Business Case: Target SQL

Scaler DS ML

Shayantan Dey

24th July

GitHub Repository for the case study

Context:

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

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

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

Dataset:

The data is available in 8 csv files at Google Drive

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

The column description for these csv files is given below.

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

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

The **sellers.csv** contains following features:

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

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

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

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

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

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

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

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

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

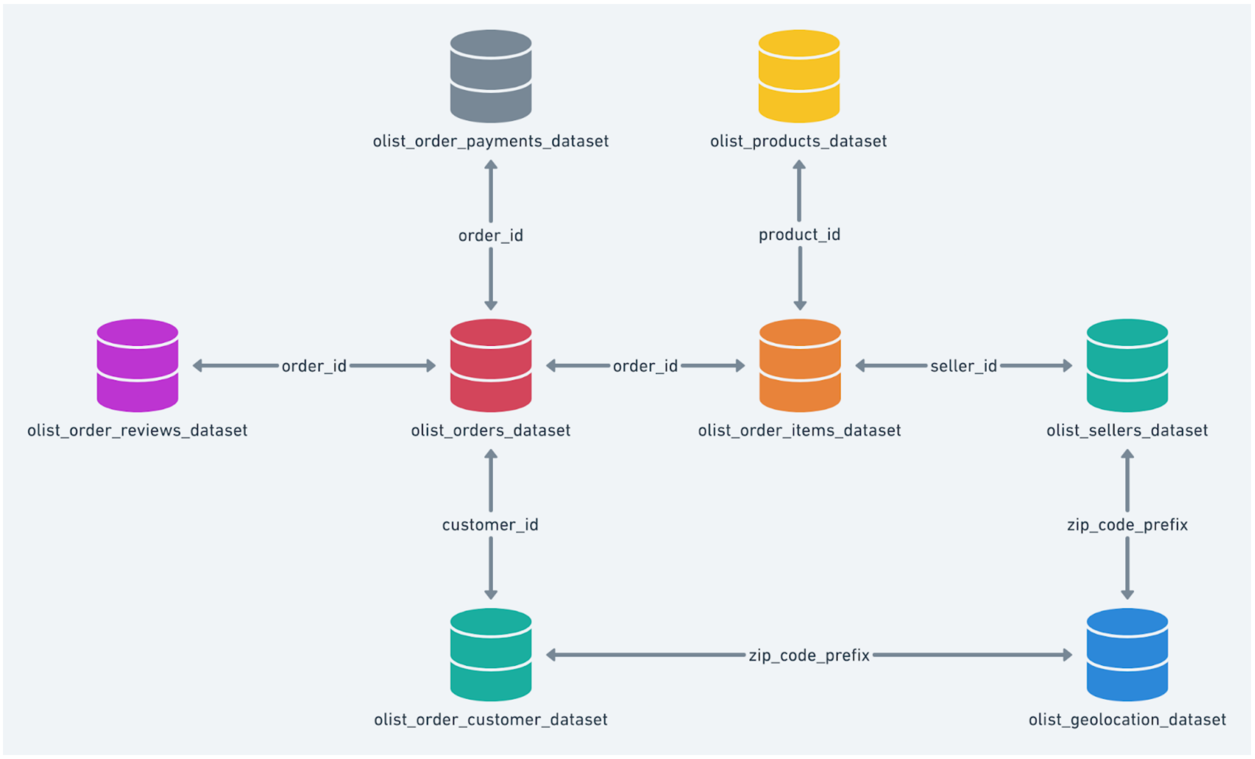
The **reviews.csv** contain following features:

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

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

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

Dataset schema:



Observations in the dataset

Two files, order_reviews.csv and geolocation.csv had unclean data.

Issues Identified in the order_reviews.csv file:

Encoding Issue: The file had to be read with ISO-8859-1 encoding instead of UTF-8.

Null Values: The review_comment_title column has many null values.

Date and Time Formatting: The review_creation_date and review_answer_timestamp columns are in string format and not properly parsed as datetime objects.

Steps to Correct Issues:

1. Ensure consistent encoding.
2. Handle null values in review_comment_title.
3. Convert date and time columns to proper datetime format.

Cleaning Data:

1. Strip leading/trailing spaces in text fields.
2. Replace any special characters or non-UTF-8 characters in text fields.
3. Check for null or empty values and handle them appropriately.

4. Convert date and time columns to datetime format.

Issues Identified in the geolocation.csv file:

Encoding Issue: The file had to be read with ISO-8859-1 encoding instead of UTF-8.

Null Values: The review_comment_title column has many null values.

Date and Time Formatting: The review_creation_date and review_answer_timestamp columns are in string format and not properly parsed as datetime objects.

Steps to Correct Issues:

1. Special characters in text fields.

2. Trailing or leading spaces.

3. Null or empty values.

4. Ensure that the file does not have any rows that might cause issues.

Cleaning Data:

1. Strip leading/trailing spaces in text fields.

2. Replace any special characters or non-UTF-8 characters in text fields.

3. Check for null or empty values and handle them appropriately.

All the 27 *geolocation_state* listed in the *geolocations.csv* file and *customer_state* in *customers.csv* are 26 states and 1 federal territory of Brazil. Hence, the data is specific to Brazil customers.

Problem Statement:

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

What does 'good' look like?

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

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

```
DESCRIBE customers;
```

Table 9: 5 records

Field	Type	Null	Key	Default	Extra
customer_id	text	YES		NA	
customer_unique_id	text	YES		NA	
customer_zip_code_prefix	text	YES		NA	
customer_city	text	YES		NA	
customer_state	text	YES		NA	

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

```
SELECT
    MIN(order_purchase_timestamp) AS order_start_date,
    MAX(order_purchase_timestamp) AS order_end_date,
    DATEDIFF(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp))
      AS order_time_range_days
FROM
    orders;
```

Table 10: 1 records

order_start_date	order_end_date	order_time_range_days
2016-09-04 21:15:19	2018-10-17 17:30:18	773

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

```
SELECT DISTINCT c.customer_city, c.customer_state, COUNT(*) AS customer_count
FROM orders AS o
JOIN customers AS c
ON o.customer_id = c.customer_id
GROUP BY c.customer_city, c.customer_state
ORDER BY customer_count DESC
```

Table 11: Displaying records 1 - 10

customer_city	customer_state	customer_count
sao paulo	SP	15540
rio de janeiro	RJ	6882
belo horizonte	MG	2773
brasilia	DF	2131
curitiba	PR	1521
campinas	SP	1444
porto alegre	RS	1379
salvador	BA	1245
guarulhos	SP	1189
sao bernardo do campo	SP	938

2. In-depth Exploration:

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

The purchases were made in the year 2016, 2017 and 2018.

```
SELECT DISTINCT YEAR(order_purchase_timestamp) AS year_of_orders
FROM orders
ORDER BY year_of_orders;
```

Table 12: 3 records

year_of_orders
2016
2017
2018

Trend for 2016 does not show conclusive evidence of a growing trend.

```sql
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2016") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2016
ORDER BY MONTH(order_purchase_timestamp);
```

Table 13: 3 records

month	month_number	order_count
September 2016	9	4
October 2016	10	324
December 2016	12	1

Trend for 2017 shows growth in month-on-month sale throughout the year.

```sql
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2017") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2017
ORDER BY MONTH(order_purchase_timestamp);
```

Table 14: Displaying records 1 - 10

month	month_number	order_count
January 2017	1	800
February 2017	2	1780
March 2017	3	2682
April 2017	4	2404
May 2017	5	3700
June 2017	6	3245
July 2017	7	4026
August 2017	8	4331
September 2017	9	4285
October 2017	10	4631

Trend for 2018 shows growth in month-on-month sale throughout the year.

```
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2018") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2018
ORDER BY MONTH(order_purchase_timestamp);
```

Table 15: Displaying records 1 - 10

month	month_number	order_count
January 2018	1	7269
February 2018	2	6728
March 2018	3	7211
April 2018	4	6939
May 2018	5	6873
June 2018	6	6167
July 2018	7	6292
August 2018	8	6512
September 2018	9	16
October 2018	10	4

Finding the sales per year shows a year-on-year growing trend.

```
SELECT DISTINCT YEAR(order_purchase_timestamp) AS year,
       COUNT(order_id) OVER(PARTITION BY YEAR(order_purchase_timestamp))
          AS count_of_orders
FROM orders;
```

Table 16: 3 records

year	count_of_orders
2016	329
2017	45101
2018	54011

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

Highest monthly sales in the given data is as follows, but it fails to show any seasonal trend:

```
SELECT YEAR(order_purchase_timestamp) as year,
       MONTHNAME(order_purchase_timestamp) as month,
       COUNT(*) as order_count
FROM orders
GROUP BY year, month
ORDER BY order_count DESC;
```

Table 17: Displaying records 1 - 10

year	month	order_count
2017	November	7544
2018	January	7269
2018	March	7211
2018	April	6939
2018	May	6873
2018	February	6728
2018	August	6512
2018	July	6292
2018	June	6167
2017	December	5673

While checking the year-wise monthly sales data, we do not see any monthly seasonality:

```sql
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2016") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2016
ORDER BY order_count DESC;
```

Table 18: 3 records

month	month_number	order_count
October 2016	10	324
September 2016	9	4
December 2016	12	1

```sql
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2017") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2017
ORDER BY order_count DESC;
```

Table 19: Displaying records 1 - 10

month	month_number	order_count
November 2017	11	7544
December 2017	12	5673
October 2017	10	4631
August 2017	8	4331
September 2017	9	4285
July 2017	7	4026
May 2017	5	3700

month	month_number	order_count
June 2017	6	3245
March 2017	3	2682
April 2017	4	2404

```sql
SELECT DISTINCT CONCAT(MONTHNAME(order_purchase_timestamp), " ", "2018") as month,
       MONTH(order_purchase_timestamp) as month_number,
       COUNT(order_id) OVER (PARTITION BY MONTH(order_purchase_timestamp))
        AS order_count
FROM orders
WHERE YEAR(order_purchase_timestamp) = 2018
ORDER BY order_count DESC;
```

Table 20: Displaying records 1 - 10

month	month_number	order_count
January 2018	1	7269
March 2018	3	7211
April 2018	4	6939
May 2018	5	6873
February 2018	2	6728
August 2018	8	6512
July 2018	7	6292
June 2018	6	6167
September 2018	9	16
October 2018	10	4

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

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

As per the data, Brazilian customers prefer placing their orders during afternoon.

```sql
SELECT DISTINCT d.time_of_day, COUNT(d.time_of_day) OVER(PARTITION BY d.time_of_day)
        AS count_of_orders
FROM
(SELECT customer_id, order_purchase_timestamp,
       CASE
          WHEN FLOOR(EXTRACT(HOUR FROM order_purchase_timestamp)) BETWEEN 0 AND 6 THEN
            "Dawn"
          WHEN FLOOR(EXTRACT(HOUR FROM order_purchase_timestamp)) BETWEEN 7 AND 12 THEN
            "Mornings"
          WHEN FLOOR(EXTRACT(HOUR FROM order_purchase_timestamp)) BETWEEN 13 AND 18 THEN
            "Afternoon"
          WHEN FLOOR(EXTRACT(HOUR FROM order_purchase_timestamp)) BETWEEN 19 AND 23 THEN
            "Night"
       END AS time_of_day
FROM orders) as d
ORDER BY count_of_orders DESC;
```

Table 21: 4 records

time_of_day	count_of_orders
Afternoon	38135
Night	28331
Mornings	27733
Dawn	5242

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

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

```
SELECT c.customer_state,
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
    MONTHNAME(o.order_purchase_timestamp) AS month_name,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month_number,
    COUNT(o.order_id) AS order_count
FROM customers as c
JOIN orders as o
ON c.customer_id = o.customer_id
GROUP By c.customer_state,
    EXTRACT(YEAR FROM o.order_purchase_timestamp),
    MONTHNAME(o.order_purchase_timestamp),
    EXTRACT(MONTH FROM o.order_purchase_timestamp)
ORDER BY c.customer_state, year, month_number;
```

Table 22: Displaying records 1 - 10

customer_state	year	month_name	month_number	order_count
AC	2017	January	1	2
AC	2017	February	2	3
AC	2017	March	3	2
AC	2017	April	4	5
AC	2017	May	5	8
AC	2017	June	6	4
AC	2017	July	7	5
AC	2017	August	8	4
AC	2017	September	9	5
AC	2017	October	10	6

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

Distribution of customers across states is as follows:

```
SELECT customer_state,
       COUNT(*) AS count_of_customers
FROM customers
GROUP BY customer_state
ORDER BY customer_state;
```

Table 23: Displaying records 1 - 10

customer_state	count_of_customers
AC	81
AL	413
AM	148
AP	68
BA	3380
CE	1336
DF	2140
ES	2033
GO	2020
MA	747

Distribution of customers across cities in those states is as follows:

```sql
SELECT customer_state, customer_city,
       COUNT(*) AS count_of_customers
FROM customers
GROUP BY customer_state, customer_city
ORDER BY customer_state, customer_city;
```

Table 24: Displaying records 1 - 10

customer_state	customer_city	count_of_customers
AC	brasileia	1
AC	cruzeiro do sul	3
AC	epitaciolandia	1
AC	manoel urbano	1
AC	porto acre	1
AC	rio branco	70
AC	senador guiomard	2
AC	xapuri	2
AL	agua branca	1
AL	anadia	2

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

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

```sql
SELECT d3.yearly_payment_value as 2017_payment_value,
       d3.lead_ as 2018_payment_value,
       ROUND((d3.diff/d3.yearly_payment_value) * 100, 2)
          as 2017_to_2018_percentage_increase
FROM
(
SELECT d2.year, d2.yearly_payment_value,
       LEAD(d2.yearly_payment_value) OVER(ORDER BY d2.yearly_payment_value) AS lead_,
       ((LEAD(d2.yearly_payment_value) OVER(ORDER BY d2.yearly_payment_value)
        - d2.yearly_payment_value)) as diff


FROM
(SELECT DISTINCT d.year,
       SUM(d.payment_value) OVER (PARTITION BY d.year) as yearly_payment_value
FROM
(SELECT EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
       p.payment_value
FROM orders as o
JOIN payments as p
ON o.order_id = p.order_id
WHERE (o.order_purchase_timestamp BETWEEN '2017-01-01' AND '2017-08-31')
      OR (o.order_purchase_timestamp BETWEEN '2018-01-01' AND '2018-08-31')
) AS d
) AS d2
) AS d3
WHERE d3.lead_ IS NOT NULL;
```

Table 25: 1 records

2017__payment__value	2018__payment__value	2017__to__2018__percentage__increase
3645107	8694670	138.53

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

```
SELECT DISTINCT c.customer_state,
       ROUND(SUM(p.payment_value) OVER(PARTITION BY c.customer_state), 2)
          AS total_order_price,
       ROUND(AVG(p.payment_value) OVER(PARTITION BY c.customer_state), 2)
          AS average_order_price
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
JOIN payments as p
ON o.order_id = p.order_id
ORDER BY c.customer_state;
```

Table 26: Displaying records 1 - 10

customer__state	total__order__price	average__order__price
AC	19680.62	234.29
AL	96962.06	227.08
AM	27966.93	181.60
AP	16262.80	232.33
BA	616645.82	170.82
CE	279464.03	199.90
DF	355141.08	161.13
ES	325967.55	154.71
GO	350092.31	165.76

customer_state	total_order_price	average_order_price
MA	152523.02	198.86

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

```sql
SELECT DISTINCT c.customer_state,
       ROUND(SUM(oi.freight_value) OVER(PARTITION BY c.customer_state), 2)
          AS total_freight_price,
       ROUND(AVG(oi.freight_value) OVER(PARTITION BY c.customer_state), 2)
          AS average_freight_price
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
JOIN order_items AS oi
ON o.order_id = oi.order_id
ORDER BY c.customer_state;
```

Table 27: Displaying records 1 - 10

customer_state	total_freight_price	average_freight_price
AC	3686.75	40.07
AL	15914.59	35.84
AM	5478.89	33.21
AP	2788.50	34.01
BA	100156.68	26.36
CE	48351.59	32.71
DF	50625.50	21.04
ES	49764.60	22.06
GO	53114.98	22.77
MA	31523.77	38.26

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

5.1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

The following are the order_status in the dataset.

```sql
SELECT DISTINCT order_status
FROM orders
```

Table 28: 8 records

order_status
delivered
invoiced
shipped
processing
unavailable
canceled
created
approved

We will consider the *delivered* order_status, for calculating *time_to_deliver*, because *order_delivered_customer_date* does not exist for the other *order_status*.

```sql
SELECT order_delivered_customer_date
FROM orders
WHERE order_status IN (
  SELECT DISTINCT order_status
  FROM orders
  WHERE order_status <> 'delivered'
)
```

Table 29: Displaying records 1 - 10

order_delivered_customer_date