimated_delivery_date}$

```
SELECT order_id,
--order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_d
ate,
date_diff(order_delivered_customer_date , order_purchase_timestamp, day) AS
time_to_deliver,
date_diff(order_estimated_delivery_date, order_delivered_customer_date , day) AS
diff_estimated_delivery,
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
WHERE order_delivered_customer_date is not null
ORDER BY time_to_deliver
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION	RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	order_id	time_to_deliver	diff_estimated_deliver		
1	e65f1eeee1f52024ad1dcd034...	0	9		
2	bb5a519e352b45b714192a02f...	0	25		
3	434cecee7d1a65fc65358a632...	0	19		
4	d3ca7b82c922817b06e5ca211...	0	11		
5	1d893dd7ca5f77ebf5f59f0d20...	0	10		
6	d5fbedc85190ba88580d6f82...	0	7		
7	79e324907160caea526fd8b94...	0	8		
8	38c1e3d4ed6a13cd0cf612d4c...	0	16		
9	8339b608be0d84fca9d8da68b...	0	27		
10	f349cdb62f69c3fae5c4d7d3f3...	0	12		

INFERENCE: In the above table, 0 represents orders delivered within 24 hours, which is good from the customer's perspective. However, the data shows multiple orders delayed beyond their estimated delivery time, which is a concern. We need to conduct a root cause analysis and ensure timely deliveries for customers.

2. Find out the top 5 states with the highest & lowest average freight value.

```

SELECT *
FROM
(SELECT customer_state, ROUND(AVG(freight_value),2) avg_freight_cost
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.order_items oi USING (order_id)
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers c USING (customer_id)
GROUP BY customer_state
ORDER BY avg_freight_cost desc
LIMIT 5)a
UNION ALL
SELECT *
FROM (
SELECT customer_state, ROUND(AVG(freight_value),2) avg_freight_cost
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.order_items oi USING (order_id)
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers c USING (customer_id)
GROUP BY customer_state
ORDER BY avg_freight_cost asc
LIMIT 5)b

```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	avg_freight_cost				
1	RR	42.98				
2	PB	42.72				
3	RO	41.07				
4	AC	40.07				
5	PI	39.15				
6	SP	15.15				
7	PR	20.53				
8	MG	20.63				
9	RJ	20.96				
10	DF	21.04				

INFERENCE: Highest freight cost is for RR states and lowest is for SP state as per the given data.

3. Find out the top 5 states with the highest & lowest average delivery time.

```

SELECT * FROM
(SELECT customer_state, "lowest_delivery_time" as delivery_time,
--order_purchase_timestamp, order_estimated_delivery_date, order_delivered_customer_d
ate,
-- date_diff(order_delivered_customer_date , order_purchase_timestamp, day) AS
time_to_deliver_in_days,
date_diff(order_delivered_customer_date , order_purchase_timestamp, minute) AS
time_to_deliver_in_min,
-- date_diff(order_estimated_delivery_date, order_delivered_customer_date , day)
AS diff_estimated_delivery,
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers USING(customer_id)
WHERE order_delivered_customer_date is not null
ORDER BY time_to_deliver_in_min
LIMIT 5)a
UNION ALL
SELECT * FROM
(SELECT customer_state, "highest_delivery_time" as delivery_time,
--order_purchase_timestamp, order_estimated_delivery_date, order_delivered_customer_d
ate,
-- date_diff(order_delivered_customer_date , order_purchase_timestamp, day) AS
time_to_deliver_in_days,
date_diff(order_delivered_customer_date , order_purchase_timestamp, minute) AS
time_to_deliver_in_min,
-- date_diff(order_estimated_delivery_date, order_delivered_customer_date , day)
AS diff_estimated_delivery,
FROM aerial-prism-427113-h2.Target_sql_dataset.orders

```



```

LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers USING(customer_id)
WHERE order_delivered_customer_date is not null
ORDER BY time_to_deliver_in_min desc
LIMIT 5)b

```

Row	customer_state	delivery_time	time_to_deliver_in_m
1	ES	highest_delivery_time	301865
2	RJ	highest_delivery_time	300026
3	PA	highest_delivery_time	281712
4	PI	highest_delivery_time	280584
5	SE	highest_delivery_time	280272
6	RJ	lowest_delivery_time	768
7	SP	lowest_delivery_time	1125
8	SP	lowest_delivery_time	1231
9	BA	lowest_delivery_time	1243
10	SP	lowest_delivery_time	1282

INFERENCE:The ES has the highest delivery delivery time and the RJ state has the lowest delivery time.

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```

SELECT customer_state FROM (
SELECT customer_state,
--
order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_date,
-- date_diff(order_delivered_customer_date , order_purchase_timestamp, day) AS
time_to_deliver_in_days,
-- date_diff(order_delivered_customer_date , order_purchase_timestamp, minute) AS
time_to_deliver_in_min,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) AS
diff_estimated_delivery,
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.customers USING(customer_id)
WHERE order_delivered_customer_date is not null
)a
ORDER BY diff_estimated_delivery
LIMIT 5

```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state					
1	SP					
2	MA					
3	RS					
4	SP					
5	RJ					

INFERENCE: SP, MA, RS, SP, RJ are the top 5 states where the order delivery is really fast as compared to the estimated date of delivery

6. Analysis based on the payments:

1. Find the month on month no. of orders placed using different payment types.

```
SELECT payment_type, purchase_month, count(*) AS order_paymt_count
FROM (
SELECT payment_type,
EXTRACT(month FROM order_purchase_timestamp) AS purchase_month,
order_id
FROM aerial-prism-427113-h2.Target_sql_dataset.orders
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.payments USING (order_id)
)a
WHERE payment_type IS NOT NULL
GROUP BY payment_type, purchase_month
ORDER BY payment_type, purchase_month
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	payment_type	purchase_month	order_paymt_count			
1	UPI	1	1715			
2	UPI	2	1723			
3	UPI	3	1942			
4	UPI	4	1783			
5	UPI	5	2035			
6	UPI	6	1807			
7	UPI	7	2074			
8	UPI	8	2077			
9	UPI	9	903			
10	UPI	10	1056			

INFERENCE: Maximum payment was made by credit card and it is around 74% of total payments and followed by UPI.

2. Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT payment_installments, COUNT(o.order_id) as order_count
FROM aerial-prism-427113-h2.Target_sql_dataset.orders o
LEFT JOIN aerial-prism-427113-h2.Target_sql_dataset.payments p USING (order_id)
WHERE order_status != "canceled" and payment_installments IS NOT NULL
GROUP BY payment_installments
ORDER BY order_count desc
```

Query results

[SAVE RESULTS](#)

JOB INFORMATION	RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	payment_installment	order_count			
1	1	52184			
2	2	12353			
3	3	10392			
4	4	7056			
5	10	5292			
6	5	5209			
7	8	4239			
8	6	3898			
9	7	1620			
10	9	638			

INFERENCE: By analysing the data majority of orders have one payment instalment and the highest instalments are 24.

7.Actionable Insights & Recommendations (10 points)

1. The given data shows that states SP has the maximum number of orders and and other states have less number of orders compared to SP, targeting more on other states can result in increase in orders and revenue.
2. As per purchase time stamp the available data is from 04/09/2016 to 17/10/2018. In 2017 we have 45101 orders and in 2018 we have beat the target and and reached to 54011 which approx 20% greater then the previous year and year is not completed yet so its show orders are increasing year on year.
3. Enhancing delivery times in regions with longer shipping durations can significantly boost customer satisfaction and drive repeat purchases. Efficient logistics and streamlined shipping processes are essential for achieving this
4. Given that states like (SP) and (RJ) already have high order counts, businesses should focus on customer retention strategies to further boost sales and foster brand loyalty. This can be achieved through personalised marketing campaigns, loyalty programs, and exceptional customer service experiences

5. Recommendation is to enhance logistics and shipping processes to shorten delivery times and boost customer satisfaction.
 6. Evaluate pricing and freight fees to ensure market competitiveness while maximising revenue and profitability. Consider adjusting prices or freight fees as appropriate.
 7. Monitor competitor activity and adjust the business strategy accordingly. This could include matching or offering better pricing, expanding product offerings, or improving customer service to stay competitive in the market.
-

Evaluation Criteria (100 points):

1. Initial exploration like checking the structure & characteristics of the data (15 points)
 2. In-depth Exploration (15 points)
 3. Evolution of E-commerce orders in the Brazil region (10 points)
 4. Impact on Economy (20 points)
 5. Analysis on sales, freight and delivery time (20 points)
 6. Analysis based on the payments (10 points)
 7. Actionable Insights & Recommendations (10 points)
-

Submission Process <IMP>:

Once you're done with the case study...

- Use a Word document to paste your SQL queries along with a screenshot of the first 10 rows from the output.
 - List down any valuable insights that you find during the analysis and provide some action items from the company's perspective in order to improve the current situation.
 - Convert your solutions doc into a PDF, and upload the same on the platform.
 - Please note that after submitting once, you will not be allowed to edit your submission.
-

General Guidelines:

- Evaluation will be kept lenient, so make sure you attempt this case study.
- Try to attempt this before it is discussed in the Live Case Discussion with the Instructor.
- It is understandable that you might struggle with getting started on this or feel stuck at some point.
In such case:

- Read the question carefully and try to understand what exactly is being asked.
- Brainstorm a little. If you're getting an error, remember that Google is your best friend.
- You can watch the lecture recordings or go through your lecture notes once again if you feel like you're getting confused over some specific topics.
- Discuss your problems with your peers. Make use of the Slack channel and WhatsApp group.
- Only if you think that there's a major issue, you can reach out to your Instructor via Slack or Email.