SQL: “Target” Business Case

Table of content

==========*==========

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 of customers ordered during the given period**

(1.1) Data type of columns in a table

Resources:
https://cloud.google.com/bigquery/docs/information-schema-intro
https://cloud.google.com/bigquery/docs/information-schema-columns

Query:

```
SELECT
  table_name,
  column_name,
  data_type
FROM Target_SQL.INFORMATION_SCHEMA.COLUMNS;
```

O/P:

Row	table_name	column_name	data_type
1	order_items	order_id	STRING
2	order_items	order_item_id	INT64
3	order_items	product_id	STRING
4	order_items	seller_id	STRING
5	order_items	shipping_limit_date	TIMESTAMP
6	order_items	price	FLOAT64
7	order_items	freight_value	FLOAT64
8	sellers	seller_id	STRING
9	sellers	seller_zip_code_prefix	INT64
10	sellers	seller_city	STRING

Actionable Insights:
The table provides details about the data types of respective columns belonging to different tables in the dataset.

(1.2) Time period for which the data is given

Query:

```
SELECT
  MIN(order_purchase_timestamp) AS start_time_period_of_data,
  MAX(order_purchase_timestamp) AS end_time_perdiod_of_data
FROM `Target_SQL.orders`;
```

O/P:

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

Actionable Insights:
We see that the "Target" company dataset was provided from the start date of "2016-09-04" until the end data "2018-10-17"

(1.3) Cities and States of customers ordered during the given period

Query:

```
SELECT
  DISTINCT
  customer_state,
  customer_city,
  COUNT(customer_unique_id) AS count_cust_order_per_state_city
FROM `Target_SQL.customers`
GROUP BY customer_state, customer_city;
```

O/P:

Row	customer_state	customer_city	count_cust_order_per_state_city
1	RN	acu	3
2	CE	ico	8
3	RS	ipe	2
4	CE	ipu	4
5	SC	ita	3
6	SP	itu	136
7	SP	jau	74
8	MG	luz	2
9	SP	poa	85
10	MG	uba	53

Actionable Insights:
The table displays the count of orders being placed by the customers in the respective states and cities during the given dataset time period.

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)?**

(2.1) 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?

Query:

```
SELECT
  DISTINCT
  COUNT(o.order_id) AS order_id,
  EXTRACT(year FROM order_purchase_timestamp) AS Year,
  EXTRACT(month FROM order_purchase_timestamp) AS Month
FROM `Target_SQL.orders` as o
JOIN `Target_SQL.customers` as c
ON o.customer_id = c.customer_id
GROUP BY Year, Month
ORDER BY Year, Month;
```

O/P:

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

Actionable Insights:

As you can see from the table output, there seem to be an increase on the number of orders being placed by the customers as the time passes by starting from the year 2016.

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

Assumptions for this analysis:

Dawn – 12AM to 7AM
Morning - 8AM to 12PM
Afternoon - 1PM to 6PM
Night - 7PM to 11:59PM

Query:

```
SELECT
  COUNT(order_id) AS count_of_orders,
  CASE
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) >= '00:00:01' AND EXT
RACT(TIME FROM order_purchase_timestamp) < '07:00:00'
    THEN 'Dawn (00.00.01 - 07.00.00)'
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) >= '08:00:00' AND EXT
RACT(TIME FROM order_purchase_timestamp) < '12:00:00'
    THEN 'Morning (08.00.00 - 12.00.00)'
    WHEN EXTRACT(TIME FROM order_purchase_timestamp) >= '13:00:00' AND EXT
RACT(TIME FROM order_purchase_timestamp) < '18:00:00'
    THEN 'Afternoon (13.00.00 - 18.00.00)'
    ELSE 'Night (19.00.00 - 23.00.00)'
 END AS time
FROM `Target_SQL.orders`
GROUP BY time
ORDER BY COUNT(order_id) DESC;
```

O/P:

Row	count_of_orders	time
1	41327	Night (19.00.00 - 23.00.00)
2	32366	Afternoon (13.00.00 - 18.00.00)
3	20507	Morning (08.00.00 - 12.00.00)
4	5241	Dawn (00.00.01 - 07.00.00)

Actionable Insights:

We can see that most of the customers tend to buy/place order during the night time while the least order is placed at the morning time.

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

1. **Get month on month orders by states**
2. **Distribution of customers across the states in Brazil**

(3.1) Get month on month orders by states

Query:

```
SELECT
  DISTINCT
  c.customer_state,
  COUNT(o.order_id) AS order_id,
  EXTRACT(year FROM order_purchase_timestamp) AS Year,
  EXTRACT(month FROM order_purchase_timestamp) AS Month
FROM `Target_SQL.orders` as o
JOIN `Target_SQL.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state, Year, Month
ORDER BY Year, Month;
```

O/P:

Row	customer_state	order_id	Year	Month
1	RR	1	2016	9
2	RS	1	2016	9
3	SP	2	2016	9
4	SP	113	2016	10
5	RS	24	2016	10
6	RJ	56	2016	10
7	MT	3	2016	10
8	GO	9	2016	10
9	MG	40	2016	10
10	CE	8	2016	10

Actionable Insights:

The table displays the total number of orders placed by customers on monthly bases starting year 2016 until the year 2018 on each state respectively as per the given dataset.

(3.2) Distribution of customers across the states in Brazil

Query:

```
SELECT
  DISTINCT
  COUNT(customer_id) AS customer_count,
  customer_state
FROM `Target_SQL.customers`
GROUP BY customer_state
ORDER BY customer_count DESC;
```

O/P:

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

Actionable Insights:

We can see that the highest customers are located on the SP state who places orders in Target marketplace.

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

1. **Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table**
2. **Mean & Sum of price and freight value by customer state**

(4.1)Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table

Query:

```
SELECT
 *,
 CONCAT(ROUND((x.tot_pay_val - (LAG(x.tot_pay_val,1) OVER(ORDER BY year, m
onth)))*100/LAG(x.tot_pay_val,1)
    OVER(ORDER BY year,  month),2),"%") AS m_o_m_per100
FROM (SELECT
        EXTRACT(YEAR FROM DATE (order_purchase_timestamp)) AS year,
        EXTRACT(MONTH FROM DATE (order_purchase_timestamp)) AS month,
        ROUND(SUM(p.payment_value),2) AS tot_pay_val,
        FROM `Target_SQL.orders` AS o
        JOIN `Target_SQL.payments` AS p
        ON o.order_id = p.order_id
        GROUP BY year,month
        ORDER BY year,month) AS x
WHERE x.year BETWEEN 2017 AND 2018 AND x.month BETWEEN 1 AND 8
ORDER BY year, month;
```

O/P:

Row	year	month	tot_pay_val	m_o_m_per100
1	2017	1	138488.04	*null*
2	2017	2	291908.01	110.78%
3	2017	3	449863.6	54.11%
4	2017	4	417788.03	-7.13%
5	2017	5	592918.82	41.92%
6	2017	6	511276.38	-13.77%
7	2017	7	592382.92	15.86%
8	2017	8	674396.32	13.84%
9	2018	1	1115004.18	65.33%
10	2018	2	992463.34	-10.99%

Actionable Insights:

We can see that the % of cost per orders has been increasing since the beginning of the month January 2017 until the August of 2018 while ignoring the months Sep, Oct, Nov & Dec of 2017.

(4.2) Mean & Sum of price and freight value by customer state

Query:

```
SELECT
    c.customer_state,
    ROUND(AVG(oi.price), 2) AS Avg_price,
    ROUND(AVG(oi.freight_value), 2) AS Avg_freight,
    ROUND(SUM(oi.price), 2) AS Sum_price,
    ROUND(sum(oi.freight_value), 2) as Sum_freight
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state;
```

O/P:

Row	customer_state	Avg_price	Avg_freight	Sum_price	Sum_freight
1	MT	148.3	28.17	156453.53	29715.43
2	MA	145.2	38.26	119648.22	31523.77
3	AL	180.89	35.84	80314.81	15914.59
4	SP	109.65	15.15	5202955.05	718723.07
5	MG	120.75	20.63	1585308.03	270853.46
6	PE	145.51	32.92	262788.03	59449.66
7	RJ	125.12	20.96	1824092.67	305589.31
8	DF	125.77	21.04	302603.94	50625.5
9	RS	120.34	21.74	750304.02	135522.74
10	SE	153.04	36.65	58920.85	14111.47

Actionable Insights:

As per the table output, we can see that the sum of price and sum of freight is highest in the state SP when compared to other states in Brazil.

5. **Analysis on sales, freight and delivery time**
    1. **Calculate days between purchasing, delivering and estimated delivery**
    2. **Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:**
        - **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**

(5.1) Calculate days between purchasing, delivering and estimated delivery

Query:

```
SELECT
  order_id,
  order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
Day) AS Days_purchased,
  TIMESTAMP_DIFF(order_estimated_delivery_date,order_purchase_timestamp, D
ay) AS Days_est_delivery,
  TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_d
ate, Day) AS Days_delivery
FROM `Target_SQL.orders`;
```

O/P:

Row	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	Days_purchased	Days_est_delivery	Days_delivery
1	f88aac7e...	2017-12-09 10:16:45 UTC	null	2018-01-29 00:00:00 UTC	null	50	null
2	790cd37...	2018-08-10 15:14:50 UTC	null	2018-08-17 00:00:00 UTC	null	6	null
3	49db794...	2017-05-13 21:23:34 UTC	null	2017-06-27 00:00:00 UTC	null	44	null
4	063b573...	2016-10-07 19:17:00 UTC	null	2016-12-01 00:00:00 UTC	null	54	null
5	a68ce16...	2016-10-05 01:47:40 UTC	null	2016-12-01 00:00:00 UTC	null	56	null
6	4597391...	2016-10-07 22:45:28 UTC	null	2016-12-01 00:00:00 UTC	null	54	null
7	cda8735...	2016-10-05 16:57:30 UTC	null	2016-12-01 00:00:00 UTC	null	56	null
8	ead2068...	2018-03-08 07:06:35 UTC	null	2018-04-19 00:00:00 UTC	null	41	null
9	6f028ccb...	2018-08-05 07:21:56 UTC	null	2018-08-09 00:00:00 UTC	null	3	null
10	8733c8d...	2018-08-05 17:00:00 UTC	null	2018-08-09 00:00:00 UTC	null	3	null

Actionable Insights:

The table above shows the day's difference between purchasing, delivering, and estimated delivery days.

(5.2) Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**
- **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

Query:

```
SELECT
  order_id,
  TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_timestamp, Day) AS Days_time_to_delivery,
  TIMESTAMP_DIFF(order_delivered_customer_date,order_estimated_delivery_date, Day) AS Days_estimated_delivery
FROM `Target_SQL.orders`;
```

O/P:

Row	order_id	Days_time_to_delivery	Days_estimated_delivery
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5

Actionable Insights:

The days' time to delivery seem to be much longer than the estimated days of delivery in the Brazil country.

(5.3)Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Query:

```
SELECT
  DISTINCT c.customer_state,
  ROUND(AVG(oi.freight_value),0) AS Avg_freight_value,
  ROUND(AVG(TIMESTAMP_DIFF(order_purchase_timestamp, order_delivered_customer_date, Day)),0) AS Avg_time_to_delivery,
  Round(avg(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, Day)),0) AS Avg_estimated_delivery
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state;
```

O/P:

Row	customer_state	Avg_freight_value	Avg_time_to_delivery	Avg_estimated_delivery
1	MT	28.0	-18.0	14.0
2	MA	38.0	-21.0	9.0
3	AL	36.0	-24.0	8.0
4	SP	15.0	-8.0	10.0
5	MG	21.0	-12.0	12.0
6	PE	33.0	-18.0	13.0
7	RJ	21.0	-15.0	11.0
8	DF	21.0	-13.0	11.0
9	RS	22.0	-15.0	13.0
10	SE	37.0	-21.0	9.0

Actionable Insights:

The table displays the mean value of freight_value, time_to_delivery and estimated_delivery on each states respectively.

(5.4) Sort the data to get the following:

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

Query:

```
SELECT
  c.customer_state,
  ROUND(AVG(oi.freight_value),0) AS Highest_freight_value
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Highest_freight_value DESC
LIMIT 5;
```

O/P:

Row	customer_state	Highest_freight_value
1	PB	43.0
2	RR	43.0
3	RO	41.0
4	AC	40.0
5	PI	39.0

Actionable Insights:

The top 5 states having the highest average fright value sorted in descending order if reflected on the table output and we see that PB and RR are the two states that's having the highest freight value of 43.

(5.4.2) Top 5 states with highest/lowest average time to delivery

Query:

```
SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(order_delivered_customer_date,order_purchase_ti
mestamp, Day)),0) AS Avg_time_to_delivery,
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY Avg_time_to_delivery DESC
LIMIT 5;
```

O/P:

Row	customer_state	Avg_time_to_delivery
1	AP	28.0
2	RR	28.0
3	AM	26.0
4	AL	24.0
5	PA	23.0

Actionable Insights:

As the table output shown above, the states AP and RR are the top 2 states with highest average time to deliver the orders to the customer and then followed by the other 3 states.

(5.4.3) Top 5 states where delivery is really fast/ not so fast compared to estimated date

Query:

Fast estimated delivery

```
SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_
customer_date, Day)),1) AS fast_estimated_delivery
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY fast_estimated_delivery ASC
LIMIT 5;
```

O/P:

Row	customer_state	fast_estimated_delivery
1	AL	8.0
2	MA	9.1
3	SE	9.2
4	ES	9.8
5	BA	10.1

Query:

Slow estimated delivery

```
SELECT
  c.customer_state,
  ROUND(AVG(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_
customer_date, Day)),1) AS slow_estimated_delivery
FROM `Target_SQL.orders` AS o
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `Target_SQL.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY slow_estimated_delivery DESC
LIMIT 5;
```

O/P:

Row	customer_state	slow_estimated_delivery
1	AC	20.0
2	RO	19.1
3	AM	19.0
4	RR	17.4
5	AP	17.4

Actionable Insights:

The number one state which delivers the orders very fast is AL while on the other side, the number one state which takes the longest time to delivery is AC.

6. Payment type analysis:

1. **Month over Month count of orders for different payment types**
2. **Count of orders based on the no. of payment instalments**

(6.1) Month over Month count of orders for different payment types

Query:

```
SELECT
  DISTINCT
  EXTRACT(year FROM order_purchase_timestamp) AS Year,
  EXTRACT(month FROM order_purchase_timestamp) AS Month,
  COUNT(p.order_id) AS orders, p.payment_type
FROM `Target_SQL.payments` AS p
JOIN `Target_SQL.orders` AS o
ON p.order_id = o.order_id
GROUP BY p.payment_type,Year, Month
ORDER BY Year, Month;
```

O/P:

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

Actionable Insights:

The table displays the count of orders being placed by customers each month based on the different payment type options available. We can also see that most of the customer's having credit_card payment type seems to be placing the orders each month when compared to other payment types like voucher or UPI.

(6.2) Count of orders based on the no. of payment instalments

Query:

```
SELECT
  COUNT(p.order_id) AS orders,
  p.payment_installments
FROM `Target_SQL.payments` AS p
JOIN `Target_SQL.orders` AS o
ON p.order_id = o.order_id
GROUP BY p.payment_installments;
```

O/P:

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

Actionable Insights:

As we can see from the table, customers wishing to have payment_installments of 1 seem to be placed the most number of orders and then followed by 2 and 3 payment instalments options.

Recommendation:

"Target" company dataset has been shared from the year 2016 at September 4th until the year 2018 at October 17th along with the various distributions of states and its respective cities.

We can also see that there is a positive growth trend in "Target" since the beginning of the dataset at the year 2016 onwards as majority of the Brazil citizens tend to spend time of placing the orders during the night time between 7PM to 11PM approximately on the E-commerce platform.

To add up on this, since the highest number of customers being placed by the “Target” e-commerce platform is by the SP State, this would again be a very important point that the company can take advantage of that would help to expand the business for obtaining more and more customer’s in that region.

Keeping that aside, I do see that the time gap between the estimated date of arrival and the delivery date seem to be quite high which makes the customer's to wait for a very long time which indeed might end up in having a negative review to the "Target" company from the customer's end.

Since the highest freight_value is on the PB state and the highest average time to delivery is on the AP state, it would be recommending to double check these sates and see what alternative approaches can be implemented to obtain a faster delivery output to the customer's end.