Brief Summary:

Target is one of the world's most recognized brands and one of America's leading retailers. 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 business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allows viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

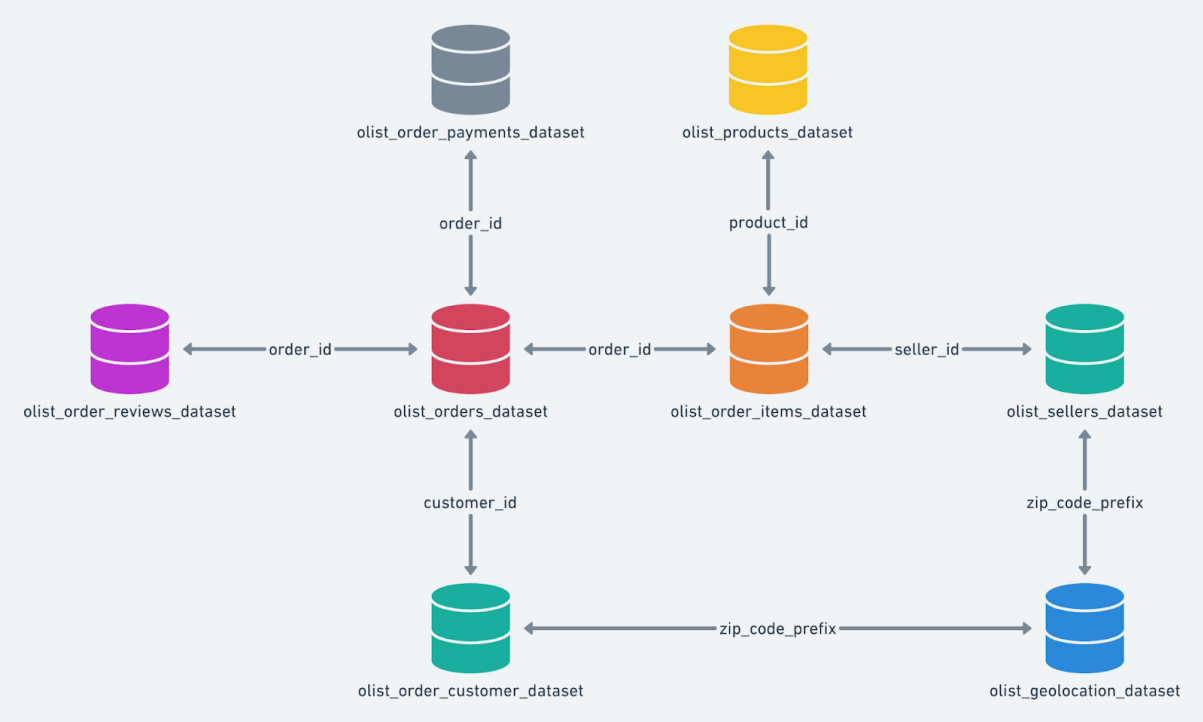
Dataset:

https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

Data is available in 8 csv files:

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

High level overview of relationship between datasets:



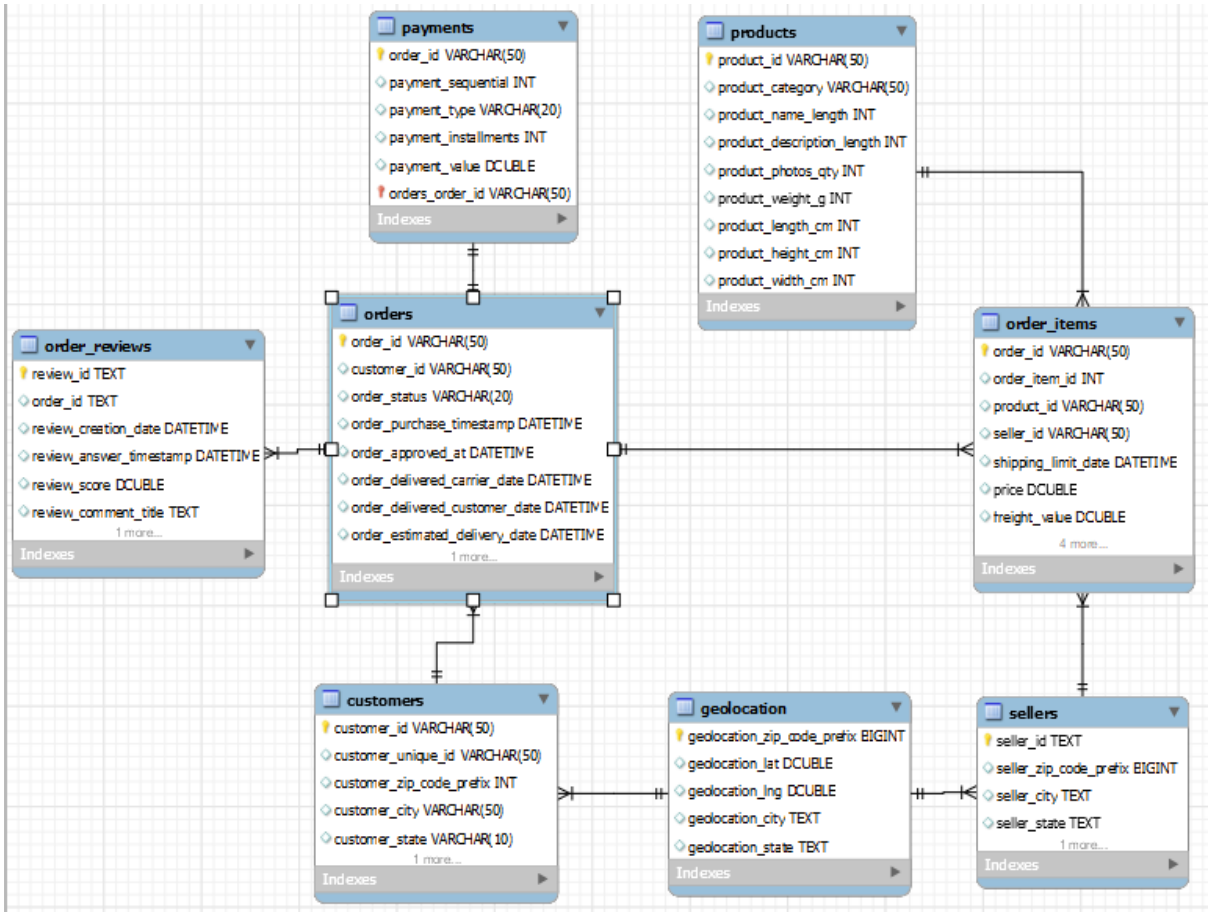
Problem Statement:

Assume you are a data scientist at Target, and are given this data to Analyze and provide some Insights and Recommendations from it.

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
    2. Time period for which the data is given
    3. Cities and States covered in the dataset
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?
    2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?
3. Evolution of E-commerce orders in the Brazil region:
    1. Get month on month orders by region, states
    2. How are customers distributed in Brazil
4. Impact on Economy: Analyze the money movemented 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)
    2. Mean & Sum of price and freight value by customer state
5. Analysis on sales, freight and delivery time
    1. Calculate days between purchasing, delivering and estimated delivery
    2. Create columns:
        - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
        - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
    3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
    4. Sort the data to get the following:
        1. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5
        2. Top 5 states with highest/lowest average time to delivery
        3. Top 5 states where delivery is really fast/ not so fast compared to estimated date
6. Payment type analysis:
    1. Month over Month count of orders for different payment types
    2. Distribution of payment installments and count of orders

E-R Diagram:

It is showing the primary key in each table, data type of entities and the relation between the tables.



Q1.1:

Data type of all columns in a table?

A1.1:

```
SELECT
  table_catalog,
  table_schema,
  table_name,
  table_type
FROM Target.INFORMATION_SCHEMA.TABLES;
```

Results:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_catalog	table_schema	table_name	table_type
1	targetsql7	Target	Sellers	BASE TABLE
2	targetsql7	Target	order_items	BASE TABLE
3	targetsql7	Target	Orders	BASE TABLE
4	targetsql7	Target	Geolocation	BASE TABLE
5	targetsql7	Target	Payments	BASE TABLE
6	targetsql7	Target	Products	BASE TABLE
7	targetsql7	Target	Order_reviews	BASE TABLE
8	targetsql7	Target	Customers	BASE TABLE

```
SELECT
  table_name,
  column_name,
  ordinal_position,
  is_nullable,
  data_type
FROM Target.INFORMATION_SCHEMA.COLUMNS
ORDER BY table_name,ordinal_position;
```

Results:

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	table_name	column_name	ordinal_posi...	is_nullable	data_type
1	Customers	customer_id	1	YES	STRING
2	Customers	customer_unique_id	2	YES	STRING
3	Customers	customer_zip_code_prefix	3	YES	INT64
4	Customers	customer_city	4	YES	STRING
5	Customers	customer_state	5	YES	STRING
6	Geolocation	geolocation_zip_code_prefix	1	YES	INT64
7	Geolocation	geolocation_lat	2	YES	FLOAT64

Q1.2:

Time period for which the data is given?

A1.2:

```
SELECT
  Min (order_purchase_timestamp) AS Order_Start_Date,
  Max (order_purchase_timestamp) AS Order_End_Date
FROM `Target.Orders`
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

Q1.3:

What all cities and states are covered in the dataset?

A1.3:

```
SELECT
  DISTINCT
  custo.customer_zip_code_prefix,
  custo.customer_city,
  custo.customer_state
FROM `Target.Customers` AS custo ;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_zi...	customer_city	customer_state
1	59650	acu	RN
2	63430	ico	CE
3	95240	ipe	RS
4	62250	ipu	CE
5	89760	ita	SC
6	13312	itu	SP
7	13313	itu	SP

```
SELECT
```

```
  DISTINCT
  Seller_zip_code_prefix,
  seller_city,
  seller_state
FROM `Target.Sellers`;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Seller_zip_c...	seller_city	seller_state
1	69900	rio branco	AC
2	69005	manaus	AM
3	48602	bahia	BA
4	44600	ipira	BA
5	44900	irece	BA
6	45658	ilheus	BA
7	46430	guanambi	BA
8	40243	salvador	BA

```
SELECT
  DISTINCT
  geolocation_zip_code_prefix,
  geolocation_city,
  geolocation_state
FROM `Target.Geolocation`;
```

Query results SA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	geolocation...	geolocation_city	geolocation_state
1	49010	aracaju	SE
2	49047	aracaju	SE
3	49030	aracaju	SE
4	49048	aracaju	SE
5	49050	aracaju	SE
6	49015	aracaju	SE

```
SELECT COUNT (distinct customer_state) from `Target.Customers`;
SELECT DISTINCT customer_state from `Target.Customers`;
```

OUTPUT: 27 Distinct States are present in the data of ['SP' 'SC' 'MG' 'PR' 'RJ' 'RS' 'PA' 'GO' 'ES' 'BA' 'MA' 'MS' 'CE' 'DF' 'RN' 'PE' 'MT' 'AM' 'AP' 'AL' 'RO' 'PB' 'TO' 'PI' 'AC' 'SE' 'RR']

Inference:

- The dataset has records for a span of 773 days.
- The entire time period of this dataset is Sep 04, 2016 to Oct 17, 2018.
- Data contains 4119 Distinct cities.
- 27 Distinct States are present in the data of ['SP' 'SC' 'MG' 'PR' 'RJ' 'RS' 'PA' 'GO' 'ES' 'BA' 'MA' 'MS' 'CE' 'DF' 'RN' 'PE' 'MT' 'AM' 'AP' 'AL' 'RO' 'PB' 'TO' 'PI' 'AC' 'SE' 'RR']

Q2.1:

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?

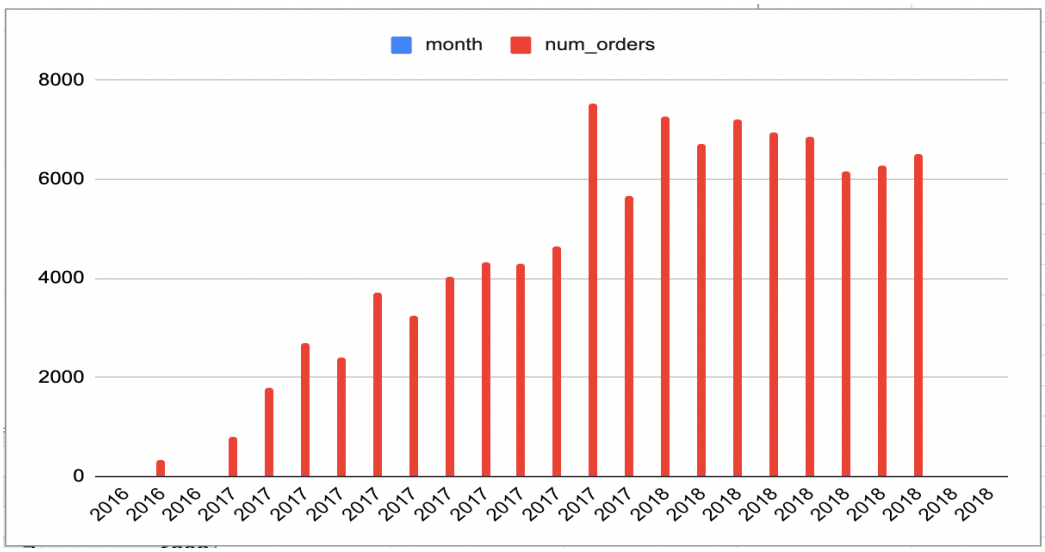
A2.1:

```
SELECT
EXTRACT(year FROM timestamp(order_purchase_timestamp)) AS year,
EXTRACT(month FROM timestamp(order_purchase_timestamp)) AS month,
COUNT(1) AS num_orders
FROM `Target.Orders` GROUP BY year, month ORDER BY year, month;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

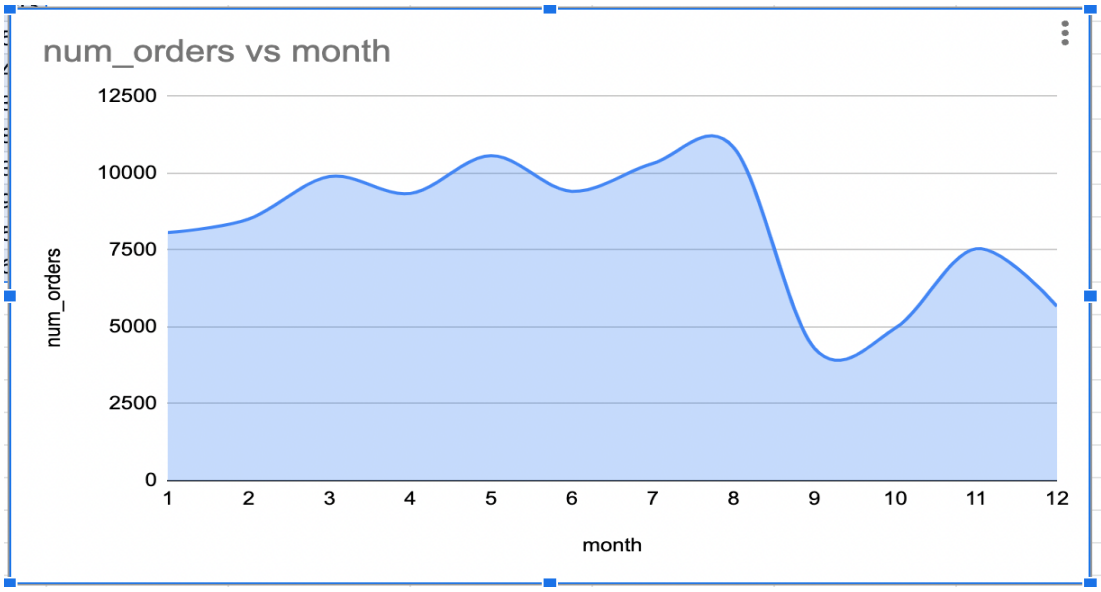
Row	year	month	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404



we can observe there was growing trend from Jan 2017 to November 2017.In 2018, it remained nearly constant. Orders peaked on November 2017.

Can we see some seasonality with peaks at specific months?

```
SELECT
EXTRACT(month FROM timestamp(order_purchase_timestamp)) AS month,
COUNT(1) AS num_orders
FROM `Target.Orders`
GROUP BY 1
ORDER BY 1;
```



Q2.2:

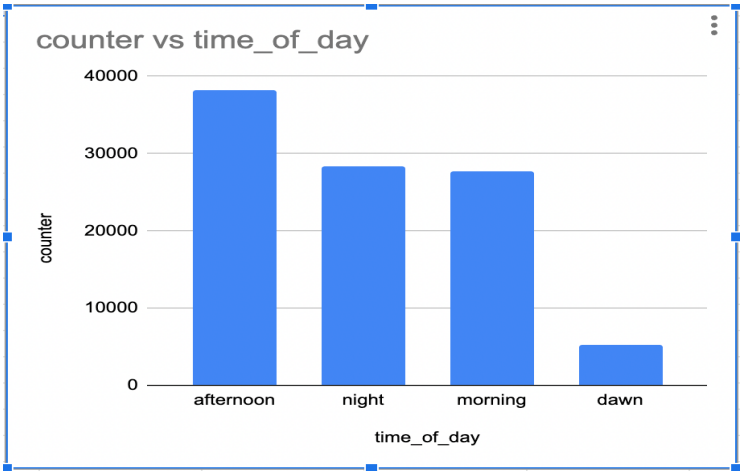
What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)

A2.2:

```
SELECT
CASE
  WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 0 AND 6
  THEN 'dawn'
  WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 7 AND 12
  THEN 'morning'
  WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 13 AND 18
  THEN 'afternoon'
  WHEN EXTRACT(hour FROM timestamp(order_purchase_timestamp)) BETWEEN 19 AND 23
  THEN 'night'
END AS time_of_day, COUNT(DISTINCT order_id) AS counter
FROM `Target.Orders` GROUP BY 1
ORDER BY 2 DESC;
```

JOB INFORMATION | RESULTS | JSON | EXI

Row	time_of_day	counter
1	afternoon	38135
2	night	28331
3	morning	27733
4	dawn	5242



Q3: Evolution of E-commerce orders in the Brazil region:

Q3.1

Get month on month orders by region, states:

```
SELECT
EXTRACT(month FROM timestamp(order_purchase_timestamp)) AS month,
g.geolocation_state,
COUNT(1) AS num_orders
FROM `Target.Orders` As o
INNER JOIN `Target.Customers` As c
ON o.customer_id = c.customer_id
INNER JOIN `Target.Geolocation` As g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state, month
ORDER BY geolocation_state DESC, month ASC;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	month	geolocation_state	num_orders
1	1	TO	1544
2	2	TO	1287
3	3	TO	1626
4	4	TO	2379
5	5	TO	2691
6	6	TO	1577
7	7	TO	743
8	8	TO	1603
9	9	TO	1236

Q3.2:

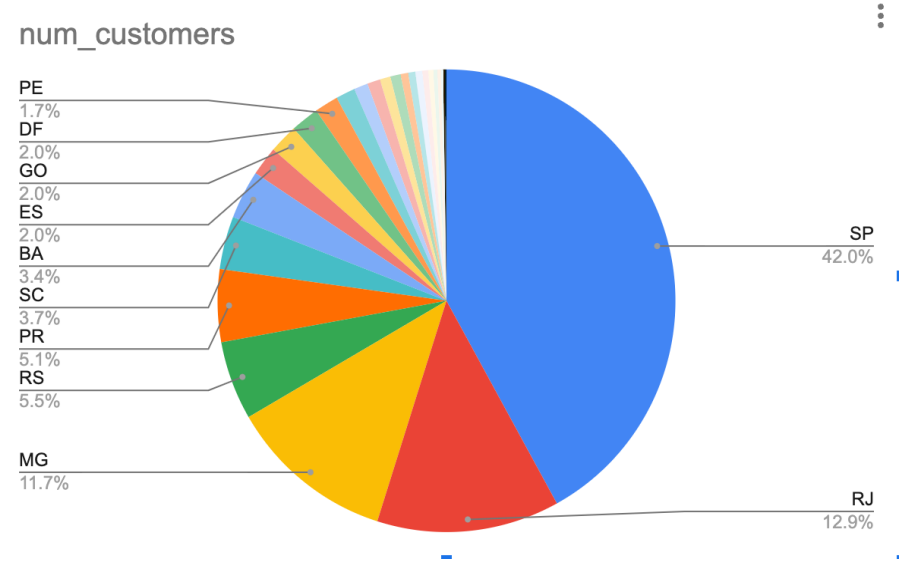
How are customers distributed in Brazil.

A3.2:

```
SELECT
g.geolocation_state,
COUNT(DISTINCT (c.customer_unique_id)) AS num_customers
FROM `Target.Customers` As c
INNER JOIN `Target.Geolocation` As g
ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
GROUP BY g.geolocation_state
ORDER BY num_customers DESC;
```

JOB INFORMATION | RESULTS | JSON | E>

Row	geolocation_state	num_custo...
1	SP	40287
2	RJ	12372
3	MG	11248
4	RS	5284
5	PR	4871
6	SC	3547
7	BA	3268
8	ES	1959
9	GO	1944
10	DF	1913
11	PE	1605
12	CE	1310



Q4:

Impact on Economy:

Analyze the money movemented by e-commerce by looking at order prices, freight and others.

Q4.1:

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

A4.1: Group the data on yearly and monthly level

```
WITH cte_table AS
(
SELECT
EXTRACT(month FROM timestamp(o.order_purchase_timestamp)) AS month,
EXTRACT(year FROM timestamp(o.order_purchase_timestamp)) AS year,
(sum(price) / COUNT( distinct o.order_id)) AS price_per_order,
(sum(freight_value) / COUNT(distinct o.order_id)) AS freight_per_order
FROM `Target.Orders` As o
INNER JOIN `Target.order_items` As i
ON o.order_id = i.order_id
GROUP BY year, month
)
SELECT (price_per_order), (freight_per_order), month, year
FROM cte_table
order by year asc, month asc ;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DE

Row	price_per_or...	freight_per_...	month	year
1	89.12	29.13	9	2016
2	160.739155...	23.7051298...	10	2016
3	10.9	8.72	12	2016
4	152.487794...	21.3886185...	1	2017
5	142.702261...	22.4914021...	2	2017
6	141.743392...	21.8494093...	3	2017
7	150.534182...	21.9552530...	4	2017
8	138.270803...	21.8906584...	5	2017
9	134.609449...	21.7359154...	6	2017
10	125.480342...	21.9047971...	7	2017

Compare YoY at a monthly level

```
WITH
cte_table AS
(
SELECT
EXTRACT(month FROM timestamp(order_purchase_timestamp)) AS month,
EXTRACT(year FROM timestamp(order_purchase_timestamp)) AS year,
sum(price) AS total_price,
sum(freight_value) AS total_freight
FROM `Target.Orders` As o
INNER JOIN `Target.order_items` i
ON o.order_id = i.order_id
GROUP BY year, month
ORDER BY year ASC, month ASC
)
SELECT
month,
price_2017,
price_2018,
round((price_2018 - price_2017) / price_2017 * 100, 2) AS yoy
FROM
(
SELECT
month,
sum(CASE WHEN year = 2017 THEN total_price ELSE 0 END) AS price_2017,
sum(CASE WHEN year = 2018 THEN total_price ELSE 0 END) AS price_2018
FROM cte_table
WHERE (year = 2017 OR year = 2018) AND month BETWEEN 1 AND 8 GROUP BY month
order by month
);
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	month	price_2017	price_2018	yoy
1	1	120312.86999999965	950030.36000003689	689.63
2	2	247303.01999999525	844178.71000003361	241.35
3	3	374344.30000000092	983213.440000031	162.65
4	4	359927.23000000004	996647.75000003015	176.9
5	5	506071.140000009	996517.68000002869	96.91
6	6	433038.60000000545	865124.31000002357	99.78
7	7	498031.48000000924	895507.22000002151	79.81
8	8	573971.68000001507	854686.330000025	48.91

Comparison Between Year 2017 & 2018:

```
WITH DATAST AS (
SELECT
Format_Date ('%Y', order_purchase_timestamp) AS Order_Puchase_Year,
Sum (price+freight_value) AS Cost_Orders
FROM `Target.Orders` AS ord
JOIN `Target.order_items` AS ord_it
ON ord.order_id = ord_it.order_id
WHERE (
Cast ( Format_Date ('%Y%m', order_purchase_timestamp) AS NUMERIC) >= 201701 AND
Cast ( Format_Date ('%Y%m', order_purchase_timestamp) AS NUMERIC) <= 201708 ) OR (
Cast ( Format_Date ('%Y%m', order_purchase_timestamp) AS NUMERIC) >= 201801 AND
Cast ( Format_Date ('%Y%m', order_purchase_timestamp) AS NUMERIC) <= 201808 )
GROUP BY Order_Puchase_Year
ORDER BY Order_Puchase_Year
)
SELECT
(D2.Cost_Orders - D1.Cost_Orders)/(D1.Cost_Orders) * 100 AS Perc_Increase_Cost
FROM DATAST D1
JOIN DATAST D2
ON D1.Order_Puchase_Year = '2017'
AND D2.Order_Puchase_Year = '2018';
```

JOB INFORMATION | RESULTS | JSON

Row	Perc_Increase_Cost
1	139.41507922890739

Q4.2

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

```
with cte_table as
(
select
c.customer_state as state,
sum(price) as total_price,
count(distinct(o.order_id)) as num_orders
from `Target.Orders` As o
inner join `Target.order_items` As i
on o.order_id= i.order_id
inner join `Target.Customers` c
on o.customer_id=c.customer_id
group by state
```

```
)
select
state, total_price, num_orders,
(total_price/num_orders) as avg_price
from cte_table
order by total_price desc;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	state	total_price	num_orders	avg_price
1	SP	5202955.05...	41375	125.751179...
2	RJ	1824092.66...	12762	142.931567...
3	MG	1585308.02...	11544	137.327445...
4	RS	750304.020...	5432	138.126660...
5	PR	683083.760...	4998	136.671420...
6	SC	520553.340...	3612	144.117757...
7	BA	511349.990...	3358	152.278138...
8	DF	302603.939...	2125	142.401854...

Q5: Analysis on sales, freight and delivery time:

Q5.1:

Calculate days between purchasing, delivering and estimated delivery:

```
SELECT
order_id,
customer_id,
order_status,
order_purchase_timestamp,
order_delivered_carrier_date,
order_delivered_customer_date,
CASE
WHEN order_status = 'delivered'
THEN Date_Diff
(
order_estimated_delivery_date, order_delivered_customer_date, DAY
)
ELSE NULL
END AS diff_estimated_delivery,
CASE
WHEN order_status = 'delivered'
```

```
THEN Date_Diff (
order_delivered_customer_date, order_purchase_timestamp, DAY )
ELSE NULL
END AS time_to_delivery
FROM `Target.Orders` AS ord;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	customer_id	order_status	order_purchase_timestamp
1	635c894d068ac37e6e03dc54e...	7a34a8e890765ad6f90db76d0...	delivered	2017-04-15 15:37:38 UTC
2	3b97562c3aee8bdedcb5c2e45...	065d53860347d845788e041c...	delivered	2017-04-14 22:21:54 UTC
3	68f47f50f04c4cb6774570cfde...	0378e1381c730d4504ebc07d2...	delivered	2017-04-16 14:56:13 UTC
4	276e9ec344d3bf029ff83a161c...	d33e520a99eb4cfc0d3ef2b6ff...	delivered	2017-04-08 21:20:24 UTC
5	54e1a3c2b97fb0809da548a59...	a0bc11375dd3d8bdd0e0bfcbc...	delivered	2017-04-11 19:49:45 UTC
6	fd04fa4105ee8045f6a0139ca5...	8fe0db7abbccaf2d788689e91...	delivered	2017-04-12 12:17:08 UTC
7	302bb8109d097a9fc6e9cefc5...	22c0028cdec95ad1808c1fd50...	delivered	2017-04-19 22:52:59 UTC
8	66057d37308e787052a32828...	dca924c5e55e17bdba2ad42ae...	delivered	2017-04-15 19:22:06 UTC
9	19135c945c554eebfd7576c73...	1c7a9b908094192a2dfae2819...	delivered	2017-07-11 14:09:37 UTC
10	4493e45e7ca1084efcd38ddeb...	a1fa003a1a17fc47164251e0e...	delivered	2017-07-11 20:56:34 UTC

Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery:

```
WITH datast AS (
SELECT
ord.order_id,
ord.customer_id,
cust.customer_state,
ord.order_status,
ord.order_purchase_timestamp,
ord.order_delivered_carrier_date,
ord.order_delivered_customer_date,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_estimated_delivery_date,
ord.order_delivered_customer_date, DAY )
ELSE NULL
END AS diff_estimated_delivery,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_delivered_customer_date, ord.order_purchase_timestamp,DAY)
ELSE NULL
END AS time_to_delivery,
ord_it.freight_value
FROM `Target.Orders` ord
JOIN `Target.order_items` ord_it
```

```
ON ord.order_id = ord_it.order_id
JOIN `Target.Customers` cust
ON ord.customer_id = cust.customer_id
)
SELECT
customer_state,
Avg (freight_value) AS Avg_Freight_Value,
Avg (diff_estimated_delivery) AS Avg_diff_estimated_delivery,
Avg (time_to_delivery) AS Avg_time_to_delivery
FROM datast
GROUP BY customer_state ;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Avg_Freight_Value	Avg_diff_estimated_de...	Avg_time_to_delivery
1	MT	28.1662843601896	13.639344262295094	17.508196721311482
2	MA	38.25700242718446	9.1099999999999923	21.203750000000017
3	AL	35.843671171171152	7.9765807962529349	23.992974238875881
4	SP	15.147275390419248	10.264141599018073	8.2596627979587751
5	MG	20.630166806306541	12.399039950449046	11.514091049860689
6	PE	32.917862679955796	12.552119129438733	17.792096219931292
7	RJ	20.96092393168248	11.139645054090357	14.6888213250371
8	DF	21.041354945968383	11.274734607218704	12.501486199575384
9	RS	21.735804330392945	13.203000163052323	14.708299364095817
10	SE	36.653168831168855	9.1653333333333276	20.978666666666651
11	PR	20.531651567944248	12.533899805275263	11.480793060718735
12	PA	35.832685185185177	13.37476280834913	23.301707779886126
13	BA	26.363958936562248	10.119467825142538	18.774640238935675
14	CE	32.714201623815995	10.256661991584851	20.537166900420793

Sort the data to get the following:

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

```
WITH datast AS (
SELECT
ord.order_id,
ord.customer_id,
cust.customer_state,
ord.order_status,
ord.order_purchase_timestamp,
ord.order_delivered_carrier_date,
ord.order_delivered_customer_date,
CASE
```

```
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_estimated_delivery_date, ord.order_delivered_customer_date, DAY)
ELSE NULL
END AS diff_estimated_delivery,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_delivered_customer_date, ord.order_purchase_timestamp, DAY)
ELSE NULL
END AS time_to_delivery,
ord_it.freight_value
FROM `Target.Orders` AS ord
JOIN `Target.order_items` ord_it
ON ord.order_id = ord_it.order_id
JOIN `Target.Customers` cust
ON ord.customer_id = cust.customer_id
)
SELECT
customer_state,
Avg (freight_value) AS Avg_Freight_Value,
Avg (diff_estimated_delivery) AS Avg_diff_estimated_delivery,
Avg (time_to_delivery) AS Avg_time_to_delivery
FROM datast
GROUP BY customer_state
ORDER BY Avg_Freight_Value DESC
LIMIT 5 ;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Avg_Freight_Value	Avg_diff_estimated_delivery	Avg_time_to_delivery
1	RR	42.984423076923093	17.434782608695652	27.826086956521738
2	PB	42.723803986710941	12.15017064846416	20.119453924914676
3	RO	41.069712230215842	19.080586080586084	19.282051282051292
4	AC	40.073369565217405	20.010989010989018	20.329670329670336
5	PI	39.147970479704767	10.682600382409184	18.931166347992352

Q5.2:

Top 5 states with highest/lowest average time to delivery

```
WITH datast AS (
SELECT
ord.order_id,
ord.customer_id,
cust.customer_state,
ord.order_status,
ord.order_purchase_timestamp,
ord.order_delivered_carrier_date,
```

```
ord.order_delivered_customer_date,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_estimated_delivery_date,
ord.order_delivered_customer_date, DAY )
ELSE NULL
END AS diff_estimated_delivery,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_delivered_customer_date, ord.order_purchase_timestamp, DAY )
ELSE NULL
END AS time_to_delivery,
ord_it.freight_value
FROM `Target.Orders` ord
JOIN `Target.order_items` ord_it
ON ord.order_id = ord_it.order_id
JOIN `Target.Customers` cust
ON ord.customer_id = cust.customer_id
)
SELECT
customer_state,
Avg (freight_value) AS Avg_Freight_Value,
Avg (diff_estimated_delivery) AS Avg_diff_estimated_delivery,
Avg (time_to_delivery) AS Avg_time_to_delivery
FROM datast
GROUP BY customer_state
ORDER BY Avg_time_to_delivery DESC
LIMIT 5;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Avg_Freight_Value	Avg_diff_estimated_delivery	Avg_time_to_delivery
1	RR	42.984423076923093	17.434782608695652	27.826086956521738
2	AP	34.006097560975618	17.444444444444443	27.753086419753075
3	AM	33.205393939393936	18.975460122699381	25.963190184049076
4	AL	35.843671171171152	7.9765807962529349	23.992974238875881
5	PA	35.832685185185177	13.37476280834913	23.301707779886126

Q5.3: Top 5 states where delivery is really fast/ not so fast compared to estimated date

```
WITH datast AS (
SELECT
ord.order_id,
ord.customer_id,
cust.customer_state,
ord.order_status,
ord.order_purchase_timestamp,
ord.order_delivered_carrier_date,
ord.order_delivered_customer_date,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_estimated_delivery_date,
ord.order_delivered_customer_date, DAY )
ELSE NULL
END AS diff_estimated_delivery,
CASE
WHEN ord.order_status = 'delivered'
THEN Date_Diff (
ord.order_delivered_customer_date,
ord.order_purchase_timestamp, DAY )
ELSE NULL
END AS time_to_delivery,
Ord_it.freight_value
FROM `Target.Orders` ord
JOIN `Target.order_items` ord_it
ON ord.order_id = ord_it.order_id
JOIN `Target.Customers` cust
ON ord.customer_id = cust.customer_id)
SELECT
customer_state,
Avg (freight_value) AS Avg_Freight_Value,
Avg (diff_estimated_delivery) AS Avg_diff_estimated_delivery,
Avg (time_to_delivery) AS Avg_time_to_delivery
FROM datast
GROUP BY customer_state
ORDER BY Avg_diff_estimated_delivery DESC
LIMIT 5;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Avg_Freight...	Avg_diff_est...	Avg_time_to...
1	AC	40.0733695...	20.0109890...	20.3296703...
2	RO	41.0697122...	19.0805860...	19.2820512...
3	AM	33.2053939...	18.9754601...	25.9631901...
4	AP	34.0060975...	17.4444444...	27.7530864...
5	RR	42.9844230...	17.4347826...	27.8260869...

Q6: Payment type analysis:

Q6.1:

Month over Month count of orders for different payment types

```
WITH datast AS (
SELECT
DISTINCT
Format_Date ('%B', ord.order_purchase_timestamp) AS Order_Puchase_Month,
payment_type
FROM `Target.Payments` pymt
JOIN `Target.Orders` ord
ON pymt.order_id = ord.order_id
)
SELECT
Order_Puchase_Month,
Count (1) AS Cnt_Orders
FROM datast
GROUP BY Order_Puchase_Month
ORDER BY Cnt_Orders DESC;
```

JOB INFORMATION | RESULTS | JSON | EXE(

Row	Order_Puchase_Month	Cnt_Orders
1	September	5
2	August	5
3	May	4
4	April	4
5	January	4
6	October	4
7	June	4
8	March	4
9	February	4
10	November	4
11	July	4
12	December	4

```
WITH datast AS (
SELECT
DISTINCT
Format_Date ('%Y%m', ord.order_purchase_timestamp) AS Order_Puchase_YearMonth,
payment_type
FROM `Target.Payments` pymt
JOIN `Target.Orders` ord
ON pymt.order_id = ord.order_id
)
SELECT
Order_Puchase_YearMonth,
payment_type,
Count (1) AS Cnt_Orders_Paymt_Type
FROM datast
GROUP BY
```

```
Order_Puchase_YearMonth,
payment_type
ORDER BY Cnt_Orders_Paymt_Type DESC;
```

Q6.2::

Distribution of payment installments and count of orders:

```
WITH datast AS (
SELECT
DISTINCT
Format_Date ('%Y%m', ord.order_purchase_timestamp) AS
Order_Puchase_YearMonth,
Payment_installments
FROM `Target.Payments` AS pymt
JOIN `Target.Orders` AS ord
ON pymt.order_id = ord.order_id
)
SELECT
Order_Puchase_YearMonth,
payment_installments,
Count (1) AS Cnt_Orders_Paymt_Inslmt
FROM datast
GROUP BY
Order_Puchase_YearMonth,
payment_installments
ORDER BY payment_installments DESC;
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Order_Puchase_YearMonth	payment_in...	Cnt_Orders_...
1	201711	24	1
2	201712	24	1
3	201801	24	1
4	201806	24	1
5	201806	23	1
6	201711	22	1
7	201711	21	1

Insights :

1. The given data is between the dates from *2016-09-04 and 2018-10-17* i.e. for *2 Years 1 month and 13 days.*
2. There are total of *19015 unique zip code* in geolocation.
3. There are *4119 cities* from where the customers are.
4. There are total of *27 States* from where the customers have placed their order.
5. The max sales is for **sao paulo city**.
6. We can see that the trend for E-Commerce in *Brazil* is increasing from *Sept.-2016 to June-2018*. After then it decreases for 3 continuous months and then there is a sudden drop in September- 2018
   The peak months are: May-2017, November-2017, December-2017.
7. Brazilian customers tend to buy at **Evening** i.e. between 4:00 pm to 9:00pm
8. The state **SP** has the *most no. of orders* placed by customers except for the month of Sept. and Dec. 2016
9. Max. customers are from *sao paulo* which has *15540 customers.*
10. The sales increased rate is **at-least 50%** from the last year. The max percent change is in January and the minimum change percent is in August.
11. The max days for which the order get delivered is **210 days** which is late by **181 days** from the estimated delivery date.
12. No. of orders which are delivered late from their expected delivery date is *6535* i.e.**6.77%** orders are delivered late from their estimated delivery date.
13. There is only 1 order which is delivered on the same day within 15 hrs .

Recommendations:

1. Target should set up their warehouses in the states and city where the order deliver date is late from estimated delivery date so that delivery will be fast and it can directly impact on the customer review which significantly increases the customer satisfaction and brand value of the company.
2. Target should introduce offers in the month of Aug- Oct. 2018 since in this 3 month, the sales dropped drastically so in order to revive, introducing offers and attracting customers can be a good decision.
3. Target should maximize their promotion in the evening as the customers tend to order more in evening time i.e. from 4-9 pm which can help in increasing the sales.
4. Since the sales are more in Jan. to Aug. in decreasing order, the stock of the items should be made accordingly in order to minimize the loss in case of any defectiveness in the product due to various factors.
5. There are 6.77% of orders which are delivered late then their expected delivery date so it should be reduced in the order to increase the customer satisfaction.
6. There are orders which are placed in one day so the model should be made which helps to deliver it on the same day if the order is placed before a particular time of the day. This helps in increasing the customer satisfaction.

7. Target should open their warehouses in the state: AP, RR, AM, AL, PA as this state's avg. delivery time is maximum.
8. Customers uses credit card more often in comparison to other payment types so Target should collaborate with those credit card companies in order to launch offers which will attract more customers which is directly beneficial for the company revenue.
9. The products in the category of insurance and service should be improved as the customer review for the category is 2.5 .