

Target Case Study

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 column_name, data_type
FROM `case-study-target-397515.Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```

```
9 SELECT column_name, data_type
10 FROM `case-study-target-397515.Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
11 WHERE table_name = 'customers'
```

Query results

[SAVE RESULTS](#) [EXPLORE DATA](#)

| JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|--------------------------|-----------|-------------------|-------|---------|-----------------|
| Row | column_name | data_type | | | | |
| 1 | customer_id | STRING | | | | |
| 2 | customer_unique_id | STRING | | | | |
| 3 | customer_zip_code_prefix | INT64 | | | | |
| 4 | customer_city | STRING | | | | |
| 5 | customer_state | STRING | | | | |

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

```
SELECT MIN(order_purchase_timestamp) as min_date,
MAX(order_purchase_timestamp) as max_date
from `Target_Case_Study.orders`;
```

```
16 SELECT MIN(order_purchase_timestamp) as min_date,
17 MAX(order_purchase_timestamp) as max_date
18 from `Target_Case_Study.orders`;
```

Query results

| JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW |
|-----------------|-------------------------|-------------------------|-------------------|-------|---------|
| Row | min_date | max_date | | | |
| 1 | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC | | | |

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

```
SELECT g.geolocation_city, g.geolocation_state, COUNT(c.customer_id) AS
customer_count
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
```

```

WHERE o.order_purchase_timestamp BETWEEN '2016-09-04 21:15:19 UTC' AND
'2018-10-17 17:30:18 UTC'
GROUP BY g.geolocation_city, g.geolocation_state
ORDER BY g.geolocation_state, g.geolocation_city;

```

```

19
20 SELECT g.geolocation_city, g.geolocation_state, COUNT(c.customer_id) AS customer_count
21 FROM `Target_Case_Study.orders` o
22 JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
23 JOIN `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
24 WHERE o.order_purchase_timestamp BETWEEN '2016-09-04 21:15:19 UTC' AND '2018-10-17 17:30:18 UTC'
25 GROUP BY g.geolocation_city, g.geolocation_state
26 ORDER BY g.geolocation_state, g.geolocation_city;

```

Query results

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRA |
|-----------------|------------------|-------------------|----------------|-------------------|-------|---------|---------------|
| Row | geolocation_city | geolocation_state | customer_count | | | | |
| 1 | brasileia | AC | 8 | | | | |
| 2 | brasiléia | AC | 5 | | | | |
| 3 | cruzeiro do sul | AC | 321 | | | | |
| 4 | epitaciolandia | AC | 6 | | | | |
| 5 | epitaciolândia | AC | 6 | | | | |
| 6 | manoel urbano | AC | 7 | | | | |
| 7 | porto acre | AC | 3 | | | | |
| 8 | rio branco | AC | 7 | | | | |

Results per page

2. In-depth Exploration:

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

```

SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  COUNT(DISTINCT order_id) AS order_count
FROM
  `Target_Case_Study.orders`
WHERE
  order_status = 'delivered'
  AND order_purchase_timestamp IS NOT NULL
GROUP BY
  year
ORDER BY
  year ASC

```

```

27
28 SELECT
29   EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
30   COUNT(DISTINCT order_id) AS order_count
31 FROM
32   `Target_Case_Study.orders`
33 WHERE
34   order_status = 'delivered'
35   AND order_purchase_timestamp IS NOT NULL
36 GROUP BY
37   year
38 ORDER BY
39   year ASC
40

```

Query results

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETAILS |
|-----------------|--------|---------------|------|-------------------|
| Row | year ▼ | order_count ▼ | | |
| 1 | 2016 | 267 | | |
| 2 | 2017 | 43428 | | |
| 3 | 2018 | 52783 | | |

- b. 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,
  COUNT(DISTINCT order_id) AS order_count
FROM
  `Target_Case_Study.orders`
WHERE
  order_status = 'delivered'
  AND order_purchase_timestamp IS NOT NULL
GROUP BY
  year
ORDER BY
  year ASC

```

```

40
41 SELECT
42 EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
43 COUNT(DISTINCT order_id) AS order_count
44 FROM
45 `Target_Case_Study.orders`
46 WHERE
47 order_status = 'delivered'
48 AND order_purchase_timestamp IS NOT NULL
49 GROUP BY
50 month
51 ORDER BY
52 month ASC
53

```

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETA |
|-----------------|---------|---------------|------|----------------|
| Row | month ▼ | order_count ▼ | | |
| 1 | 1 | 7819 | | |
| 2 | 2 | 8208 | | |
| 3 | 3 | 9549 | | |
| 4 | 4 | 9101 | | |
| 5 | 5 | 10295 | | |
| 6 | 6 | 9234 | | |
| 7 | 7 | 10031 | | |
| 8 | 8 | 10544 | | |
| 9 | 9 | 4151 | | |
| 10 | 10 | 4743 | | |
| 11 | 11 | 7289 | | |
| 12 | 12 | 5514 | | |

- c. 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

```

SELECT
CASE
    WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP)) AT TIME
ZONE 'America/Sao_Paulo' >= 5 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP)) AT TIME ZONE 'America/Sao_Paulo' < 12 THEN 'Morning'

```

```

        WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP)) AT TIME
ZONE 'America/Sao_Paulo') >= 12 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP)) AT TIME ZONE 'America/Sao_Paulo') < 18 THEN 'Afternoon'
        WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP)) AT TIME
ZONE 'America/Sao_Paulo') >= 18 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP)) AT TIME ZONE 'America/Sao_Paulo') < 24 THEN 'Night'
        ELSE 'Dawn'
    END AS time_of_day,
    COUNT(DISTINCT orders.customer_id) AS num_customers
FROM
    Target_Case_Study.orders
JOIN
    Target_Case_Study.customers
ON
    orders.customer_id = customers.customer_id
WHERE
    customers.customer_state = 'BR'
    AND CAST(order_purchase_timestamp AS TIMESTAMP) AT TIME ZONE
'America/Sao_Paulo' BETWEEN '2016-09-04 21:15:19' AND '2018-10-17 17:30:18'
GROUP BY
    time_of_day
ORDER BY
    num_customers DESC;

```

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

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

```

SELECT DATE_TRUNC(o.order_purchase_timestamp, MONTH) as month_year,
c.customer_state, COUNT(DISTINCT o.order_id) as num_orders
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c
ON o.customer_id = c.customer_id
GROUP BY month_year, customer_state
ORDER BY month_year, customer_state

```

```

90 |
91 | SELECT DATE_TRUNC(o.order_purchase_timestamp, MONTH) as month_year, c.customer_state, COUNT(DISTINCT o.order_id) as num_orders
92 | FROM `Target_Case_Study.orders` o
93 | JOIN `Target_Case_Study.customers` c
94 | ON o.customer_id = c.customer_id
95 | GROUP BY month_year, customer_state
96 | ORDER BY month_year, customer_state
97 |

```

Query results

[SAVE RESULTS](#)

| JOB INFORMATION | | | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|-------------------------|----------------|------------|---------|------|-------------------|-------|---------|-----------------|
| Row | month_year | customer_state | num_orders | | | | | | |
| 1 | 2016-09-01 00:00:00 UTC | RR | 1 | | | | | | |
| 2 | 2016-09-01 00:00:00 UTC | RS | 1 | | | | | | |
| 3 | 2016-09-01 00:00:00 UTC | SP | 2 | | | | | | |
| 4 | 2016-10-01 00:00:00 UTC | AL | 2 | | | | | | |
| 5 | 2016-10-01 00:00:00 UTC | BA | 4 | | | | | | |
| 6 | 2016-10-01 00:00:00 UTC | CE | 8 | | | | | | |
| 7 | 2016-10-01 00:00:00 UTC | DF | 6 | | | | | | |
| 8 | 2016-10-01 00:00:00 UTC | ES | 4 | | | | | | |
| 9 | 2016-10-01 00:00:00 UTC | GO | 9 | | | | | | |

Results per page: 50 1 50 of 565

Query results

[SAVE RESULTS](#)

| JOB INFORMATION | | | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|-------------------------|----------------|------------|---------|------|-------------------|-------|---------|-----------------|
| Row | month_year | customer_state | num_orders | | | | | | |
| 1 | 2016-09-01 00:00:00 UTC | RR | 1 | | | | | | |
| 2 | 2016-09-01 00:00:00 UTC | RS | 1 | | | | | | |
| 3 | 2016-09-01 00:00:00 UTC | SP | 2 | | | | | | |
| 4 | 2016-10-01 00:00:00 UTC | AL | 2 | | | | | | |
| 5 | 2016-10-01 00:00:00 UTC | BA | 4 | | | | | | |
| 6 | 2016-10-01 00:00:00 UTC | CE | 8 | | | | | | |
| 7 | 2016-10-01 00:00:00 UTC | DF | 6 | | | | | | |
| 8 | 2016-10-01 00:00:00 UTC | ES | 4 | | | | | | |
| 9 | 2016-10-01 00:00:00 UTC | GO | 9 | | | | | | |
| 10 | 2016-10-01 00:00:00 UTC | MA | 4 | | | | | | |
| 11 | 2016-10-01 00:00:00 UTC | MG | 40 | | | | | | |
| 12 | 2016-10-01 00:00:00 UTC | MT | 3 | | | | | | |

Results per page: 50 1 - 50 of 565

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

```

select customer_state, count(distinct customer_unique_id) as customer_num
from `Target_Case_Study.customers`
group by customer_state
order by customer_num DESC

```

2023-08-30 23:01:43

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

Syntax error: Expected end of input but got keyword

101

from Target_Case_Study.customers ;

102

103

select customer_state, count(distinct customer_unique_id) as customer_num

104

from Target_Case_Study.customers

105

group by customer_state

106

order by customer_num DESC

Press Alt+F1 for accessibility

Query results

SAVE RESULTS

EXPLORE DATA

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

CHART

PREVIEW

EXECUTION GRAPH

| Row | customer_state | customer_num |
|-----|----------------|--------------|
| 1 | SP | 40302 |
| 2 | RJ | 12384 |
| 3 | MG | 11259 |
| 4 | RS | 5277 |
| 5 | PR | 4882 |
| 6 | SC | 3534 |
| 7 | BA | 3277 |
| 8 | DF | 2075 |
| 9 | ES | 1964 |
| 10 | GO | 1952 |
| 11 | PE | 1600 |

Results per page: 50

1 - 27 of 27

PERSONAL HISTORY

PROJECT HISTORY

REFRESH

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

- a. 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.

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

```

SELECT
customer_state,
AVG(price) AS avg_price,
SUM(price) AS total_price,
AVG(freight_value) AS avg_freight_value,
SUM(freight_value) AS total_freight_value
FROM
`Target_Case_Study.orders` o
JOIN
`Target_Case_Study.order_items` oi
ON
o.order_id = oi.order_id
JOIN
`Target_Case_Study.customers` c

```

ON

o.customer_id = c.customer_id

GROUP BY

customer_state

| | | | | | | |
|--|---|-----------------------|------------------------------|-------------------------|----------------------------|-------------------------|
| 🔍 | 2023-08-30 23:01:43 | ▶ RUN | 💾 SAVE QUERY | 👤 SHARE | 🕒 SCHEDULE | ⚙️ MORE |
| 108 | SELECT | | | | | |
| 109 | customer_state, | | | | | |
| 110 | AVG(price) AS avg_price, | | | | | |
| 111 | SUM(price) AS total_price, | | | | | |
| 112 | AVG(freight_value) AS avg_freight_value, | | | | | |
| 113 | SUM(freight_value) AS total_freight_value | | | | | |
| 114 | FROM | | | | | |
| 115 | `Target_Case_Study.orders` o | | | | | |
| 116 | JOIN | | | | | |
| 117 | `Target_Case_Study.order_items` oi | | | | | |
| 118 | ON | | | | | |
| 119 | o.order_id = oi.order_id | | | | | |
| 120 | JOIN | | | | | |
| 121 | `Target_Case_Study.customers` c | | | | | |
| 122 | ON | | | | | |
| 123 | o.customer_id = c.customer_id | | | | | |
| 124 | GROUP BY | | | | | |
| 125 | customer_state | | | | | |
| Press A | | | | | | |
| Query results | | | | | | |
| 📄 SAVE RESULTS 📊 EXPORT | | | | | | |
| JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH | | | | | | |
| Row | customer_state | avg_price | total_price | avg_freight_value | total_freight_value | |
| 1 | SP | 109.6536291597... | 5202955.050002... | 15.14727539041... | 718723.0699999... | |
| 2 | RJ | 125.1178180945... | 1824092.669999... | 20.96092393168... | 305589.3100000... | |
| 3 | PR | 119.0041393728... | 683083.7600000... | 20.53165156794... | 117851.6800000... | |
| Results per page: 50 1 - 27 of 27 | | | | | | |

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

SELECT

customer_state,

AVG(price) AS avg_price,

SUM(price) AS total_price,

AVG(freight_value) AS avg_freight_value,

SUM(freight_value) AS total_freight_value

FROM

`Target_Case_Study.orders` o

JOIN

`Target_Case_Study.order_items` oi

ON

o.order_id = oi.order_id

JOIN

`Target_Case_Study.customers` c

ON

o.customer_id = c.customer_id

GROUP BY

customer_state

| | |
|-----|---|
| 108 | SELECT |
| 109 | customer_state, |
| 110 | AVG(price) AS avg_price, |
| 111 | SUM(price) AS total_price, |
| 112 | AVG(freight_value) AS avg_freight_value, |
| 113 | SUM(freight_value) AS total_freight_value |
| 114 | FROM |
| 115 | `Target_Case_Study.orders` o |
| 116 | JOIN |
| 117 | `Target_Case_Study.order_items` oi |
| 118 | ON |
| 119 | o.order_id = oi.order_id |
| 120 | JOIN |
| 121 | `Target_Case_Study.customers` c |
| 122 | ON |
| 123 | o.customer_id = c.customer_id |
| 124 | GROUP BY |
| 125 | customer_state, |

Query results
[SAVE RESULTS](#)
[EX](#)

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|----------------|-------------------|-------------------|-------------------|---------------------|---------|-----------------|
| Row | customer_state | avg_price | total_price | avg_freight_value | total_freight_value | | |
| 1 | SP | 109.6536291597... | 5202955.050002... | 15.14727539041... | 718723.0699999... | | |
| 2 | RJ | 125.1178180945... | 1824092.669999... | 20.96092393168... | 305589.3100000... | | |
| 3 | PR | 119.0041393728... | 683083.7600000... | 20.53165156794... | 117851.6800000... | | |

Results per page: 50 1 – 27 of 27

5. 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.

SELECT

order_id,

DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS

days_to_deliver,

DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) AS

days_to_estimated_delivery

FROM

`Target_Case_Study.orders`

WHERE

order_purchase_timestamp >= '2017-01-01' AND order_purchase_timestamp <

'2018-09-01'

```

126
127 SELECT
128 order_id,
129 DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS days_to_deliver,
130 DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) AS days_to_estimated_delivery
131 FROM
132 `Target_Case_Study.orders`
133 WHERE
134 order_purchase_timestamp >= '2017-01-01' AND order_purchase_timestamp < '2018-09-01'
135

```

Press Alt+F1 for accessibility

Query results [SAVE RESULTS](#) [EXPLORE DATA](#)

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|-------------------------------|-----------------|---------------------|-------------------|-------|---------|-----------------|
| Row | order_id | days_to_deliver | days_to_estimated_d | | | | |
| 1 | f88aac7ebccb37f19725a0753... | null | 50 | | | | |
| 2 | 790cd37689193dca0d00d2feb... | null | 6 | | | | |
| 3 | 49db7943d60b6805c3a41f547... | null | 44 | | | | |
| 4 | ead20687129da8f5d89d831bb... | null | 41 | | | | |
| 5 | 6f028ccb7d612af251aa442a1f... | null | 3 | | | | |
| 6 | 8733c8d440c173e524d2fab80... | null | 3 | | | | |

Results per page: 50 1 - 50 of 99092 [REFRESH](#)

PERSONAL HISTORY PROJECT HISTORY

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_estimated_delivery_date - order_delivered_customer_date

```

SELECT
order_id,
TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_delivery,
TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
AS diff_estimated_delivery
FROM
`Target_Case_Study.orders`;

```

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

```

SELECT
customer_state,
AVG(freight_value) AS avg_freight
FROM
`Target_Case_Study.order_items`
JOIN
`Target_Case_Study.orders`
ON
`Target_Case_Study.order_items`.order_id =
`Target_Case_Study.orders`.order_id
JOIN
`Target_Case_Study.customers`
ON

```

```

        `Target_Case_Study.orders`.customer_id =
        `Target_Case_Study.customers`.customer_id
GROUP BY
    customer_state
ORDER BY
    avg_freight DESC
LIMIT
5

```

2023-08-30 23:01:43

RUN
 SAVE QUERY
 SHARE
 SCHEDULE
 MORE

```

142
143 SELECT
144     customer_state,
145     AVG(freight_value) AS avg_freight
146 FROM
147     `Target_Case_Study.order_items`
148 JOIN
149     `Target_Case_Study.orders`
150 ON
151     `Target_Case_Study.order_items`.order_id = `Target_Case_Study.orders`.order_id
152 JOIN
153     `Target_Case_Study.customers`
154 ON
155     `Target_Case_Study.orders`.customer_id = `Target_Case_Study.customers`.customer_id
156 GROUP BY

```

Query results

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

CHART

PREVIEW

EXECUTION GRAPH

| Row | customer_state | avg_freight |
|-----|----------------|-------------------|
| 1 | RR | 42.98442307692... |
| 2 | PB | 42.72380398671... |
| 3 | RO | 41.06971223021... |
| 4 | AC | 40.07336956521... |
| 5 | PI | 39.14797047970... |

5

Five states with lowest freight value

```

SELECT
    customer_state,
    AVG(freight_value) AS avg_freight
FROM
    `Target_Case_Study.order_items`
JOIN
    `Target_Case_Study.orders`
ON
    `Target_Case_Study.order_items`.order_id =
    `Target_Case_Study.orders`.order_id
JOIN
    `Target_Case_Study.customers`
ON

```

```

        `Target_Case_Study.orders`.customer_id =
        `Target_Case_Study.customers`.customer_id
    GROUP BY
        customer_state
    ORDER BY
        avg_freight ASC
    LIMIT 5

```

The screenshot shows a SQL query editor interface. The query is as follows:

```

150 ON `Target_Case_Study.orders`.order_id = `Target_Case_Study.orders`.order_id
151 JOIN `Target_Case_Study.customers`
152 ON `Target_Case_Study.orders`.customer_id = `Target_Case_Study.customers`.customer_id
153 GROUP BY customer_state
154 ORDER BY avg_freight ASC
155 LIMIT 5

```

Below the query editor, the 'Query results' section is visible. It includes tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', 'CHART', 'PREVIEW', and 'EXECUTIC'. The 'RESULTS' tab is active, showing a table with the following data:

| Row | customer_state | avg_freight |
|-----|----------------|-------------------|
| 1 | SP | 15.14727539041... |
| 2 | PR | 20.53165156794... |
| 3 | MG | 20.63016680630... |
| 4 | RJ | 20.96092393168... |
| 5 | DF | 21.04135494596... |

At the bottom of the interface, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY'.

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

Highest

SELECT

```

    g.geolocation_state AS state,
    AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
    o.order_purchase_timestamp, DAY)) AS avg_delivery_time
FROM
    `Target_Case_Study.orders` o
JOIN
    `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN

```

+ ADD

IK

🏠

×

🔍 *2023-08-30 23:01:43

×

+

🔍 2023-08-30 23:01:43

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

```

164 | g.geolocation_state AS state,
165 | AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY)) AS avg_delivery_time
166 | FROM
167 | `Target_Case_Study.orders` o
168 | JOIN
169 | `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
170 | JOIN
171 | `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
172 | WHERE
173 | o.order_status = 'delivered'
174 | GROUP BY
175 | g.geolocation_state
176 | ORDER BY
177 | avg_delivery_time DESC
178 | LIMIT 5

```

Query results

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

CHART

PREVIEW

EXECUTION GRAPH

| Row | state | avg_delivery_time |
|-----|-------|-------------------|
| 1 | AP | 27.99122623772... |
| 2 | AM | 24.65119678421... |
| 3 | RR | 24.52060133630... |
| 4 | AL | 23.14352789271... |
| 5 | PA | 22.55023982441... |

```

Lowest
SELECT
    g.geolocation_state AS state,
    AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)) AS avg_delivery_time
FROM
    `Target_Case_Study.orders` o
JOIN
    `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN
    `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
WHERE
    o.order_status = 'delivered'
GROUP BY

```

```

    g.geolocation_state
ORDER BY
    avg_delivery_time ASC
LIMIT 5

```

2023-08-30 23:01:43
RUN
SAVE QUERY
SHARE
SCHEDULE
MORE

```

179
180 SELECT
181     g.geolocation_state AS state,
182     AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS avg_delivery_time
183 FROM
184     `Target_Case_Study.orders` o
185 JOIN
186     `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
187 JOIN
188     `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
189 WHERE
190     o.order_status = 'delivered'
191 GROUP BY
192     g.geolocation_state
193 ORDER BY

```

Query results
SAVE RESULTS

| JOB INFORMATION | | RESULTS | JSON | EXECUTION DETAILS | CHART | PREVIEW | EXECUTION GRAPH |
|-----------------|-------|-------------------|------|-------------------|-------|---------|-----------------|
| Row | state | avg_delivery_time | | | | | |
| 1 | SP | 8.470555045096... | | | | | |
| 2 | PR | 11.03876404770... | | | | | |
| 3 | MG | 11.41821678372... | | | | | |
| 4 | DF | 12.49651789233... | | | | | |
| 5 | SC | 14.48408434580... | | | | | |

PERSONAL HISTORY
PROJECT HISTORY

- f. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.


Top 5 states where the delivery was not so fast


```


SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS delivery_diff
FROM
    `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
WHERE
    o.order_status = 'delivered'
GROUP BY
    c.customer_state
HAVING
    delivery_diff < 0


```


```
ORDER BY
    delivery_diff ASC
LIMIT 5
```


 2023-08-30 23:01:43

 RUN

 SAVE QUERY

 SHARE

 SCHEDULE

 MORE

Query results 

Top 5 states where the delivery was fast

```
SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS delivery_diff
FROM
    `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
WHERE
    o.order_status = 'delivered'
GROUP BY
    c.customer_state
HAVING
    delivery_diff < 0
ORDER BY
    delivery_diff DESC
LIMIT
    5
```

2023-08-30 23:01:43

RUN

SAVE QUERY

SHARE

SCHEDULE

MORE

```

213 SELECT
214   c.customer_state,
215   AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS delivery_diff
216 FROM
217   `Target_Case_Study.orders` o
218 JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
219 WHERE
220   o.order_status = 'delivered'
221 GROUP BY
222   c.customer_state
223 HAVING
224   delivery_diff < 0
225 ORDER BY
226   delivery_diff DESC

```

Query results

JOB INFORMATION

RESULTS

JSON

EXECUTION DETAILS

CHART

PREVIEW

EXECUTION GRAPH

| Row | customer_state | delivery_diff |
|-----|----------------|-------------------|
| 1 | AL | -7.94710327455... |
| 2 | MA | -8.76847977684... |
| 3 | SE | -9.17313432835... |
| 4 | ES | -9.61854636591... |
| 5 | BA | -9.93488943488... |

e. **Analysis based on the payments:**

- Find the month on month no. of orders placed using different payment types.

```

SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
       p.payment_type,
       COUNT(DISTINCT o.order_id) AS order_count
FROM `Target_Case_Study.payments` p
JOIN `Target_Case_Study.orders` o ON p.order_id = o.order_id
GROUP BY month, p.payment_type
ORDER BY month, p.payment_type

```


2023-08-30 23:01:43

```

229
230 SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
231        p.payment_type,
232        COUNT(DISTINCT o.order_id) AS order_count
233 FROM `Target_Case_Study.payments` p
234 JOIN `Target_Case_Study.orders` o ON p.order_id = o.order_id
235 GROUP BY month, p.payment_type
236 ORDER BY month, p.payment_type

```

Query results

| Row | month | payment_type | order_count |
|-----|-------|--------------|-------------|
| 1 | 1 | UPI | 1715 |
| 2 | 1 | credit_card | 6093 |
| 3 | 1 | debit_card | 118 |
| 4 | 1 | voucher | 337 |
| 5 | 2 | UPI | 1723 |
| 6 | 2 | credit_card | 6582 |
| 7 | 2 | debit_card | 82 |
| 8 | 2 | voucher | 288 |
| 9 | 3 | UPI | 1942 |

Results per page: 50 1 - 50 of 50

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

```

SELECT payment_installments,
       COUNT(DISTINCT order_id) AS order_count
FROM `Target_Case_Study.payments`
GROUP BY 1
ORDER BY 1;

```

```

236 ORDER BY month, p.payment_type
237
238 SELECT payment_installments,
239        COUNT(DISTINCT order_id) AS order_count
240 FROM `Target_Case_Study.payments`
241 GROUP BY 1
242 ORDER BY 1;

```

Query results

| Row | payment_installment | order_count |
|-----|---------------------|-------------|
| 1 | 0 | 2 |
| 2 | 1 | 49060 |
| 3 | 2 | 12389 |
| 4 | 3 | 10443 |
| 5 | 4 | 7088 |
| 6 | 5 | 5234 |
| 7 | 6 | 3916 |
| 8 | 7 | 1623 |
| 9 | 8 | 4253 |

Results per page: 50 1 - 24 of 24

Actionable Insights and recommendations

The dataset contains the data between the time range of 2016-09-04 21:15 : 19 UTC to 2018-10-17 17:30:18 UTC.

Based on the geolocation city and customer count data the following three cities

| Row | geolocation_city | customer_count |
|-----|------------------|----------------|
| 1 | rio de janeiro | 1913913 |
| 2 | sao paulo | 1164470 |
| 3 | belo horizonte | 737556 |

Have the highest customer foot fall so the marketing strategies can be focused on these three major cities.

As per the analysis 2016 has lower number of orders however there is tremendous growth observed in 2017 which is 43428 and also there is good improvement in 2018 as well and the highest orders were placed in the year 2018 which is 52783.

And, also there is seasonality in the sales observed in the months of August , May, July and March where in August the highest number of orders were placed and September month has the least number of orders . so the Inventory should be planned based on this outcome.

More focus on SP RJ MG states has contributed the most.

Target should focus on providing offers to the people who are making payments in 0,1,2,3 instalments has they make the majority of the transacΘons -

Offers and easy pay opΘons using credit cards and UPI will help the customers

Focus on the area where average delivery is higher than 20 days.

Focus on more products to the areas where there is a fastest delivery which will increase the sales.

