

# Target

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.

**Dataset:** <https://drive.google.com/drive/folders/1TGEc66YKbD443nsIRi1bWgVd238gJCnb>

The data is available in 8 csv files:

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

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

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

SELECT \*

FROM Target.customers;

The screenshot shows a SQL IDE interface. On the left is an 'Explorer' pane with a tree view containing 'sql-end-project', 'External connections', and 'Target'. Under 'Target', there are icons for 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor area shows a query: 'SELECT \* FROM Target.customers;'. Below the editor is a 'Query results' section with tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', 'CHART', 'PREVIEW', and 'EXECUTION GRAPH'. The 'RESULTS' tab is active, displaying a table with 5 columns: 'customer\_id', 'customer\_unique\_id', 'customer\_zip\_code', 'customer\_city', and 'customer\_state'. The first three rows of data are visible. At the bottom, there are 'PERSONAL HISTORY' and 'PROJECT HISTORY' sections, and a 'REFRESH' button.

Row	customer_id	customer_unique_id	customer_zip_code	customer_city	customer_state
1	0735e7e4298a2ebbb4664934...	fc003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619e4eb...	46824822b15da44e983b021d...	59650	acu	RN
3	38c9766e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN

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

SELECT MIN(order\_purchase\_timestamp) AS start\_time,

MAX(order\_purchase\_timestamp) AS end\_time

FROM Target.orders;

The screenshot shows the Google Cloud SQL End Project interface. The Explorer on the left lists resources under 'sql-end-project', including 'External connections', 'Target', and various tables like 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor shows a query in 'Untitled 6':

```
1 SELECT MIN(order_purchase_timestamp) AS start_time,
2        MAX(order_purchase_timestamp) AS end_time
3 FROM Target.orders;
```

The 'Query results' section shows a table with two columns: 'start\_time' and 'end\_time'. The results are:

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

At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

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

```
SELECT COUNT(DISTINCT customer_city) AS num_unique_cities,
       COUNT(DISTINCT customer_state) AS num_unique_states
FROM Target.customers;
(there is no given period)
```

The screenshot shows the Google Cloud SQL End Project interface. The Explorer on the left lists resources under 'sql-end-project', including 'External connections', 'Target', and various tables like 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'sellers'. The main editor shows a query in 'Untitled 6':

```
1 SELECT COUNT(DISTINCT customer_city) AS num_unique_cities,
2        COUNT(DISTINCT customer_state) AS num_unique_states
3 FROM Target.customers;
4
5
```

The 'Query results' section shows a table with two columns: 'num\_unique\_cities' and 'num\_unique\_states'. The results are:

Row	num_unique_cities	num_unique_states
1	4119	27

At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

## 2.1 In-depth Exploration:

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

```
SELECT EXTRACT(YEAR FROM order_approved_at) AS order_year,
       COUNT(DISTINCT order_id) AS num_orders
FROM Target.orders
```

```
GROUP BY order_year
ORDER BY order_year;
```

The screenshot shows the Google Cloud SQL End Project interface. The query editor displays the following SQL:

```
1 SELECT EXTRACT(YEAR FROM order_approved_at) AS order_year,
2       COUNT(DISTINCT order_id) AS num_orders
3 FROM Target.orders
4 GROUP BY order_year
5 ORDER BY order_year;
```

The query results table shows the following data:

Row	order_year	num_orders
1	2016	160
2	2017	322
3	2017	44973
4	2018	53986

Explanation: Based on the orders placed we can see the increasing number of orders over the year.

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

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
       EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
       COUNT(order_id) AS no_of_orders
FROM Target.orders
GROUP BY year, month
ORDER BY no_of_orders DESC;
```

The screenshot shows the Google Cloud SQL End Project interface. The query editor displays the following SQL:

```
7 SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
8       EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
9       COUNT(order_id) AS no_of_orders
10 FROM Target.orders
11 GROUP BY year, month
12 ORDER BY no_of_orders DESC;
```

The query results table shows the following data:

Row	year	month	no_of_orders
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	2	6728
7	2018	8	6512
8	2018	7	6292
9	2018	6	6167
10	2017	12	5673

Explanation: Nov 2017 has the highest orders, maybe due to holiday season. Followed by 2018 Jan has the highest orders. In 2018, every month has orders of about 5000-6000 orders.

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

WITH OrderTimeCategories AS (

```
SELECT order_id,
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
ELSE 'Night'
END AS order_time_category
FROM Target.orders
)
```

```
SELECT order_time_category,
COUNT(order_id) AS no_of_orders
FROM OrderTimeCategories
GROUP BY order_time_category
ORDER BY no_of_orders DESC;
```

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project structure with 'Target' as the selected dataset. The main editor shows a SQL query titled 'Untitled 5' that defines a CTE 'OrderTimeCategories' and then counts the number of orders for each time category. The 'Query results' pane at the bottom shows the output of the query, which is a table with two columns: 'order\_time\_category' and 'no\_of\_orders'. The results are sorted in descending order of the number of orders.

Row	order_time_category	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

During **Afternoon** there is a higher orders placed in Brazil.

3.1 **Evolution of E-commerce orders in the Brazil region:** Get the month on month no. of orders placed in each state.

SELECT

```
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
c.customer_state,
COUNT(o.order_id) AS num_orders
```

```

FROM
  Target.orders o
JOIN
  Target.customers c ON o.customer_id = c.customer_id
GROUP BY
  order_year,
  order_month,
  c.customer_state
ORDER BY
  order_year,
  order_month,
  c.customer_state;

```

The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane showing the project hierarchy. The main area displays a query titled 'Untitled 5' with the following SQL code:

```

1 SELECT
2   EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
3   EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
4   c.customer_state,
5   COUNT(o.order_id) AS num_orders

```

Below the query editor, the 'Query results' section shows a table with the following data:

Row	order_year	order_month	customer_state	num_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4

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

```

SELECT customer_state, COUNT(DISTINCT customer_id) AS no_of_customers
FROM Target.customers
GROUP BY customer_state
ORDER BY no_of_customers DESC;

```

The screenshot shows the Google Cloud BigQuery console interface. The query editor displays the following SQL code:

```

1 SELECT customer_state, COUNT(DISTINCT customer_id) AS no_of_customers
2 FROM Target.customers
3 GROUP BY customer_state
4 ORDER BY no_of_customers DESC;

```

The 'Query results' section shows a table with the following data:

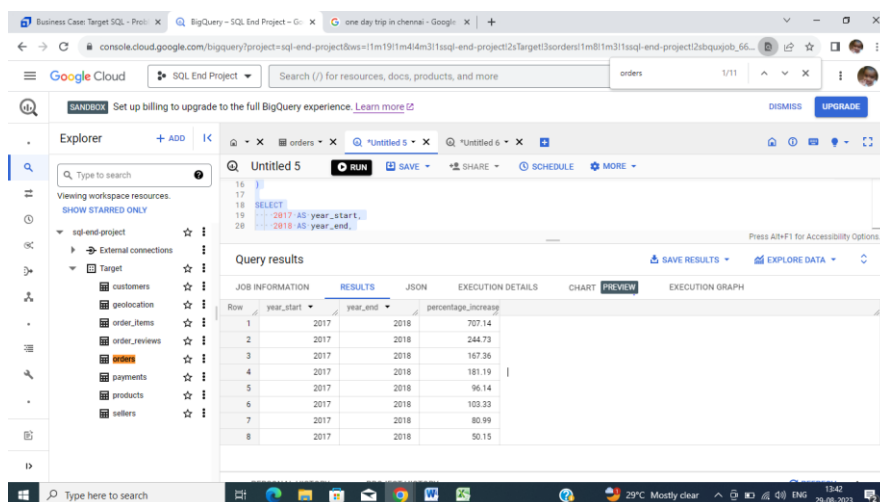
Row	customer_state	no_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	GO	3437

State code: SP (**São Paulo**) has highest number of customers and RR (Roraima) has the lowest number of customers

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

```
WITH YearlyCost AS (  
    SELECT  
        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,  
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,  
        SUM(oi.price + oi.freight_value) AS total_cost  
    FROM  
        Target.orders o  
    JOIN  
        Target.order_items oi ON o.order_id = oi.order_id  
    WHERE  
        EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)  
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8  
    GROUP BY  
        order_year,  
        order_month  
)  
  
SELECT  
    2017 AS year_start,  
    2018 AS year_end,  
    ROUND(((yc2018.total_cost - yc2017.total_cost) / NULLIF(yc2017.total_cost, 0)) * 100,  
2) AS percentage_increase  
FROM  
    YearlyCost yc2017  
JOIN  
    YearlyCost yc2018 ON yc2017.order_month = yc2018.order_month AND yc2017.order_year =  
2017 AND yc2018.order_year = 2018  
ORDER BY  
    yc2017.order_month;
```



The screenshot shows the Google Cloud BigQuery console. On the left is the Explorer pane showing the project structure. The main area displays a SQL query in the editor and its results in a table. The query calculates the percentage increase in order costs from 2017 to 2018, grouped by order month. The results table has 8 rows, one for each month from January to August.

Row	year_start	year_end	percentage_increase
1	2017	2018	707.14
2	2017	2018	244.73
3	2017	2018	167.36
4	2017	2018	181.19
5	2017	2018	96.14
6	2017	2018	103.33
7	2017	2018	80.99
8	2017	2018	50.15

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

## SELECT

```
c.customer_state,  
SUM(oi.price + oi.freight_value) AS total_order_price,  
AVG(oi.price + oi.freight_value) AS avg_order_price  
FROM  
Target.orders o  
JOIN  
Target.order_items oi ON o.order_id = oi.order_id  
JOIN  
Target.customers c ON o.customer_id = c.customer_id  
GROUP BY  
c.customer_state  
ORDER BY  
c.customer_state;
```

The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane showing the project structure with 'orders' selected. The main editor displays a SQL query (Untitled 5) that joins 'Target.orders', 'Target.order\_items', and 'Target.customers' tables, grouping by 'c.customer\_state' and ordering by 'c.customer\_state'. Below the editor, the 'Query results' section is visible, showing a table with 6 rows and 3 columns: 'customer\_state', 'total\_order\_price', and 'avg\_order\_price'. The results are as follows:

Row	customer_state	total_order_price	avg_order_price
1	AC	19669.7	213.8010869565...
2	AL	96229.4	216.7328828828...
3	AM	27835.72999999...	168.7013939393...
4	AP	16262.79999999...	198.3268292682...
5	BA	611506.6700000...	160.9651671492...
6	CE	275606.2999999...	186.4724627875...

At the bottom of the console, the Windows taskbar is visible, showing the system clock as 13:46 on 29-08-2023.

## 4.3 SELECT

```
c.customer_state,  
ROUND(SUM(oi.freight_value), 2) AS total_freight,  
ROUND(AVG(oi.freight_value), 2) AS avg_freight  
FROM  
Target.orders o  
JOIN  
Target.order_items oi ON o.order_id = oi.order_id  
JOIN  
Target.customers c ON o.customer_id = c.customer_id  
GROUP BY  
c.customer_state  
ORDER BY  
c.customer_state;
```

The screenshot shows the Google Cloud BigQuery console interface. The Explorer on the left lists the 'sql-end-project' and its datasets: 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'orders' dataset is selected. The main editor shows a SQL query for 'Untitled 6' that calculates total and average freight for each customer state. The 'Query results' section displays a table with 4 rows of data.

```

SELECT
  c.customer_state,
  ROUND(SUM(oi.freight_value), 2) AS total_freight,
  ROUND(AVG(oi.freight_value), 2) AS avg_freight
FROM
  Target.orders o
JOIN
  Target.order_items oi ON o.order_id = oi.order_id
JOIN
  Target.customers c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state

```

Row	customer_state	total_freight	avg_freight
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01

5.1 Analysis based on sales, freight and delivery time.

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:

$\text{time\_to\_deliver} = \text{order\_delivered\_customer\_date} - \text{order\_purchase\_timestamp}$

$\text{diff\_estimated\_delivery} = \text{order\_delivered\_customer\_date} - \text{order\_estimated\_delivery\_date}$

SELECT

```

o.order_id,
o.order_delivered_customer_date - o.order_purchase_timestamp AS time_to_deliver,
o.order_estimated_delivery_date - o.order_delivered_customer_date AS

```

diff\_estimated\_delivery

FROM

```
Target.orders o;
```

The screenshot shows the Google Cloud BigQuery console interface. The Explorer on the left lists the 'sql-end-project' and its datasets: 'customers', 'geolocation', 'order\_items', 'order\_reviews', 'orders', 'payments', 'products', and 'sellers'. The 'orders' dataset is selected. The main editor shows a SQL query for 'Untitled' that calculates the time to deliver and the difference between estimated and actual delivery dates for each order. The 'Query results' section displays a table with 4 rows of data.

```

SELECT
  o.order_id,
  o.order_delivered_customer_date - o.order_purchase_timestamp AS time_to_deliver,
  o.order_estimated_delivery_date - o.order_delivered_customer_date AS diff_estimated_delivery
FROM
  Target.orders o;

```

Row	order_id	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e05...	0-0 0 5031:5:12	0-0 0 -4358:36:39
2	1b3190b2cfa9d789e1f14c05b...	0-0 0 5000:26:32	0-0 0 -4535:24:7
3	440d0d17af552815d15a9e41a...	0-0 0 4695:12:59	0-0 0 -3975:12:50
4	2fb597c2772eca01b1f5c561b...	0-0 0 4676:24:15	0-0 0 -3734:33:17



Explanation: Improve logistics and shipping processes to reduce delivery times and enhance customer satisfaction.

5.2 Find out the top 5 states with the highest & lowest average freight value.

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

The screenshot shows the Google Cloud BigQuery console interface. The SQL query is displayed in the editor, and the results are shown in a table below. The table has two columns: 'customer\_state' and 'avg\_freight'. The results are ordered by 'avg\_freight' in descending order, showing the top 5 states.

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

Explanation Roraima has highest freight value and Piauí has the lowest value

5.3. Find out the top 5 states with the highest & lowest average delivery time.

```
SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    p.payment_type,
    COUNT(o.order_id) AS num_orders
```

6.1 Analysis based on the payments: Find the month on month no. of orders placed using different payment types.

```
FROM
    Target.orders o
JOIN
    Target.payments p ON o.order_id = p.order_id
GROUP BY
    order_year,
    order_month,
    payment_type
ORDER BY
    order_year,
    order_month,
    payment_type;
```

The screenshot shows the Google Cloud BigQuery console interface. The SQL query is displayed in the editor, and the results are shown in a table format. The query is a JOIN between Target.orders and Target.payments, grouped by order\_year, order\_month, and payment\_type, and ordered by the same columns. The results table shows 5 rows of data.

Row	order_year	order_month	payment_type	num_orders
1	2017	11	credit_card	5897
2	2018	3	credit_card	5691
3	2018	1	credit_card	5520
4	2018	5	credit_card	5497
5	2018	4	credit_card	5455

Explanation: credit card is the most favoured payment method.

6.2 Find the no. of orders placed on the basis of the payment installments that have been paid.

```
SELECT
    payment_installments,
    COUNT(order_id) AS num_orders
FROM
    Target.payments
GROUP BY
    payment_installments
ORDER BY
    payment_installments;
```

Business Case: Target SQL - x BigQuery - SQL End Project - x WhatsApp - x Harnessing SQL Power: Ins - x (31) dimla TL | LinkedIn - x

console.cloud.google.com/bigquery?project=sql-end-project&ws=11m191m414m311sssql-end-project12sTarget13orders11m811m311sssql-end-project12sbqjob\_2e83f5...

Google Cloud SQL End Project Search (/) for resources, docs, products, and more

SANDBOX Set up billing to upgrade to the full BigQuery experience. [Learn more](#) DISMISS UPGRADE

Explorer + ADD

Viewing workspace resources. SHOW STARRED ONLY

- sql-end-project
  - External connections
  - Target

Untitled RUN SAVE SHARE SCHEDULE MORE

```
2 payment_installments,
3 COUNT(order_id) AS num_orders
4 FROM
5 Target.payments
6 GROUP BY
7 payment_installments
8 ORDER BY
9 payment_installments
10
```

Processing location: US Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

Row	payment_installment	num_orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098

Results per page: 50 1 - 24 of 24

Type here to search 28°C Partly cloudy 12:03 30-08-2023

Most of customers paid in single instalments.