

Project

Target- An Online and In-Store

Context

Target stands as one of the globally renowned American retail corporations that extensively offers, both online and in-store, everything from groceries and essential items to clothing and electronics. Opt for convenient and contactless options such as pickup or delivery for a seamless shopping experience. Shop with Target today for all your needs in the United States.

[Visit the online store: Expect more, Pay Less](#)

This business case encompasses data from 100,000 orders spanning the years 2016 to 2018, reflecting transactions conducted at Target in Brazil. The dataset encompasses diverse dimensions, allowing a comprehensive view of orders, encompassing aspects such as order status, pricing, payment and freight performance, customer locations, product attributes, and customer reviews.

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

- a. The data type of columns in a table
- b. The time period for which the data is given
- c. Cities and states of customers ordered during the given period

Explanation of dataset:

In this dataset, there are 8 tables given by name:

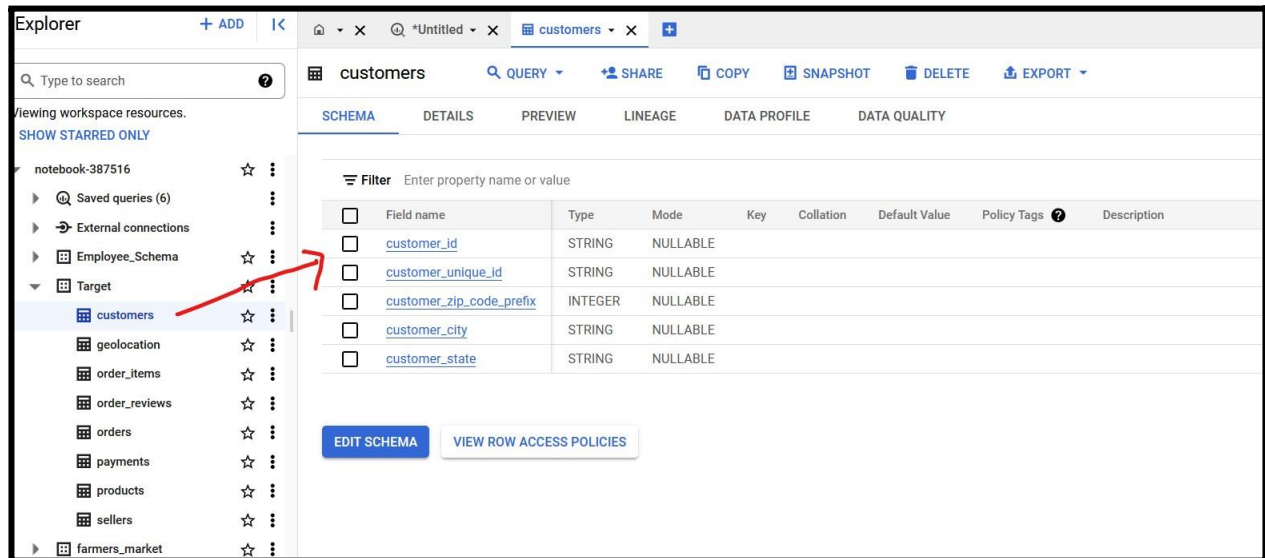
- a. customers.csv
- b. sellers.csv
- c. order_item.csv
- d. geolocation.csv
- e. payments.csv
- f. reviews.csv
- g. orders.csv
- h. products.csv

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

- a. **Explanation:** To check the data type of any table using BigQuery, Double-click on the table's name, which is mentioned on the left side under the dataset 'target'. On the right side, we can check the data types of all columns.
- b. **Screenshot of the result:**

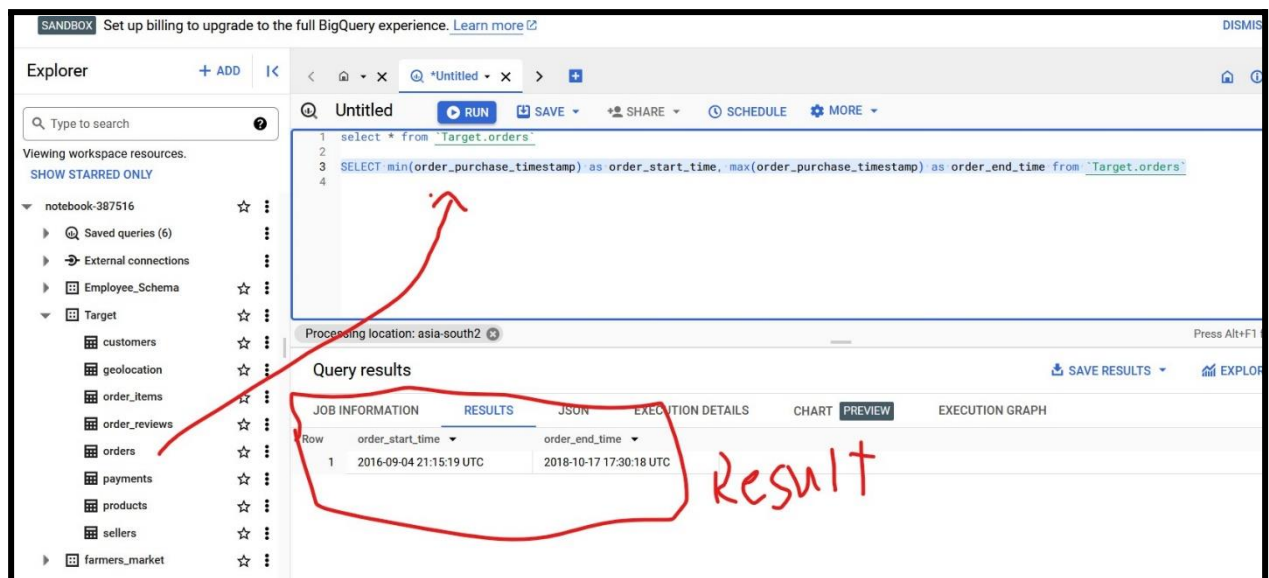
Project

Target- An Online and In-Store



2. Time range between which the orders were placed

- Explanation:** To find the time range, we will look for the table name 'order' in which we already have 'order_purchase_timestamp' col - time range here means the day and time when the first order was placed and the last day when the order was placed. Using the Avg function Max and Min will give us the time range - the ideal option when the data type is in the format of date/time.
- Analysis:** Analyzing the provided information can provide actionable insights for improving operational efficiency, enhancing customer experience, and tailoring marketing strategies to align with customer behavior patterns over time.
- Query:** `SELECT min(order_purchase_timestamp) as order_start_time, max(order_purchase_timestamp) as order_end_time from `Target.orders``
- Screenshot of the result:**



Project

Target- An Online and In-Store

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

- Explanation:** Here, we have found out all orders placed from total cities and states in the given time range, which we have above - in this query, we have used aggregate functions like count, min, max, and left join because we have to find out all orders placed by customers.
- Analysis:** The analysis based on the provided information offers insights into the geographical distribution of orders, overall order count, and temporal aspects of order placement. These insights are valuable for strategic decision-making, resource allocation, and improving customer-centric operations.
- Query:** select count(distinct c.customer_city) as city, count(distinct c.customer_state) as state, count(distinct o.order_id) as total_order, min(o.order_purchase_timestamp) as order_start_time, max(o.order_purchase_timestamp) as order_end_time from Target.customers c left join Target.orders o on c.customer_id=o.customer_id
- Screenshot of the result:**

The screenshot shows a SQL IDE interface. On the left is an 'Explorer' pane with a search bar and a list of workspace resources including 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The 'Target' schema is expanded, showing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main editor area shows a SQL query with line numbers 4 to 16. The query is:
4 SELECT min(order_purchase_timestamp) as order_start_time, max(order_purchase_timestamp) as order_end_time from 'Target.orders'
5
6
7 #Count the Cities & States of customers who ordered during the given period.
8
9
10 select count(distinct c.customer_city) as city, count(distinct c.customer_state) as state, count(distinct o.order_id) as total_order,
11
12 min(o.order_purchase_timestamp) as order_start_time, max(o.order_purchase_timestamp) as order_end_time
13
14 from Target.customers c left join Target.orders o on c.customer_id=o.customer_id
15
16
Below the query editor, it says 'Processing location: asia-south2'. The 'Query results' section is active, showing a table with 7 columns: 'Row', 'city', 'state', 'total_order', 'order_start_time', and 'order_end_time'. The first row of data shows: city: 4119, state: 27, total_order: 99441, order_start_time: 2016-09-04 21:15:19 UTC, and order_end_time: 2018-10-17 17:30:18 UTC.

Row	city	state	total_order	order_start_time	order_end_time
1	4119	27	99441	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

4. A growing trend in the no. of orders placed over the past years

- Explanation:** The growing trend here means that with time, there is an increase in total orders. Now that we know that we have to show total orders we focus on time, which is in question in years - means (2016, 2017, 2018 ...n).
- Analysis:** The analysis based on the provided query offers insights into the temporal dynamics of order volume, allowing businesses to make data-driven decisions related to inventory management, marketing planning, and overall business strategy.
- Query:** select Extract(year from order_purchase_timestamp) as years_count, count(order_id) as order_count from Target.orders group by years_count order by

Project

Target- An Online and In-Store

- years_count;
- d. Screenshot of the result:

The screenshot shows a data query interface with a sidebar on the left listing workspace resources like 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The main area displays a SQL query: `select Extract(year from order_purchase_timestamp) as years_count, count(order_id) as order_count from Target.orders group by years_count order by years_count;`. Below the query, the 'Query results' tab is active, showing a table with 3 rows of data. A red box highlights the 'Query results' tab and the data table.

Row	years_count	order_count
1	2016	329
2	2017	45101
3	2018	54011

5. Monthly seasonality in terms of the no. of orders being placed.

- a. **Explanation:** The provided information will extract information about the monthly seasonality of order placements.
- b. **Analysis:** The analysis based on the provided information will offer a detailed view of monthly seasonality regarding order counts. This information empowers businesses to adapt their strategies, optimize operations, and enhance customer experiences in alignment with the cyclical patterns observed throughout the year.
- c. **Query:** `select Extract(month from order_purchase_timestamp) as month_count, count(order_id) as order_count from Target.orders group by month_count order by month_count asc;`

- d. Screenshot of the result:

The screenshot shows a data query interface with a sidebar on the left listing workspace resources like 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The main area displays a SQL query: `select Extract(month from order_purchase_timestamp) as month_count, count(order_id) as order_count from Target.orders group by month_count order by month_count asc;`. Below the query, the 'Query results' tab is active, showing a table with 10 rows of data.

Row	month_count	order_count
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

Project

Target- An Online and In-Store

6. What time of the day do the Brazilian customers place their orders? (Dawn, Morning, Afternoon or Night)
- Explanation:** Here, we will segregate the data by the range that we can use the 'Case' condition, which will segregate the data into 4 sections: 'Dawn', 'Morning', 'Afternoon' and 'Night'; we will also count the total orders placed by the customers for each section segregation this is where group by roles come into the play. Also, we could use the where clause with the country as 'Brazil', but we assume that the complete data is of a "Brazil" country.
 - Analysis:** the analysis based on the provided query offers valuable insights into the temporal distribution of orders, allowing businesses to make data-driven decisions related to operational efficiency, resource allocation, and targeted marketing strategies aligned with customer behaviour patterns throughout the day.
 - Query:**

```
SELECT order_purchase_timestamp,
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN '0-6 hrs: Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN '7-12 hrs: Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN '13-18 hrs: Afternoon'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN '19-23 hrs: Night'
END AS order_time_of_day,
COUNT(*) AS order_count from `Target.orders`
GROUP BY order_purchase_timestamp, order_time_of_day ORDER BY order_time_of_day;
```
 - Screenshot of the result**

The screenshot displays a data analytics tool interface. On the left is an 'Explorer' panel with a search bar and a tree view of workspace resources including 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The 'Target' schema is expanded, showing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main area shows a SQL query in a text editor, which is the same query provided in the text. Below the editor, the 'Query results' section is visible, showing a table with 7 rows of data. The table has columns for 'Row', 'order_purchase_timestamp', 'order_time_of_day', and 'order_count'. All 'order_time_of_day' values are '0-6 hrs: Dawn'.

Row	order_purchase_timestamp	order_time_of_day	order_count
1	2017-12-05 01:07:58 UTC	0-6 hrs: Dawn	1
2	2017-12-05 01:07:52 UTC	0-6 hrs: Dawn	1
3	2018-01-16 01:25:39 UTC	0-6 hrs: Dawn	1
4	2017-12-07 00:02:16 UTC	0-6 hrs: Dawn	1
5	2018-03-05 03:47:11 UTC	0-6 hrs: Dawn	1
6	2017-04-17 00:10:48 UTC	0-6 hrs: Dawn	1
7	2018-05-07 01:24:47 UTC	0-6 hrs: Dawn	1

Project

Target- An Online and In-Store

7. Month-on-month number of orders placed in each state.

- Explanation:** As mentioned, we require month-by-month (1,2,3,4, n) orders placed for each state. In this, we will use the customer and order table to collect states and total order count using left join. The key point here is to extract data by month and in the "order_purchased_timestamp." The month is not available separately so that we will extract all months.
- Analysis:** The analysis is based on the provided query that offers a detailed view of order counts, allowing businesses to make informed decisions related to regional strategies, marketing campaigns, and operational efficiency based on both temporal and geographical considerations.
- Query:** Select extract(month from o.order_purchase_timestamp) as order_month, c.customer_state, count(*) as order_count from Target.orders o join Target.customers c on o.customer_id=c.customer_id group by o.order_purchase_timestamp,c.customer_state order by order_month, c.customer_state
- Screenshot of the result:**

The screenshot shows a data query interface with a sidebar on the left listing workspace resources like 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The 'Target' schema is expanded, showing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main area displays a SQL query titled 'Untitled' with the following code:

```
1 #Get the month on month no. of orders placed in each state.
2
3
4 Select extract(month from o.order_purchase_timestamp) as order_month, c.customer_state, count(*) as order_count
5
6 from Target.orders o join Target.customers c on o.customer_id=c.customer_id
7
8 group by o.order_purchase_timestamp,c.customer_state
9
10 order by order_month, c.customer_state
```

Below the query, the 'Query results' section is visible, showing a table with columns 'order_month', 'customer_state', and 'order_count'. The results are as follows:

Row	order_month	customer_state	order_count
1	1	AC	1
2	1	AC	1
3	1	AC	1
4	1	AC	1
5	1	AC	1
6	1	AC	1
7	1	AC	1
8	1	AC	1
9	1	AL	1

8. Customers are distributed across all the states.

- Explanation:** Customer distribution across all the states means we have to find and count the total unique customers for each state.
- Analysis:** The analysis based on the provided query offers a comprehensive view of customer distribution across states, enabling businesses to make data-driven decisions for market expansion, customer engagement, and strategic planning.
- Query:** SELECT distinct customer_state, COUNT(customer_unique_id) AS toal_customer FROM Target.customers GROUP BY customer_state ORDER BY customer_state;
- Screenshot of the result:**

Project

Target- An Online and In-Store

The screenshot shows a data analytics tool interface. On the left is a sidebar with a search bar and a list of workspace resources including 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The 'Target' schema is expanded, showing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main area displays a SQL query titled 'Untitled' with the following code:

```

1 #How are the customers distributed across all the states?
2 select * from 'Target.customers'
3
4 SELECT distinct customer_state, COUNT(customer_unique_id) AS toal_customer
5 FROM Target.customers
6 GROUP BY customer_state
7 ORDER BY customer_state;

```

Below the query editor, the 'Query results' section is active, showing a table with two columns: 'customer_state' and 'toal_customer'. The results are as follows:

Row	customer_state	toal_customer
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020

9. Getting the % increase in the cost of orders from 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.
 - a. **Explanation:** The provided information will calculate the percentage increase in the cost of orders from 2017 to 2018, considering the months from January to August only. The analysis is based on the "payment_value" column in the "payments" table.
 - b. **Analysis:** The analysis offers insights into the percentage increase in the cost of orders from 2017 to 2018, emphasizing the months between January and August. It is a valuable metric for evaluating financial growth and identifying potential areas for further investigation or improvement.
 - c. **Query:** With MonthlyPayments as (select o.order_id, Extract(year from o.order_purchase_timestamp) as order_year, Extract(Month from o.order_purchase_timestamp) as order_month, Sum(P.payment_value) as total_payment from Target.orders o Join Target.payments p on o.order_id=p.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 o.order_id, extract (year from o.order_purchase_timestamp), extract(month from o.order_purchase_timestamp)) SELECT (SUM(CASE WHEN order_year = 2018 THEN total_payment ELSE 0 END) - SUM(CASE WHEN order_year = 2017 THEN total_payment ELSE 0 END)) / SUM(CASE WHEN order_year = 2017 THEN total_payment ELSE 0 END) 100 AS percentage_increase FROM MonthlyPayments WHERE order_year IN (2017, 2018) GROUP BY order_year;
10. Calculating the Total & Average value of the order price for each state.
 - a. **Explanation:** In this problem, there will be three tables will be engaged which are 'Customers' from which we will get each state, 'Orders' orders table will give us the order_id for which we will find out the total sum and avg of the price from the table 'Payment.'

Project

Target- An Online and In-Store

- b. **Analysis:** The information provides valuable insights into the finding patterns of customers in different states, helping businesses make informed decisions related to marketing, operations, and customer engagement strategies.
- c. **Query:** SELECT c.customer_state AS State, SUM(p.payment_value) AS TotalOrderPrice, AVG(p.payment_value) AS AverageOrderPrice FROM Target.customers c JOIN Target.orders o ON c.customer_id = o.customer_id JOIN Target.payments p o.order_id = p.order_id GROUP BY c.customer_state ORDER BY c.customer_state;
- d. **Screenshot of the result**

The screenshot shows a data analytics tool interface. On the left is a sidebar with a search bar and a list of workspace resources including 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and 'Target'. The 'Target' database is expanded, showing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main area displays a SQL query titled 'Untitled' with the following code:

```

1 #Calculate the Total & Average value of order price for each state.
2
3 SELECT c.customer_state AS State, SUM(p.payment_value) AS TotalOrderPrice, AVG(p.payment_value) AS AverageOrderPrice
4 FROM Target.customers c
5 JOIN Target.orders o ON c.customer_id = o.customer_id
6 JOIN Target.payments p ON o.order_id = p.order_id
7 GROUP BY c.customer_state
8 ORDER BY c.customer_state;

```

Below the query, it indicates the 'Processing location: asia-south2'. The 'Query results' section shows a table with 8 rows of data. The table has columns for 'Row', 'State', 'TotalOrderPrice', and 'AverageOrderPrice'. The results are as follows:

Row	State	TotalOrderPrice	AverageOrderPrice
1	AC	19680.619999999...	234.2930952380...
2	AL	96962.059999999...	227.0774238875...
3	AM	27966.929999999...	181.6034415584...
4	AP	16262.799999999...	232.3257142857...
5	BA	616645.820000000...	170.8160166204...
6	CE	279464.029999999...	199.90273962804
7	DF	355141.080000000...	161.1347912885...
8	ES	325967.550000000...	154.7069530137...

Conclusion for "Target" - An E-commerce and In-store American Corporation:

Based on the provided analysis, queries, and explanations, several conclusions can be drawn regarding Target, an American e-commerce and in-store corporation:

1. **Geographical distribution and market presence:** Target is a recognised brand and leading retailer in the United States, maintaining a significant online and physical market presence. To reach maximum users in local markets, 'target' can customise products depending on the geo-location. For example, the company could target students in schools and colleges if they have a range of amazing t-shirts, school/college bags, shoes, etc.
2. **Order dynamics and seasonal trends:** The Company comprehensively understands its order dynamics, utilizing advanced data analysis techniques to identify trends and patterns in customer behaviour. Seasonal trends, monthly seasonality, and yearly growth are analysed to optimize operations and marketing strategies.
3. **Customer engagement and regional variation:** Target employs customer-centric strategies, tailoring marketing efforts based on regional customer engagement patterns. Analysing customer distribution across states provides insights into regional variations in customer behavior.

Project

Target- An Online and In-Store

4. **Operational efficiency and resource allocation:** The Company is committed to operational efficiency, utilizing data analysis to optimise resource allocation, staffing levels, and inventory management. The segmentation of orders by time of day allows for strategic planning and operational optimization during peak hours.

Project

Target- An Online and In-Store

Project

Target- An Online and In-Store

Q.4. Answer.c

Query

```
select c.customer_state, sum(od.freight_value) as sum_freight_value, avg(od.freight_value) as
total_freight_avg from Target.customers c
join Target.orders o on c.customer_id=o.customer_id
join Target.order_items od on o.order_id=od.order_id
group by customer_state
```

Explanation:

Just like the above question, in this problems also, we will use three tables 'Customers', 'Order_items' and 'Orders' - Freight_value can be calculated using the table order_items although to connect with order_items we will use 'order_id' which is a common col for both tables "order_items" and 'Orders' and result for each state 'Customer table' we will use group by.

Screenshot

The screenshot displays a data analytics tool interface. On the left is a sidebar with a search bar and a list of workspace resources, including 'notebook-387516', 'Saved queries (6)', 'External connections', 'Employee_Schema', and a 'Target' database containing tables like 'customers', 'geolocation', 'order_items', 'order_reviews', 'orders', 'payments', 'products', 'sellers', and 'farmers_market'. The main area shows a SQL query in a text editor, titled 'Untitled', with a 'RUN' button. The query is:
1 #Calculate the Total & Average value of order freight for each state.
2
3 select c.customer_state, sum(od.freight_value) as sum_freight_value, avg(od.freight_value) as total_freight_avg from Target.customers c
4 join Target.orders o on c.customer_id=o.customer_id
5 join Target.order_items od on o.order_id=od.order_id
6 group by customer_state
7
Below the query editor, it indicates the 'Processing location: asia-south2'. The 'Query results' section shows a table with columns 'customer_state', 'sum_freight_value', and 'total_freight_avg'. The results are displayed in a table with 9 rows, corresponding to different states: RN, CE, RS, SC, SP, MG, BA, RJ, and GO. The bottom right corner shows 'Results per page: 50' and '1 - 27 of 27'.

Row	customer_state	sum_freight_value	total_freight_avg
1	RN	18860.09999999...	35.65236294896...
2	CE	48351.58999999...	32.71420162381...
3	RS	135522.7400000...	21.73580433039...
4	SC	89660.26000000...	21.47036877394...
5	SP	718723.0699999...	15.14727539041...
6	MG	270853.4600000...	20.63016680630...
7	BA	100156.6799999...	26.36395893656...
8	RJ	305589.3100000...	20.96092393168...
9	GO	53114.97999999...	22.76681525932...

Project

Target- An Online and In-Store

Q.5. answer.1.

Query:

```
select order_id, (order_estimated_delivery_date-order_purchase_timestamp) as
delivery_time,
(order_estimated_delivery_date - order_delivered_customer_date) as
diffe_estimated_delivery
from `Target.orders`;
```

Screenshot

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

```
1 #Find the no. of days taken to deliver each order from the order's purchase date as delivery time.Also, calculate
2 estimated & actual delivery date of an order.Do this in a single query.
3 select order_id, (order_estimated_delivery_date-order_purchase_timestamp) as delivery_time,
4 (order_estimated_delivery_date - order_delivered_customer_date) as diffe_estimated_delivery
5 from `Target.orders`;
6
```

The query results are displayed in a table with the following columns: Row, order_id, delivery_time, and diffe_estimated_delivery. The results show 10 rows of data.

Row	order_id	delivery_time	diffe_estimated_delivery
1	7a4df5d8cff4090e541401a20a...	0-0 0 396:49:27	null
2	35de4050331c6c644cddc86f4...	0-0 0 814:52:2	null
3	b5359909123fa03c50bdb0cfe...	0-0 0 886:52:8	null
4	dba5062fbd3af4fb6c33b1e04...	0-0 0 606:38:56	null
5	90ab3e7d52544ec7bc3363c82...	0-0 0 586:47:26	null
6	fa65dad1b0e818e3ccc5cb0e3...	0-0 0 659:14:26	null
7	1df2775799eecd9dd8502425...	0-0 0 756:56:55	null
8	6190a94657e1012983a274b8...	0-0 0 802:23:30	null
9	58ce513a55c740a3a81e8c8b7...	0-0 0 365:54:53	null
10	088683f795a3d30bfd61152c4f...	0-0 0 757:57:13	null

Q.5. Answer.b.

Query

```
SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight_value
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
GROUP BY c.customer_state
ORDER BY avg_freight_value DESC
LIMIT 5

UNION ALL
```

Project

Target- An Online and In-Store

```
SELECT c.customer_state,AVG(oi.freight_value) AS avg_freight_value
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
GROUP BY c.customer_state
ORDER BY avg_freight_value ASC
LIMIT 5;
```

This query gave me error everytime -

Q.5.c Query

```
SELECT
    c.customer_state,
    AVG(DATEDIFF(order_delivered_customer_date, order_purchase_timestamp)) AS
avg_delivery_time
FROM
    customers AS c
JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_delivery_time DESC
LIMIT 5
```

UNION ALL

```
SELECT
    c.customer_state,
    AVG(DATEDIFF(order_delivered_customer_date, order_purchase_timestamp)) AS
avg_delivery_time
FROM
    customers AS c
JOIN
    orders AS o
ON
```

Project

Target- An Online and In-Store

```
c.customer_id = o.customer_id  
GROUP BY  
    c.customer_state  
ORDER BY  
    avg_delivery_time ASC  
LIMIT 5;
```

Q.4. Answer.c.

Query

```
SELECT c.customer_state,  
       AVG(DATE_DIFF(o.order_delivered_customer_date, o.Order_Estimated_Delivery_Date,  
DAY)) AS AvgDeliveryDifference  
FROM `Target.customers` as c  
JOIN `Target.orders` as o ON c.Customer_ID = o.Customer_ID  
GROUP BY c.customer_state  
ORDER BY AvgDeliveryDifference ASC  
LIMIT 5;
```

Screenshot

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

```
1 SELECT c.customer_state,  
2       AVG(DATE_DIFF(o.order_delivered_customer_date, o.Order_Estimated_Delivery_Date, DAY)) AS AvgDeliveryDifference  
3 FROM `Target.customers` as c  
4 JOIN `Target.orders` as o ON c.Customer_ID = o.Customer_ID  
5 GROUP BY c.customer_state  
6 ORDER BY AvgDeliveryDifference ASC  
7 LIMIT 5;  
8
```

The query results are displayed in a table with the following data:

Row	customer_state	AvgDeliveryDifference
1	AC	-19.7625
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

Q.5. Answer.1.

Query

```
SELECT  
    EXTRACT(YEAR FROM o.Order_purchase_timestamp) AS Year, EXTRACT(MONTH FROM  
o.Order_purchase_timestamp) AS Month,  
    p.Payment_Type, COUNT(o.Order_ID) AS NumberOfOrders  
FROM `Target.orders` o  
JOIN `Target.payments` as p
```


Project

Target- An Online and In-Store

ON o.Order_ID = p.Order_ID
GROUP BY Year, Month, Payment_Type
ORDER BY Year, Month, Payment_Type;

Explanation

As asked in the question, we will use 'extract' years and month from the table orders and in order to find payment type we will use 'Payment' table and use the inner join to connect both tables using 'order_id' because month we have to find on month no. of orders placed.

Screenshot

The screenshot shows a SQL query editor with the following query:

```
1 SELECT
2   EXTRACT(YEAR FROM o.Order_purchase_timestamp) AS Year, EXTRACT(MONTH FROM o.Order_purchase_timestamp) AS Month,
3   p.Payment_Type, COUNT(o.Order_ID) AS NumberOfOrders
4 FROM `Target.orders` o
5 JOIN `Target.payments` as p
6 ON o.Order_ID = p.Order_ID
7 GROUP BY Year, Month, Payment_Type
8 ORDER BY Year, Month, Payment_Type;
```

The query results are displayed in a table with the following columns: Row, Year, Month, Payment_Type, and NumberOfOrders.

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

Question.5.Answer.2

Query

```
SELECT p.payment_installments,COUNT(o.Order_ID) AS NumberOfOrders FROM
`Target.orders` as o LEFT JOIN `Target.payments` p ON o.Order_ID = p.Order_ID
GROUP BY p.payment_installments
ORDER BY p.payment_installments
```

Screenshot

The screenshot shows a SQL query editor with the following query:

```
1 #Find the no. of orders placed on the basis of the payment installments that have been paid.
2
3 SELECT p.payment_installments,COUNT(o.Order_ID) AS NumberOfOrders FROM `Target.orders` as o LEFT JOIN `Target.payments` p ON o.Order_ID = p.Order_ID
4 GROUP BY p.payment_installments
5 ORDER BY p.payment_installments
```

The query results are displayed in a table with the following columns: Row, payment_installment, and NumberOfOrders.

Row	payment_installment	NumberOfOrders
1	null	1
2	0	2
3	1	52546
4	2	12413
5	3	10461
6	4	7098
7	5	5239
8	6	3920
9	7	1626
10	8	4268

Project

Target- An Online and In-Store

Valuable Insights

- ☐ Here we have the range of all orders placed and purchased, this customized data can help 'Target' to understand the historical time of business operations. Also, this data will also help understanding in what hours the company should be proactive when it comes to showcase customized products to the users of brazil.
- ☐ From this result, we could do extensive research on how many users from all cities are purchasing and how the purchasing values of users could be enhanced.
- ☐ These timestamps can be used to calculate various performance metrics, such as the average order frequency or the total number of orders within the specified time frame.