TARGET SQL BUSINESS CASE SOLUTIONS DOC

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

- Data type of columns in a table

```
SELECT Column_name, Data_type
FROM sql_target.INFORMATION_SCHEMA.COLUMNS;
```

This query can give an overall view of all the data types of all columns of tables available in the db in a single view.

Row	Column_name	Data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64
8	seller_id	STRING
9	seller_zip_code_prefix	INT64
10	seller_city	STRING

- Time period for which data is given

```
SELECT MIN(order_purchase_timestamp) AS start_datetime,
MAX(order_purchase_timestamp) AS end_datetime
FROM `sql_target.orders`;
```

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

We can observe that as per the data available, first order was placed on 04/09/2016 at 21:15 pm and last order was placed at 17/10/2018 at 17:30 pm.

- Cities and States of customers ordered during the given period

```
SELECT DISTINCT customer_state, customer_city
FROM `sql_target.customers`;
```

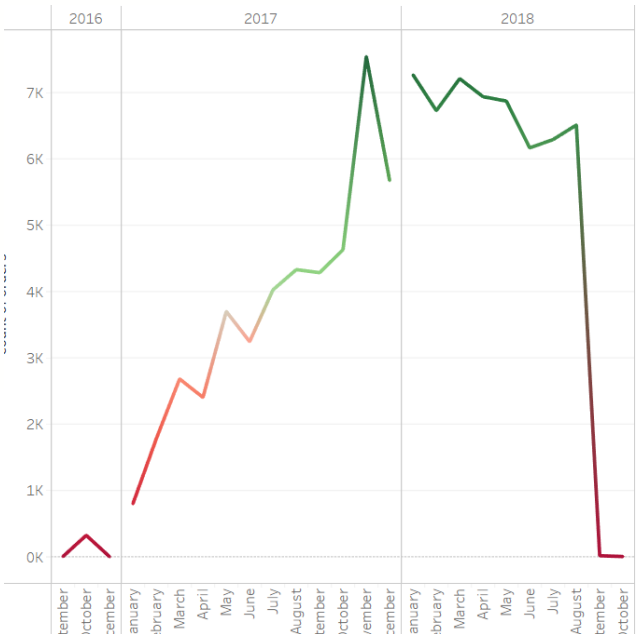
Row	customer_state	customer_city
1	RN	acu
2	CE	ico
3	RS	ipe
4	CE	ipu
5	SC	ita
6	SP	itu
7	SP	jau
8	MG	luz
9	SP	poa
10	MG	uba

2. In-depth exploration

- Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

```
SELECT EXTRACT(year from order_purchase_timestamp) AS year,
EXTRACT(month from order_purchase_timestamp) AS month,
COUNT(order_id) AS orders
FROM `sql_target.orders`
GROUP BY year, month
ORDER BY year ASC, month ASC
```

Row	year	month	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
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026



Since 2016 there has been a steady rise in shopping on Target's online website. 2017 saw an exponential growth in e-commerce trends. In November 2017, the highest number of orders i.e 7544 orders were placed. 2018 started off in the green zone with orders 6K+ but September saw a steep decline in numbers and stayed flatlined for October 2018.

To analyze the trends in terms of sales:

```
SELECT EXTRACT(year from order_purchase_timestamp) AS year,
SUM(p.payment_value) AS total_sales
FROM `sql_target.orders` o
JOIN `sql_target.payments` p
ON o.order_id = p.order_id
GROUP BY year
ORDER BY year ASC;
```

Row	year	total_sales
1	2016	59362.3400000000...
2	2017	7249746.72999974...
3	2018	8699763.04999977...

The above table shows us the total sales made in the respective years of the data given. From the above sales values it is clear that the e-commerce business has seen constant growth.

Recommendation: It would be recommended that Target expand their operations in Brazil.

- Can we see some seasonality with peaks at specific months?

```
SELECT EXTRACT(month from order_purchase_timestamp) AS months,
COUNT(order_id) AS total_orders
FROM `sql_target.orders`
GROUP BY months
ORDER BY months;
```

Row	months	total_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

If we analyze the seasonality in terms of sales:

```
SELECT EXTRACT(month from order_purchase_timestamp) AS month,
SUM(p.payment_value) AS total_sales
FROM `sql_target.orders` o
JOIN `sql_target.payments` p
ON o.order_id = p.order_id
GROUP BY month
ORDER BY month ASC;
```

Row	month	total_sales
1	1	1253492.22...
2	2	1284371.35...
3	3	1609515.71...
4	4	1578573.50...
5	5	1746900.96...
6	6	1535156.88...
7	7	1658923.67...
8	8	1696821.64...
9	9	732454.229...
10	10	839358.029...
11	11	1194882.80...
12	12	878421.100...

As observed from both sales and orders tables, it can be seen that from around April to August, there have been more than 9K+ online orders from the website and revenue for each month is more than 1 million, with highest grossing months being May and August.

Recommendation: It is observed that sales rise around Brazil's festive periods, hence it is advised that festive related and seasonal items be made available for shopping on the website.

- What time do Brazilian customers tend to buy?

```
SELECT EXTRACT(hour from order_purchase_timestamp)
AS time_of_the_day,
COUNT(order_id) AS total_orders
FROM `sql_target.orders`
GROUP BY 1
ORDER BY 1;
```

Row	time_of_the_day	total_orders
1	0	2394
2	1	1170
3	2	510
4	3	272
5	4	206
6	5	188
7	6	502
8	7	1231
9	8	2967
10	9	4785
11	10	6177
12	11	6578
13	12	5995
14	13	6518
15	14	6569
16	15	6454
17	16	6675
18	17	6150
19	18	5769
20	19	5982
21	20	6193
22	21	6217
23	22	5816
24	23	4123

From the above results we can observe that most purchases are being made during the afternoon and night.

Recommendations: Since dawn hours i.e between 02:00 am to 06:00 am see comparatively less orders on website, any website maintenance work and/or updates can be done during these hours

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

- Get month on month orders by states

```
SELECT DISTINCT c.customer_state AS state,
EXTRACT(month from o.order_purchase_timestamp) AS month,
COUNT(o.order_id) AS total_orders
FROM `sql_target.orders` o
LEFT JOIN `sql_target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state, month
ORDER BY state ASC, month ASC;
```

Row	state	month	total_orders
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39
14	AL	2	39
15	AL	3	40
16	AL	4	51

From above query we can get monthly insights on orders from all the states.

- Distribution of customers across the states in Brazil

```
SELECT customer_state AS state,
COUNT(customer_id) AS total_customers
FROM `sql_target.customers`
GROUP BY state
ORDER BY 2 DESC;
```

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

From the above table we can see the states having most to least customers.

Recommendation: Based on the number of customers, Target can decide on better marketing and expansion strategies for states with less customers, and they can do surveys, NPS, etc to know more about their customer base in states with more customers and upkeep those numbers.

4. Impact on economy:

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

```
SELECT
ROUND((total_payment_2018-total_payment_2017)*100/total_payment_2
017,2) AS total_pct_increase
FROM( SELECT
SUM(
CASE
WHEN  EXTRACT(year  from  o.order_purchase_timestamp)=2017  AND
EXTRACT(month from o.order_purchase_timestamp) BETWEEN 1 AND 8
THEN p.payment_value
END) AS total_payment_2017,
SUM(
CASE
WHEN  EXTRACT(year  from  o.order_purchase_timestamp)=2018  AND
EXTRACT(month from o.order_purchase_timestamp) BETWEEN 1 AND 8
THEN p.payment_value
END) AS total_payment_2018,
FROM `sql_target.orders` o
JOIN `sql_target.payments` p
ON o.order_id = p.order_id)
```

Row	total_pct_increase
1	136.98

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

```
SELECT c.customer_state,
ROUND(SUM(oi.price),2) AS sum_price,
ROUND(AVG(oi.price),2) AS mean_price,
ROUND(SUM(oi.freight_value),2) AS sum_freight_value,
ROUND(AVG(oi.freight_value),2) AS mean_freight_value
FROM `sql_target.order_items` oi
JOIN `sql_target.orders` o
ON oi.order_id = o.order_id
JOIN `sql_target.customers` c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state;
```

Row	customer_state	sum_price	mean_price	sum_freight_valu	mean_freight_va
1	SP	5202955.05	109.65	718723.07	15.15
2	RJ	1824092.67	125.12	305589.31	20.96
3	PR	683083.76	119.0	117851.68	20.53
4	SC	520553.34	124.65	89660.26	21.47
5	DF	302603.94	125.77	50625.5	21.04
6	MG	1585308.03	120.75	270853.46	20.63
7	PA	178947.81	165.69	38699.3	35.83
8	BA	511349.99	134.6	100156.68	26.36
9	GO	294591.95	126.27	53114.98	22.77
10	RS	750304.02	120.34	135522.74	21.74

5. Analysis on sales, freight and delivery time

- Calculate days between purchasing, delivering and estimated delivery

```
SELECT order_id,
DATE_DIFF(date(order_delivered_customer_date),
date(order_purchase_timestamp),day) AS time_to_delivery,
DATE_DIFF(date(order_estimated_delivery_date),
date(order_delivered_customer_date),day)
AS diff_estimated_delivery
FROM `sql_target.orders`;
```

Row	order_id	time_to_delivery	diff_estimated_delivery
1	1950d777989f6a877539f53795b4c3c3	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28c11c30	31	29
3	65d1e226dfaeb8cdc42f665422522d14	36	17
4	635c894d068ac37e6e03dc54eccb6189	31	2
5	3b97562c3aee8bdedcb5c2e45a50d5e1	33	1
6	68f47f50f04c4cb6774570cfde3a9aa7	30	2
7	276e9ec344d3bf029ff83a161c6b3ce9	44	-4
8	54e1a3c2b97fb0809da548a59f64c813	41	-4
9	fd04fa4105ee8045f6a0139ca5b49f27	37	-1
10	302bb8109d097a9fc6e9cefc5917d1f3	34	-5

Above result table shows days between purchase and delivery, and days between estimated time of delivery to actual day of delivery.
Here, negative(-) sign in diff_estimated_delivery indicates that the delivery was done x days later than the estimated dates.
And no sign / +ve day shows that the delivery was done x days before estimated delivery date.

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

```
SELECT c.customer_state,
ROUND(AVG(oi.freight_value),2) AS avg_freight_value,
ROUND(AVG(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp,day)),2) AS time_to_delivery,
ROUND(AVG(DATE_DIFF(date(order_estimated_delivery_date),
date(order_delivered_customer_date),day)),2)
AS diff_estimated_delivery
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state;
```

Row	customer_state	avg_freight_value	time_to_delivery	diff_estimated_delivery
1	RN	35.65	18.87	13.06
2	CE	32.71	20.54	10.26
3	RS	21.74	14.71	13.2
4	SC	21.47	14.52	10.67
5	SP	15.15	8.26	10.27
6	MG	20.63	11.52	12.4
7	BA	26.36	18.77	10.12
8	RJ	20.96	14.69	11.14
9	GO	22.77	14.95	11.37
10	MA	38.26	21.2	9.11

Above table gives us insights into average delivery time and difference between estimated and delivery date in terms of days for every state, and also average freight value in cost.

Recommendation: After observing the above table it can be concluded that states with negative or less average diff_estimated_delivery need to be focused on and improved in terms of faster delivery.

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

  Sort the data to get the following:

  - Top 5 states with highest average freight value

```
SELECT c.customer_state,
ROUND(avg(oi.freight_value),2) AS avg_freight_value
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY avg_freight_value DESC
LIMIT 5;
```

Row	customer_state	avg_freight_valu
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Freight value is the cost of shipping a product from one location to another. Above states have the highest average freight values.

Recommendation : It would be recommended that products / product categories with high freight values for mentioned states be extracted and involve strategies to marginalize their freight values by identifying local vendors for finding cheaper alternatives without degrading the product quality.

- Top 5 states with lowest average freight value

```
SELECT c.customer_state,
ROUND(avg(oi.freight_value),2) AS avg_freight_value
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY avg_freight_value ASC
LIMIT 5;
```

Row	customer_state	avg_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

These are the states with least freight values.

- Top 5 states with highest average time for delivery

```
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF(order_delivered_customer_date),
order_purchase_timestamp,day)),2) AS time_to_delivery,
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY time_to_delivery DESC
LIMIT 5;
```

Row	customer_state	time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

Recommendations: As the above mentioned states are also amongst the states with a weak customer base, more focus on lowering the delivery time and business strategies to attract more customers online should be made.

- Top 5 states with lowest average time for delivery

```
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF((order_delivered_customer_date),
(order_purchase_timestamp),day)),2) AS time_to_delivery,
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY time_to_delivery ASC
LIMIT 5;
```

Row	customer_state	time_to_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

- Top 5 states where delivery is really fast compared to estimated date:

```
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date,day)),2) AS diff_estimated_delivery
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY diss_estimated_delivery DESC
LIMIT 5;
```

Row	customer_state	diff_estimated_delivery
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

- Top 5 states where delivery is not so fast compared to estimated date:

```
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF((order_estimated_delivery_date),
(order_delivered_customer_date),day)),2)
AS diff_estimated_delivery
FROM `sql_target.customers` c
JOIN `sql_target.orders` o
ON c.customer_id = o.customer_id
JOIN  `sql_target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY diff_estimated_delivery ASC
LIMIT 5;
```

Row	customer_state	diff_estimated_d
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

Payment type analysis :

- Month over month analysis of orders for different types of payment types

```
SELECT EXTRACT(month from o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(o.order_id) AS total_orders
FROM `sql_target.orders` o
JOIN `sql_target.payments` p
ON o.order_id = p.order_id
GROUP BY p.payment_type, month
ORDER BY p.payment_type, month;
```

Row	month	payment_type	total_orders
1	1	UPI	1715
2	2	UPI	1723
3	3	UPI	1942
4	4	UPI	1783
5	5	UPI	2035
6	6	UPI	1807
7	7	UPI	2074
8	8	UPI	2077
9	9	UPI	903
10	10	UPI	1056
11	11	UPI	1509
12	12	UPI	1160
13	1	credit_card	6103
14	2	credit_card	6609
15	3	credit_card	7707
16	4	credit_card	7301

- Count of orders based on number of payment installments

```
SELECT payment_installments,
COUNT(order_id) AS total_orders
FROM `sql_target.payments`
GROUP BY payment_installments
ORDER BY 1;
```

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