A Retail Chain: SQL + Tableau Analysis

The business case is about a company that is one of the world's most recognized brands and one of America's leading retailers. It 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 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.

In this business case, we will analyse the data and provide some insights and recommendations from it.

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

Each feature or columns of different CSV files are described 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_prefixZip	Code of the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made(Ex- Sao 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 location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- Sao 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.
shipping_limit_date	The date before which shipping of the ordered product must be completed.
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 name
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.(Ex-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.

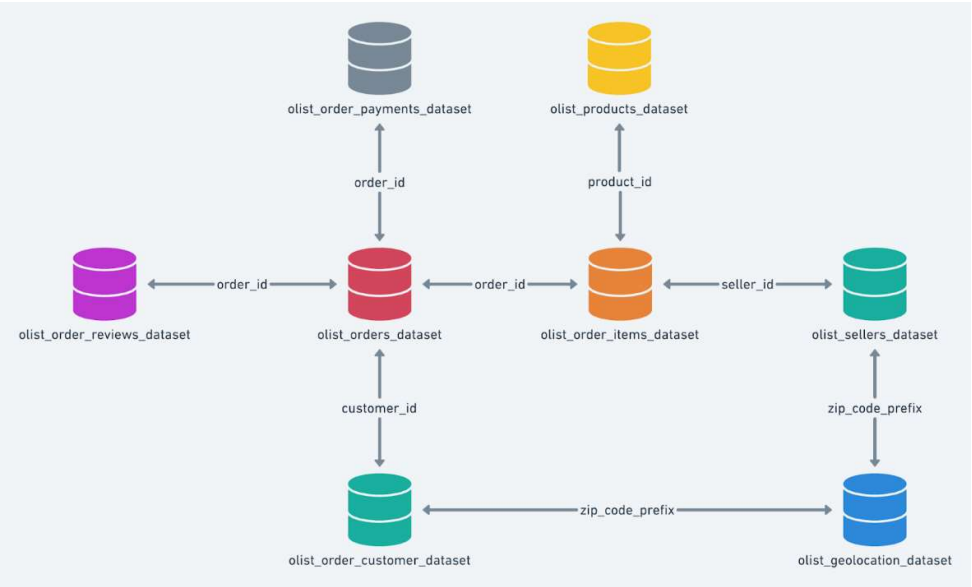
The **reviews.csv** contain following features:

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 the 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.

High level overview of relationship between datasets:



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

Solution:

SQL Query:

```
SELECT
  table_name,
  column_name,
  data_type
FROM
  `tgt-sql-20221006.Tgt_01.INFORMATION_SCHEMA.COLUMNS`
WHERE
  table_name='orders';
```

Output:

table_name	column_name	data_type
orders	order_id	STRING
orders	customer_id	STRING
orders	order_status	STRING
orders	order_purchase_timestamp	TIMESTAMP
orders	order_approved_at	TIMESTAMP
orders	order_delivered_carrier_date	TIMESTAMP
orders	order_delivered_customer_date	TIMESTAMP
orders	order_estimated_delivery_date	TIMESTAMP

Query Explanation:

All the fields present in different tables and their data type were explored. The above query shows example of 'Order' table fields and their data type.

Import the dataset and do usual exploratory analysis steps like checking the structure characteristics of the dataset

2. Time period for which the data is given

Solution:

SQL Query:

```
SELECT
  MIN(order_purchase_timestamp) AS Purchase_Time_from,
  greatest(MAX(order_delivered_customer_date), MAX(order_purchase_timestamp)) AS Purchase
_Time_to
FROM
  `tgt-sql-20221006.Tgt_01.orders`;
```

Output:

Purchase_Time_from	Purchase_Time_to
2016-09-04 21:15:19.000000 UTC	2018-10-17 17:30:18.000000 UTC

Query Explanation:

Data is available from 4th September 2016. It is the date when the first order was made.

Data is available till 17th October 2018. It is latest date between the last order delivered and the last purchase was made.

Import the dataset and do usual exploratory analysis steps like checking the structure characteristics of the dataset

3. Cities and States covered in the dataset

Solution:

SQL Query:

```
SELECT
  DISTINCT customer_city AS City,
  customer_state AS State
FROM
  `tgt-sql-20221006.Tgt_01.customers`
UNION DISTINCT
SELECT
  DISTINCT seller_city AS City,
  seller_state AS State
FROM
  `tgt-sql-20221006.Tgt_01.sellers`
ORDER BY
  2,
  1;
```

Sample Output:

City	State
brasileia	AC
cruzeiro do sul	AC
epitaciolandia	AC
manoel urbano	AC
porto acre	AC
rio branco	AC
senador guiomard	AC
xapuri	AC
agua branca	AL
anadia	AL

Query Explanation:

Data set includes data of sellers and customers from **4196 Cities** and **27 states** of **Brazil**.

In-depth Exploration:

1. e-commerce trend and seasonality in Brazil.

Solution:

a. Number of orders Monthly Trend

SQL Query:

```
SELECT
  EXTRACT(year FROM order_purchase_timestamp) AS year,
  EXTRACT(month FROM order_purchase_timestamp) AS month,
  COUNT(*) AS Number_of_Orders
FROM `tgt-sql-20221006.Tgt_01.orders`
WHERE order_purchase_timestamp between '2016-10-01' AND '2018-09-30'
GROUP BY 1, 2
ORDER BY 1, 2;
```

Output:

year	month	Number_of_Orders
2016	10	324
2016	12	1
2017	1	800
2017	2	1780
2017	3	2682
2017	4	2404
2017	5	3700
2017	6	3245
2017	7	4026
2017	8	4331
2017	9	4285
2017	10	4631
2017	11	7544
2017	12	5673
2018	1	7269
2018	2	6728
2018	3	7211
2018	4	6939
2018	5	6873
2018	6	6167
2018	7	6292
2018	8	6512
2018	9	16

Query Explanation:

Number of orders from October 2016 till September 2018 are analysed to understand the trend. Data of September 2016 and October 2018 is not considered for the analysis since data is not available to the entire month.

b. Number of orders Quarterly Trend

SQL Query:

```
SELECT
  EXTRACT(year FROM order_purchase_timestamp) AS year,
  CASE
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (1, 2, 3) THEN 'Q1'
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (4, 5, 6) THEN 'Q2'
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (7, 8, 9) THEN 'Q3'
  ELSE 'Q4'
END
  AS Quarter,
  COUNT(*) AS Number_of_Orders
FROM `tgt-sql-20221006.Tgt_01.orders`
WHERE order_purchase_timestamp>'2016-10-01'
  AND order_purchase_timestamp<'2018-09-30'
GROUP BY 1, 2
ORDER BY 1, 2;
```

Output:

year	Quarter	Number_of_Orders
2016	Q4	325
2017	Q1	5262
2017	Q2	9349
2017	Q3	12642
2017	Q4	17848
2018	Q1	21208
2018	Q2	19979
2018	Q3	12820

Query Explanation:

Number of orders for each quarter is analysed to understand the trend.

c. Sales and Profit Monthly Trend

SQL Query:

```
SELECT
  EXTRACT(year
  FROM order_purchase_timestamp) AS year,
  EXTRACT(month FROM order_purchase_timestamp) AS month,
  sum(payment_value) as Sales,
  sum(payment_value) - (sum(price)+sum(freight_value)) as Profit
FROM `tgt-sql-20221006.Tgt_01.orders` AS o
LEFT JOIN `tgt-sql-20221006.Tgt_01.order_items` AS oi
ON o.order_id=oi.order_id
LEFT JOIN `tgt-sql-20221006.Tgt_01.payments` AS p
ON o.order_id=p.order_id
where order_status='delivered'
  and order_purchase_timestamp between '2016-10-01' AND '2018-09-30'
GROUP BY 1, 2
ORDER BY 1, 2;
```

Output:

year	month	Sales	Profit
2016	10	61746.94	13613.64
2016	12	19.62	0
2017	1	176491.5	39742.07
2017	2	325782.7	40522.16
2017	3	505735.8	65671.18
2017	4	456108.3	42252.91
2017	5	701313.6	90760.58
2017	6	585401	71817.74
2017	7	716070	109167.3
2017	8	842689.9	167422.4
2017	9	996279.6	252754.9
2017	10	998609.6	213303.1
2017	11	1548683	353809
2017	12	1020067	145999.5
2018	1	1374064	253133.5
2018	2	1279970	274472.7
2018	3	1435458	266928.8
2018	4	1466607	294992.8
2018	5	1480668	309710.3
2018	6	1285926	221926.8
2018	7	1307228	243136.1
2018	8	1211240	191003.9

Query Explanation:

Total monthly sales and profit from October 2016 till September 2018 are analysed to understand the trend. Data of September 2016 and October 2018 is not considered for the analysis since data is not available to the entire month.

d. Sales and Profit Quarterly Trend

SQL Query:

```
SELECT
  EXTRACT(year FROM order_purchase_timestamp) AS year,
  CASE
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (1, 2, 3) THEN 'Q1'
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (4, 5, 6) THEN 'Q2'
    WHEN EXTRACT(month FROM order_purchase_timestamp) IN (7, 8, 9) THEN 'Q3'
    ELSE 'Q4'
  END AS Quarter,
  sum(payment_value) as Sales,
  sum(payment_value) - (sum(price)+sum(freight_value)) as Profit
FROM `tgt-sql-20221006.Tgt_01.orders` AS o
LEFT JOIN `tgt-sql-20221006.Tgt_01.order_items` AS oi
ON o.order_id=oi.order_id
LEFT JOIN `tgt-sql-20221006.Tgt_01.payments` AS p
ON o.order_id=p.order_id
where order_status='delivered'
  and order_purchase_timestamp between '2016-10-01' AND '2018-09-30'
GROUP BY 1, 2
ORDER BY 1, 2;
```

Output:

year	Quarter	Sales	Profit
2016	Q4	61766.56	13613.64
2017	Q1	1008010	145935.4
2017	Q2	1742823	204831.2
2017	Q3	2555040	529344.6
2017	Q4	3567360	713111.6
2018	Q1	4089493	794535
2018	Q2	4233201	826629.9
2018	Q3	2518468	434140

Query Explanation:

Total quarterly sales and profit is analysed to understand the trend.

In-depth Exploration:

2. Brazilian customers buying pattern by time of the day (Dawn, Morning, Afternoon or Night)

Solution:

SQL Query:

```
SELECT
  CASE
    WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN '(12AM-6AM) Dawn'
    WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN '(6AM-12PM) Morning'
    WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN '(12PM-6PM) Afternoon'
    ELSE '(6PM-12AM) Night'
  END AS time_of_day,
  COUNT(*) AS Number_of_Orders
FROM `tgt-sql-20221006.Tgt_01.orders`
WHERE order_purchase_timestamp>'2016-09-05'
  AND order_purchase_timestamp<'2018-10-16'
GROUP BY 1
ORDER BY 2 DESC;
```

Output:

time_of_day	Number_of_Orders
(12PM-6PM) Afternoon	38134
(6PM-12AM) Night	28329
(6AM-12PM) Morning	27733
(12AM-6AM) Dawn	5242

Query Explanation:

24 hours time is divided in 4 categories based on the hourly number of orders analysis. Orders of 4th September 2016 and 17 October 2018 are not considered for analysis since the data of entire day is not available.

Evolution of E-commerce orders in the Brazil region:

1. Get month on month orders by region, states

Solution:

SQL Query:

```
WITH states_region AS (
  SELECT
    geolocation_state AS states,
    AVG(geolocation_lat) AS lat_state,
    AVG(geolocation_lng) AS lng_state,
    CASE
      WHEN AVG(geolocation_lng) < -58 THEN 'West'
      WHEN AVG(geolocation_lat) < -18 THEN 'South'
      WHEN AVG(geolocation_lng) > -41 THEN 'East'
      WHEN AVG(geolocation_lat) > -7.5 THEN 'North'
    ELSE 'Center'
  END AS region
  FROM `tgt-sql-20221006.Tgt_01.geolocation` AS g
  GROUP BY 1),
  state_monthly_orders AS(
  SELECT
    region,
    states AS state,
    lat_state,
    lng_state,
    EXTRACT(year FROM order_purchase_timestamp) AS year,
    EXTRACT(month FROM order_purchase_timestamp) AS month,
    COUNT(order_id) AS order_count
  FROM `tgt-sql-20221006.Tgt_01.orders` AS o
  JOIN `tgt-sql-20221006.Tgt_01.customers` AS c
  ON o.customer_id=c.customer_id
  JOIN states_region AS sr
  ON c.customer_state=sr.states
  where order_purchase_timestamp between '2016-10-01' and '2018-09-30'
  GROUP BY 1, 2, 3, 4, 5, 6 )
SELECT
  cur_smo.region,
  cur_smo.state,
  cur_smo.lat_state,
  cur_smo.lng_state,
  cur_smo.year,
  cur_smo.month,
  cur_smo.order_count,
  prev_smo.order_count AS prev_month_count,
  ROUND((cur_smo.order_count-
prev_smo.order_count)*100.0/prev_smo.order_count, 2) AS orders_MoM_growth_prc
FROM state_monthly_orders AS cur_smo
LEFT JOIN state_monthly_orders AS prev_smo
ON cur_smo.state=prev_smo.state
  AND ((cur_smo.year=prev_smo.year
      AND cur_smo.month=prev_smo.month+1)
    OR (cur_smo.year=prev_smo.year+1
      AND cur_smo.month=1
      AND prev_smo.month=12))
ORDER BY 1, 2, 5, 6;
```

Sample Output:

region	state	lat_state	lng_state	year	month	order_count	prev_month_count	orders_MoM_growth_prc
Center	DF	-15.8108848	-47.9696305	2016	10	6		
Center	DF	-15.8108848	-47.9696305	2017	1	13		
Center	DF	-15.8108848	-47.9696305	2017	2	24	13	84.62
Center	DF	-15.8108848	-47.9696305	2017	3	57	24	137.5
Center	DF	-15.8108848	-47.9696305	2017	4	35	57	-38.6
Center	DF	-15.8108848	-47.9696305	2017	5	64	35	82.86
Center	DF	-15.8108848	-47.9696305	2017	6	70	64	9.38
Center	DF	-15.8108848	-47.9696305	2017	7	77	70	10
Center	DF	-15.8108848	-47.9696305	2017	8	87	77	12.99
Center	DF	-15.8108848	-47.9696305	2017	9	97	87	11.49

Query Explanation:

The states are divided into 5 regions based on their geographical locations. Inner join is preferred over Lag window function, since some months do not have any sales (i.e., missing months).

Evolution of E-commerce orders in the Brazil region:

2. How are customers distributed in Brazil

Solution:

SQL Query:

```
WITH
  states_region AS (
  SELECT
    geolocation_state AS states,
    AVG(geolocation_lat) AS lat_state,
    AVG(geolocation_lng) AS lng_state,
    CASE
      WHEN AVG(geolocation_lng) < -58 THEN 'West'
      WHEN AVG(geolocation_lat) < -18 THEN 'South'
      WHEN AVG(geolocation_lng) > -41 THEN 'East'
      WHEN AVG(geolocation_lat) > -7.5 THEN 'North'
      ELSE 'Center'
    END AS region
  FROM `tgt-sql-20221006.Tgt_01.geolocation` AS g
  GROUP BY 1)
SELECT
  region,
  states AS state,
  lat_state,
  lng_state,
  COUNT(customer_id) AS customer_count
FROM `tgt-sql-20221006.Tgt_01.customers` AS c
JOIN states_region AS sr
ON c.customer_state=sr.states
GROUP BY 1, 2, 3, 4
ORDER BY 1, 2;
```

Sample Output:

region	state	lat_state	lng_state	customer_count
Center	DF	-15.8108848	-47.9696305	2140
Center	GO	-16.5776446	-49.334195	2020
Center	MT	-14.1564817	-55.7089558	907
Center	TO	-9.50370037	-48.3486611	280
East	AL	-9.59972897	-36.0520168	413
East	BA	-13.0493609	-39.5606487	3380
East	CE	-4.36315123	-39.0041398	1336
East	PB	-7.08829791	-35.8216783	536
East	PE	-8.1790984	-35.7588656	1652
East	RN	-5.85670206	-35.9900786	485
East	SE	-10.8661994	-37.181169	350

Query Explanation:

The states are divided into 5 regions based on their geographical locations.

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)

Solution:

SQL Query:

```
with yearly_cost as(
select
  extract(year from order_purchase_timestamp) as year,
  sum(price) + sum(freight_value) as cost_of_orders
from `tgt-sql-20221006.Tgt_01.orders` AS o
join `tgt-sql-20221006.Tgt_01.order_items` AS oi
on o.order_id=oi.order_id
where extract(month from order_purchase_timestamp) between 1 and 8
and order_status in ('delivered', 'shipped')
group by 1)
select
  round((cur.cost_of_orders-
prev.cost_of_orders)*100.0/prev.cost_of_orders, 2) as increase_cost_prc
from yearly_cost as cur
join yearly_cost as prev on cur.year=2018 and prev.year=2017;
```

Output:

increase_cost_prc
142.79

Query Explanation:

Only the orders that are either delivered or shipped are considered for the cost of orders analysis.

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

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

Solution:

SQL Query:

```
select
  customer_state,
  sum(price) as total_price,
  avg(price) as mean_price,
  sum(freight_value) as total_freight_value,
  avg(freight_value) as mean_freight_value
from `tgt-sql-20221006.Tgt_01.orders` AS o
join `tgt-sql-20221006.Tgt_01.order_items` AS oi
on  o.order_id=oi.order_id
join `tgt-sql-20221006.Tgt_01.customers` AS c
on o.customer_id=c.customer_id
where order_status in ('delivered', 'shipped')
group by 1;
```

Sample Output:

customer_state	total_price	mean_price	total_freight_value	mean_freight_value
MT	154105.53	146.4881464	29519.87	28.06071293
MA	119233.82	145.7626161	31354.04	38.33012225
AL	79957.37	183.3884633	15660.62	35.91885321
SP	5108326.33	109.1732669	707599.33	15.12255199
MG	1559989.82	120.0823509	267830.37	20.61660919
PE	257175.36	143.9951624	58513.37	32.76224524
RJ	1799153.62	124.5175182	302426.93	20.9306478
DF	300140.04	125.7921375	50272.71	21.06987008
RS	735616.73	119.1668119	133686.21	21.65660295
SE	58898.95	153.3826823	14090.32	36.69354167
PR	671849.26	118.1585051	116389.9	20.46955681

Query Explanation:

Only the orders that are either delivered or shipped are considered for the cost of orders analysis.

Analysis on sales, freight and delivery time

1. Calculate days between purchasing, delivering and estimated delivery

Solution:

SQL Query:

```
select
  timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as purchas
ing_to_delivering,
  timestamp_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as purchas
ing_to_estimated_delivery,
  timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as de
livering_to_extimated_delivery
from `tgt-sql-20221006.Tgt_01.orders` as o
where order_status = 'delivered';
```

Sample Output:

purchasing_to_delivering	purchasing_to_estimated_delivery	delivering_to_extimated_delivery
23	33	9
12	7	-5
12	25	12
7	8	1
12	21	9
1	7	5
6	7	0
21	29	7
7	7	0
30	32	1
20	25	5

Query Explanation:

Above columns represent the actual delivery time to estimated delivery time and shows the how early or late the order is delivered compared to estimated delivery date.

Analysis on sales, freight and delivery time

2. Create columns:
    - time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
    - diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

Solution:

SQL Query:

```
select
  timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as purchasing_to_delivering,
  timestamp_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as purchasing_to_estimated_delivery,
  timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, day) as delivering_to_extimated_delivery,
  timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_delivery,
  timestamp_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery
from `tgt-sql-20221006.Tgt_01.orders` as o
where order_status = 'delivered';
```

Sample Output:

purchasing_to_delivering	purchasing_to_estimated_delivery	delivering_to_extimated_delivery	time_to_delivery	diff_estimated_delivery
23	33	9	23	-9
12	7	-5	12	5
12	25	12	12	-12
7	8	1	7	-1
12	21	9	12	-9
1	7	5	1	-5
6	7	0	6	0
21	29	7	21	-7
7	7	0	7	0
30	32	1	30	-1
20	25	5	20	-5

Query Explanation:

Created the table according to the explanation in the question. Since time to delivery can never be negative, I considered the second date (eg. order_delivered_customer_date) – first date (eg. order_purchase_timestamp) in the equation. Considering the same logic, the second equation i.e. diff_estimated_delivery is also calculated.

diff_estimated_delivery date represents the number of days the order is delivered early (negative value) or late (positive value) compared to estimated delivery date.

Analysis on sales, freight and delivery time

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

Solution:

a. From Sellers State to Customers State Analysis

SQL Query:

```
select
  seller_state as from_state,
  customer_state as to_state,
  avg(freight_value) as mean_freight_value,
  avg(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as mean_ti
me_to_delivery,
  avg(timestamp_diff(order_delivered_customer_date, order_estimated_delivery_date, day)) as me
an_diff_estimated_delivery
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.order_items` as oi
on o.order_id=o.order_id
join `tgt-sql-20221006.Tgt_01.customers` as c
on o.customer_id=c.customer_id
join `tgt-sql-20221006.Tgt_01.sellers` as s
on oi.seller_id=s.seller_id
where order_status = 'delivered'
group by 1, 2
order by 1, 2;
```

Sample Output:

from_state	to_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
AC	AC	32.84	20.6375	-19.7625
AC	AL	32.84	24.04030227	-7.947103275
AC	AM	32.84	25.9862069	-18.60689655
AC	AP	32.84	26.73134328	-18.73134328
AC	BA	32.84	18.86640049	-9.934889435
AC	CE	32.84	20.81782643	-9.957779515
AC	DF	32.84	12.50913462	-11.11875
AC	ES	32.84	15.33182957	-9.618546366
AC	GO	32.84	15.15074093	-11.26724578
AC	MA	32.84	21.11715481	-8.768479777
AC	MG	32.84	11.54218778	-12.29910164

Query Explanation:

Since the analysis is related to transportation of goods and time for delivery from seller to customer. The analysis considers relative analysis of cost and time required for delivery of orders “from state” to “to state”.

b. Customers State Analysis

SQL Query:

```
select
  customer_state as to_state,
  avg(freight_value) as mean_freight_value,
  avg(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as mean_ti
me_to_delivery,
  avg(timestamp_diff(order_delivered_customer_date, order_estimated_delivery_date, day)) as me
an_diff_estimated_delivery
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.order_items` as oi
on o.order_id=o.order_id
join `tgt-sql-20221006.Tgt_01.customers` as c
on o.customer_id=c.customer_id
where order_status = 'delivered'
group by 1
order by 1;
```

Sample Output:

to_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
AC	19.99031993	20.6375	-19.7625
AL	19.99031993	24.04030227	-7.947103275
AM	19.99031993	25.9862069	-18.60689655
AP	19.99031993	26.73134328	-18.73134328
BA	19.99031993	18.86640049	-9.934889435
CE	19.99031993	20.81782643	-9.957779515
DF	19.99031993	12.50913462	-11.11875
ES	19.99031993	15.33182957	-9.618546366
GO	19.99031993	15.15074093	-11.26724578
MA	19.99031993	21.11715481	-8.768479777
MG	19.99031993	11.54218778	-12.29910164

Query Explanation:

Just to understand the other perspective, the cost and time for delivery of orders are analysed with respect to only the customer state (i.e., to state).

Analysis on sales, freight and delivery time

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

Solution:

SQL Query:

```
select
  customer_state as to_state,
  avg(freight_value) as mean_freight_value,
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.order_items` as oi
on o.order_id=o.order_id
join `tgt-sql-20221006.Tgt_01.customers` as c
on o.customer_id=c.customer_id
where order_status = 'delivered'
group by 1
order by 2 desc
limit 5;
```

Output:

to_state	mean_freight_value
SP	19.99031993
MA	19.99031993
PI	19.99031993
RR	19.99031993
AP	19.99031993

Query Explanation:

Top 5 states having highest freight value are listed to identify the stats with highest delivery time so that we can understand the reasons, why more freight value is charged for specific states.

Analysis on sales, freight and delivery time

4. Sort the data to get the following:
    2) Top 5 states with highest/lowest average time to delivery

Solution:

SQL Query:

```
with statewise_mean_time_to_delivery as
(select
  customer_state as to_state,
  avg(timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as mean_ti
me_to_delivery,
  dense_rank() over(order by avg(timestamp_diff(order_delivered_customer_date, order_purchase_
timestamp, day)) desc) as rank
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.order_items` as oi
on o.order_id=o.order_id
join `tgt-sql-20221006.Tgt_01.customers` as c
on o.customer_id=c.customer_id
where order_status = 'delivered'
group by 1)
select
  to_state,
  mean_time_to_delivery
from statewise_mean_time_to_delivery
where rank <= 5
order by rank;
```

Output:

to_state	mean_time_to_delivery
RR	28.97560976
AP	26.73134328
AM	25.9862069
AL	24.04030227
PA	23.31606765

Query Explanation:

Top 5 states having highest average time to delivery are listed to identify the stats with highest delivery time so that we can understand the reasons, why more delivery time is required for specific states.

Analysis on sales, freight and delivery time

4. Sort the data to get the following:
    3) Top 5 states where delivery is really fast/ not so fast compared to estimated date

Solution:

SQL Query:

```
with statewise_mean_time_to_delivery as
(select
  customer_state as to_state,
  avg(timestamp_diff(order_estimated_delivery_date, order_purchase_timestamp, day)) as mean_ti
me_to_estimated_date,
  dense_rank() over(order by avg(timestamp_diff(order_estimated_delivery_date, order_purchase_
timestamp, day)) desc) as rank
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.order_items` as oi
on o.order_id=o.order_id
join `tgt-sql-20221006.Tgt_01.customers` as c
on o.customer_id=c.customer_id
where order_status = 'delivered'
group by 1)
select
  to_state,
  mean_time_to_estimated_date
from statewise_mean_time_to_delivery
where rank <= 5
order by rank;
```

Output:

to_state	mean_time_to_estimated_date
AP	45.86567164
RR	45.63414634
AM	44.92413793
AC	40.725
RO	38.38683128

Query Explanation:

Top 5 states having not so fast delivery time are listed to identify the stats with slowest delivery time so that we can understand the reasons, why more delivery time is required for specific states.

Payment type analysis:

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

Solution:

SQL Query:

```
with monthly_orders_by_payment_type as
(select
  payment_type,
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
  count(p.order_id) as orders_count
from `tgt-sql-20221006.Tgt_01.orders` as o
join `tgt-sql-20221006.Tgt_01.payments` as p
on o.order_id=p.order_id
group by 1, 2, 3)
select
  cur.payment_type,
  cur.year,
  cur.month,
  cur.orders_count,
  prev.orders_count as prev_month_orders_count,
  round((cur.orders_count-
prev.orders_count)*100.0/prev.orders_count, 2) as orders_MoM_growth_prc
from monthly_orders_by_payment_type as cur
left join monthly_orders_by_payment_type as prev
on cur.payment_type=prev.payment_type and ((cur.year=prev.year and cur.month=prev.month+1
) or
    (cur.year=prev.year+1 and cur.month=1 and prev.month=12))
order by 1, 2, 3;
```

Sample Output:

payment_type	year	month	orders_count	prev_month_orders_count	orders_MoM_growth_prc
UPI	2016	10	63		
UPI	2017	1	197		
UPI	2017	2	398	197	102.03
UPI	2017	3	590	398	48.24
UPI	2017	4	496	590	-15.93
UPI	2017	5	772	496	55.65
UPI	2017	6	707	772	-8.42
UPI	2017	7	845	707	19.52
UPI	2017	8	938	845	11.01
UPI	2017	9	903	938	-3.73
UPI	2017	10	993	903	9.97

Query Explanation:

Inner join is preferred over Lag window function, since some months do not have any sales (i.e., missing months).

Payment type analysis:

2. Distribution of payment instalments and count of orders

Solution:

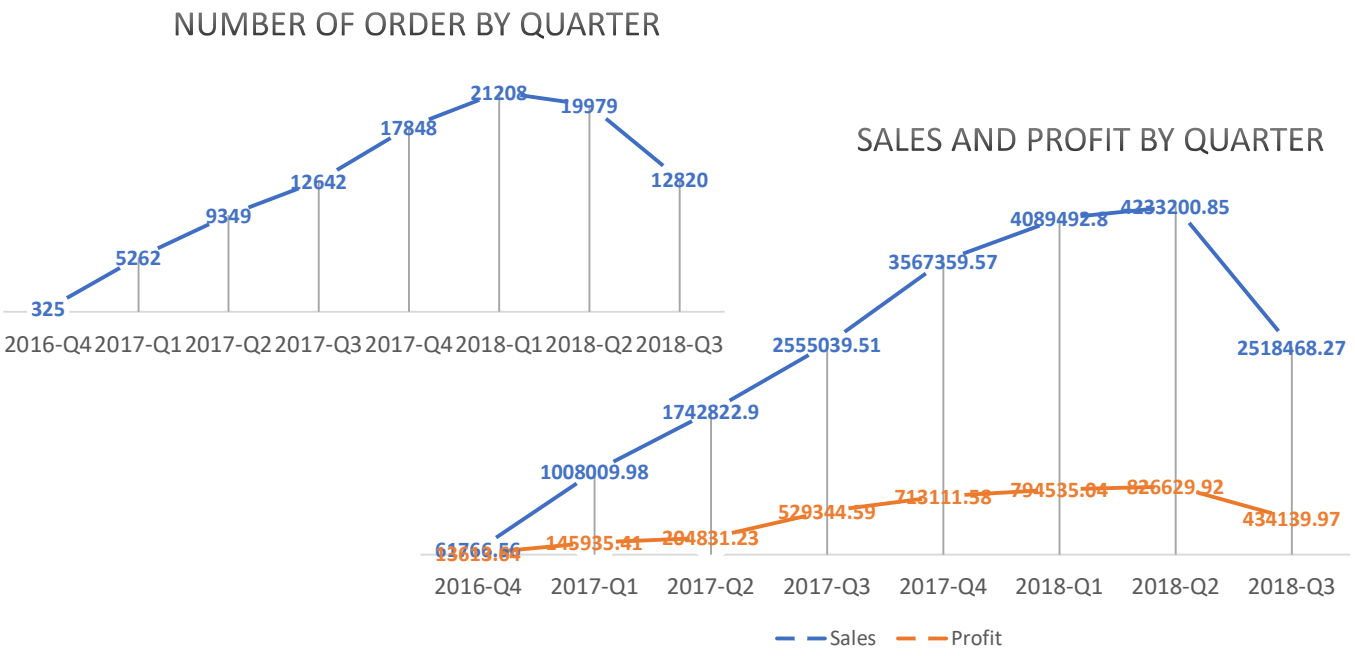
SQL Query:

```
select
  payment_installments,
  count(order_id) as orders_count
from `tgt-sql-20221006.Tgt_01.payments`
group by 1;
```

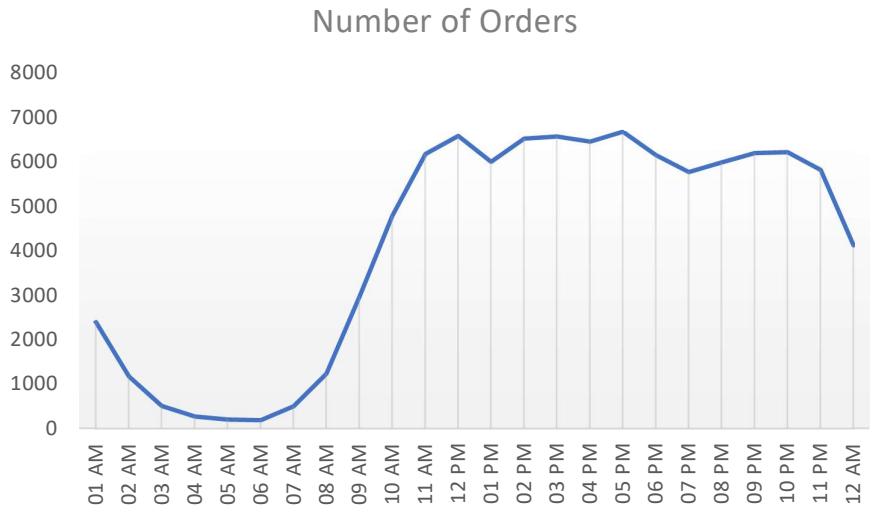
Sample Output:

payment_installments	orders_count
0	2
1	52546
2	12413
3	10461
4	7098
5	5239
6	3920
7	1626
8	4268
9	644
10	5328

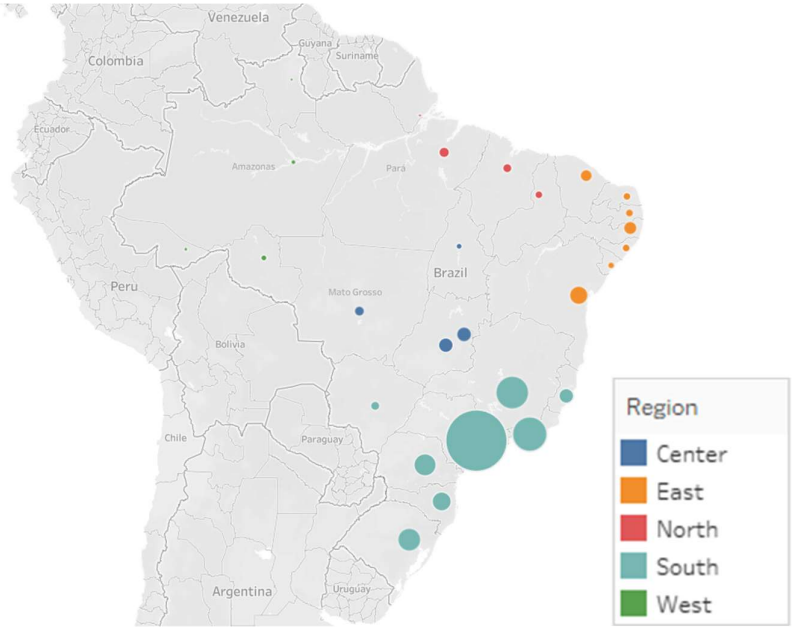
Actionable Insights:



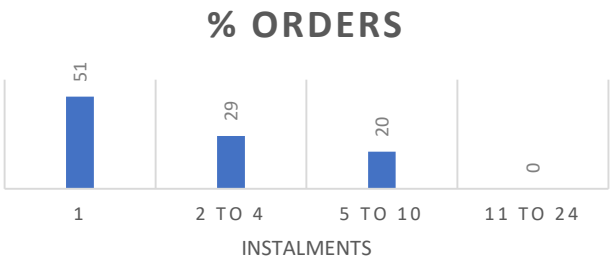
- On analysing number of orders, sales, and profit, we observed growing trend from the 4th quarter of 2016 (i.e., from October 2016) till the 1st quarter of 2018 (i.e., March 2018). Then the trend is stable for quarter 2 of 2018 and thereafter it starts decreasing in the later phase, i.e., quarter 3 of 2018.



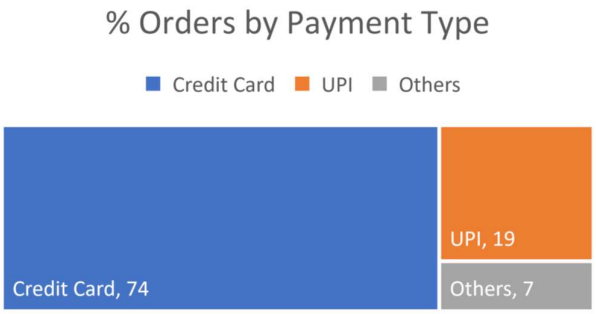
- Customers tend to purchase more from 10AM to 11PM compared to the rest of the day. The consumer foot fall is almost same throughout that period. After that the foot fall starts to reduce till 3AM. Form 3AM to 6AM it is almost negligible. After 6AM it starts to increase again till it reaches the peak.



- Customers are mainly concentrated in southern region of Brazil. Whereas western region of Brazil has the least number of customers.
- There is 142.79% increase in the cost of order from 2017 to 2018 (considering months from January to August).
- On average it takes around 13 days to deliver the orders when it is estimated (average) to deliver in 24 days. Overall orders are delivered much earlier than the estimated delivery date.



- Half of the customers prefer to buy their products by paying upfront (in 1 instalment). 29% of the customers paid in 2 to 4 instalments. 20% of the customers paid in 5 to 10 instalments. And remaining 1% paid in 11 to 24 instalments.



- 74% of the customers prefer to buy using credit card payment system and around 19 % prefer to buy using UPI.

customer_state	total_price	total_freight_value	mean_freight_value
SP	5108326.33	707599.33	15.12255199
PR	671849.26	116389.9	20.46955681
PB	114416.49	25565.65	42.82353434
RR	7739.44	2209.53	43.32411765

- On observing state wise total and mean price and freight value; freight value per item decreases with the increase in the total freight value of all the items in the state. i.e, for state "SP" the total freight and total price is highest, but the mean price and mean freight value is lowest. On the other hand, for state "RR" the total freight and total price is lowest, but the mean price and mean freight value is highest.

from_state	to_state	mean_freight_value	mean_diff_estimated_delivery
AC	AC	32.84	40.40
AC	AL	32.84	31.99
AM	AC	27.27	40.40
AM	AL	27.27	31.99
BA	AL	30.64	31.99
BA	AM	30.64	44.59

- The freight value is constant for each seller state irrespective of where it is going. e.g. For seller state "AC", when an order is delivered to the same (customer) state "AC", the mean freight value is same as when an order is delivered to a customer state that is farthest form the seller state "AC". From seller state "AC" the mean freight value is $32.84 for all the customer states. Whereas this value is different for seller state "AM" (avg. $ 27.27) and so on for all the other seller states.
- The mean time to delivery and the mean difference between estimated and actual delivery for each customer state is constant irrespective of where it is coming from. e.g. For customer state "AC", when an order is received from the same (seller) state "AC", the mean estimated delivery time is same as when an order is received from a seller state that is farthest form the customer state "AC". From customer state "AC" the mean estimated delivery time is 40.4 days for all the seller states. Whereas this value is different for customer state "AL" (avg. 31.99 days) and so on for all the other customer states.

Recommendations:

- Since most of the customers are concentrated in the southern region, we can plan new outlets in this region. In western region there is less penetration of our brand, so we can plan targeted advertisement and promotional marketing strategies near our existing stores.
- Since the customers prefer to buy during 10AM to 11PM, management should ensure to have efficient staff working and effective inventory management so that the customers need not wait, and the most favourite items are always on the shelf. Also, to encourage customer to buy during non-peak hours (11PM to 6AM), we can offer discounts or flash sale. Most of the maintenance and restocking activities shall be carried out during the non-peak hours (11PM to 6AM).
- The mechanism to decide the freight value and estimated delivery time needs rethinking. Present system does not consider the distance from seller location to customer location, which is the main factor affecting the freight value and delivery time and need to be considered.
- On average the orders are delivered much earlier than the estimated delivery date. It shows the mechanism to calculate estimated delivery date needs rethinking. We can aim to achieve 1 to 2 days safe buffer between actual delivery and estimated delivery date (which is presently around 10 days).
- Most of the customers prefer to pay in lesser instalments, shows that the customers have good purchasing power. We can encourage those customers to purchase more by targeted offers.
- Since most of the customers prefer payment using credit cards and UPI, during low sales period e.g., during non-peak hours, we can have exclusive offers or sales for credit card customers or UPI customers.
- As the mean freight value decreases with the increase in sales, we shall try to improve our sales to minimize the overhead and to have higher profit margins.