

--

Q1 Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

--Q1.1 Data type of columns in a table

```
select column_name,data_type
from scaler-dsml-sql-378016.Target.INFORMATION_SCHEMA.COLUMNS;
```

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar shows the project name 'scaler-dsml-sql-378016'. The left sidebar contains the 'Explorer' panel with a search bar and a list of resources, including 'Target'. The main area shows a SQL query editor with the following code:

```
--Q1 Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
--Q1.1 Data type of columns in a table
select column_name,data_type
from scaler-dsml-sql-378016.Target.INFORMATION_SCHEMA.COLUMNS;
```

Below the query editor, the 'Query results' section is visible, showing a table with 9 rows and 2 columns: 'column\_name' and 'data\_type'. The results are as follows:

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64
8	seller_id	STRING
9	seller_zip_code_prefix	INT64

The bottom of the screen shows a Windows taskbar with the date '27-03-2023' and time '01:39'.

--Q1.2. Time period for which the data is give

SELECT

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

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

FROM Target.orders;

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar shows the Google Cloud logo and the project name 'Scaler-DSML-SQL'. The left sidebar contains the 'Explorer' panel with a search bar and a list of resources, including 'Target' and its associated tables like 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main area shows a SQL query editor with the following code:

```
--Q1.2. Time period for which the data is give
SELECT
MIN(order_purchase_timestamp) AS start_time,
MAX(order_purchase_timestamp) AS end_time
FROM Target.orders;
```

The query has been executed successfully, as indicated by the 'Query completed.' status. The 'Query results' section shows a table with two columns: 'start\_time' and 'end\_time'. The results are as follows:

Row	start_time	end_time
1	2016-08-04 21:15:19 UTC	2016-10-17 17:30:16 UTC

The bottom of the screen shows the Windows taskbar with the date and time '27-03-2023' and '01:40'.

--Q1.3. Cities and States of customers ordered during the given period

select

customer\_city, customer\_state

from Target.customers;

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the project name 'Scaler-DSML-SQL', and a search bar. The left sidebar shows the 'Explorer' view with a tree structure of resources, including 'Target' and its sub-resources like 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main panel shows a query execution result for a query named 'Q1.3: Cities and States of customers ordered during the given period'. The query is displayed in a code editor, and the results are shown in a table format. The table has two columns: 'customer\_city' and 'customer\_state'. The results are paginated, showing 1 to 50 of 99441 results. The bottom status bar shows the system time as 01:42 on 27-03-2023.

Query results

Row	customer_city	customer_state
1	acu	RN
2	acu	RN
3	acu	RN
4	ico	CE
5	ico	CE
6	ico	CE
7	ico	CE
8	ico	CE
9	ico	CE

--Q2.1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

```
SELECT *,
ROUND((No_of_order-
LAG(No_of_order)OVER(ORDER BY quarter))/LAG(No_of_order)OVER(ORDER BY quarter)*100,2)percent
change_in_order
FROM (
SELECT
EXTRACT(quarter FROM order_purchase_timestamp) quarter,
COUNT(order_id) No_of_order
FROM Target.orders
GROUP BY quarter)
ORDER BY quarter;
```

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

```
20 --Q2.1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific
21 months?
22 SELECT *,
23 ROUND((No_of_order-LAG(No_of_order)OVER(ORDER BY quarter))/LAG(No_of_order)OVER(ORDER BY quarter)*100,2)percent_change_in_order
24 FROM (
25 SELECT
26 EXTRACT(quarter FROM order_purchase_timestamp) quarter,
27 COUNT(order_id) No_of_order
28 FROM Target.orders
29 GROUP BY quarter)
30 ORDER BY quarter;
```

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

Row	quarter	No_of_order	percent_change
1	1	26470	NaN
2	2	29028	10.8
3	3	25466	-13.17
4	4	18177	-28.62

--growing trend from Q1 to Q2 . But after Q2 significant drop in no of orders .

--Q2.1.2(Month wise)

```
SELECT *,
ROUND((No_of_order-
LAG(No_of_order)OVER(ORDER BY month))/LAG(No_of_order)OVER(ORDER BY month)*100,2)percent_ch
ange_in_order
FROM (
SELECT
EXTRACT(month FROM order_purchase_timestamp) month,
COUNT(order_id) No_of_order
FROM Target.orders
GROUP BY month)
ORDER BY month;
```

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

```
--Q2.1.2(Month wise)
SELECT *,
ROUND((No_of_order-
LAG(No_of_order)OVER(ORDER BY month))/LAG(No_of_order)OVER(ORDER BY month)*100,2)percent_change_in_order
FROM (
SELECT
EXTRACT(month FROM order_purchase_timestamp) month,
COUNT(order_id) No_of_order
FROM Target.orders
GROUP BY month)
ORDER BY month;
```

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

Row	month	No_of_order	percent_change
1	1	8069	NaN
2	2	8508	5.44
3	3	9893	16.28
4	4	9343	-5.56
5	5	10572	13.16

The interface also shows a sidebar with project resources, including 'Em', 'Northwind', and 'Target'. The 'Target' project is selected, showing a list of tables: 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'.

--AUGUST has highest no of orders and by july.

--Q2.1.3 (year wise)

```
SELECT *,
ROUND((No_of_order-
LAG(No_of_order)OVER(ORDER BY year))/LAG(No_of_order)OVER(ORDER BY year)*100,2)percent_chan
ge_in_order
FROM (
SELECT
EXTRACT(year FROM order_purchase_timestamp) year,
COUNT(order_id) No_of_order
FROM Target.orders
GROUP BY year)
ORDER BY year;
```

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, a search bar, and various icons for project management and billing. The left sidebar shows the 'Explorer' view with a tree structure of resources, including 'Target' and its associated tables like 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main area shows a SQL query being executed. The query is as follows:

```
--Q2.1.3 (year wise)
SELECT *,
ROUND((No_of_order-
LAG(No_of_order)OVER(ORDER BY year))/LAG(No_of_order)OVER(ORDER BY year)*100,2)percent_change_in_order
FROM (
SELECT
EXTRACT(year FROM order_purchase_timestamp) year,
COUNT(order_id) No_of_order
FROM Target.orders
GROUP BY year)
ORDER BY year;
```

The query results are displayed in a table with the following columns: Row, year, No\_of\_order, and percent\_change. The results are as follows:

Row	year	No_of_order	percent_change
1	2016	329	NA
2	2017	45101	13608.51
3	2018	54011	19.76

The bottom of the screen shows the Windows taskbar with the date and time as 27-03-2023, 01:45.



--Q2.2 What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
select Period,count(o.order_id) No_of_order
from Target.orders o
join
    ( select order_id,
      case when extract (hour from order_purchase_timestamp) between 6 and 12 then "Morning"
    "
      when extract (hour from order_purchase_timestamp) between 12 and 18 then "Afternoon"
      when extract (hour from order_purchase_timestamp) between 18 and 4 then "night"
      else "Dawn"
      end Period
      from Target.orders) X
on o.order_id = X.order_id
group by Period;
```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the SQL query for Q2.2. The query results are displayed in a table below the editor. The table has three columns: Row, Period, and No\_of\_order. The results show that the majority of purchases occur in the Afternoon (38136 orders), followed by Dawn (33071 orders) and Morning (28235 orders).

Row	Period	No_of_order
1	Morning	28235
2	Dawn	33071
3	Afternoon	38136

--Brazilian most preferably like to buy in afternoon.

--Q3. Evolution of E-commerce orders in the Brazil region:  
--Q3.1. Get month on month orders by states

```
select * from
(
  SELECT
  customer_state,
  COUNT(o.order_id) No_of_order,
  extract(month from order_purchase_timestamp) month
FROM Target.orders o
JOIN Target.customers c
ON o.customer_id =c.customer_id
GROUP BY customer_state,month
)
ORDER BY customer_state,month;
```

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL query:

```
--Q3. Evolution of E-commerce orders in the Brazil region:
--Q3.1. Get month on month orders by states

select * from
(
  SELECT
  customer_state,
  COUNT(o.order_id) No_of_order,
  extract(month from order_purchase_timestamp) month
FROM Target.orders o
JOIN Target.customers c
ON o.customer_id =c.customer_id
GROUP BY customer_state,month
)
ORDER BY customer_state,month;
```

The query results are displayed in a table with the following columns: Row, customer\_state, No\_of\_order, and month. The results show two rows for the state 'AC'.

Row	customer_state	No_of_order	month
1	AC	8	1
2	AC	6	2

The interface also includes an Explorer panel on the left showing the project structure, a top navigation bar with Google Cloud logo and search, and a bottom status bar with system information.



--Q3.2. Distribution of customers across the states in Brazil

```
SELECT
customer_state,
COUNT(o.customer_id) No_of_customers
FROM Target.orders o
JOIN Target.customers c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY No_of_customers DESC;
```

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the project name 'Scaler-DSML-SQL', and a search bar. The left sidebar shows the 'Explorer' panel with a tree view of resources, including 'Em', 'Northwind', 'Target', 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main panel shows a SQL query being executed. The query is as follows:

```
--Q3.2. Distribution of customers across the states in Brazil
SELECT
customer_state,
COUNT(o.customer_id) No_of_customers
FROM Target.orders o
JOIN Target.customers c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY No_of_customers DESC;
```

The query results are displayed in 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

The bottom of the screen shows the Windows taskbar with the date and time as 27-03-2023, 01:48.

--Q4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

--

4.1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment\_value" column in payments table

```
SELECT
  (sum_payments-
LEAD(sum_payments)OVER(ORDER BY year DESC))/(sum_payments)*100 per_change_cost_order
FROM
  (
    SELECT
      EXTRACT(year FROM order_purchase_timestamp) year,
      SUM(payment_value) sum_payments,
      FROM Target.payments p
      JOIN Target.orders o
      ON p.order_id =o.order_id
      WHERE
        (EXTRACT(year FROM order_purchase_timestamp) BETWEEN 2017 AND 2018)
      AND
        (EXTRACT(month FROM order_purchase_timestamp) BETWEEN 1 AND 8)
      GROUP BY EXTRACT(year FROM order_purchase_timestamp) )
    ORDER BY per_change_cost_order DESC
    LIMIT 1;
```

The screenshot displays the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane shows a project named 'scaler-dsml-sql-378016' with a folder 'Target' containing tables like 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main editor shows a SQL query that calculates the percentage change in the sum of payment values from 2017 to 2018, specifically for months 1 through 8. The query uses a window function LEAD to compare the current year's sum with the previous year's sum. Below the query editor, the 'Query results' section is visible, showing a single row with the value 57.8017891... under the column 'per\_change\_cost\_order'. The bottom of the screen shows a Windows taskbar with the date 27-03-2023 and time 01:49.

--Q4.2. Mean & Sum of price and freight value by customer state

SELECT

```
customer_state,  
round(sum(price),2) Sum_price ,  
round(Avg(price),2) Avg_price,  
round(sum(freight_value),2) Sum_freight_value,  
round(Avg(freight_value),2) Avg_freight_value
```

FROM Target.orders o

JOIN Target.customers c

ON o.customer\_id = c.customer\_id

JOIN Target.order\_items ot

ON o.order\_id = ot.order\_id

group by customer\_state

order by Sum\_price desc;

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, a search bar, and various icons for settings and notifications. The left sidebar shows the 'Explorer' panel with a tree view of resources, including 'Target' and its sub-resources like 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main panel shows a SQL query being executed. The query is as follows:

```
--Q4.2. Mean & Sum of price and freight value by customer state  
  
SELECT  
  customer_state,  
  round(sum(price),2) Sum_price ,  
  round(Avg(price),2) Avg_price,  
  round(sum(freight_value),2) Sum_freight_value,  
  round(Avg(freight_value),2) Avg_freight_value  
FROM Target.orders o  
JOIN Target.customers c  
ON o.customer_id = c.customer_id  
  
JOIN Target.order_items ot  
ON o.order_id = ot.order_id  
  
group by customer_state  
order by Sum_price desc;
```

The query results are displayed in a table with the following columns: Row, customer\_state, Sum\_price, Avg\_price, Sum\_freight\_value, and Avg\_freight\_value. The results are as follows:

Row	customer_state	Sum_price	Avg_price	Sum_freight_value	Avg_freight_value
1	SP	5202955.05	109.65	710723.07	15.15
2	RJ	1824092.67	125.12	305589.31	20.66

The bottom of the screen shows the Windows taskbar with the date and time as 27-03-2023, 01:50.

--Q5. Analysis on sales, freight and delivery time  
--Q5.1. Calculate days between purchasing, delivering and estimated delivery

```
SELECT
  date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) diff_estimated_delivery,
  date_diff(order_delivered_customer_date,order_purchase_timestamp,day) time_to_delivery
FROM Target.orders
ORDER BY diff_estimated_delivery desc;
```

The screenshot displays the Google Cloud BigQuery console interface. The left sidebar shows the Explorer view with a tree structure of resources, including a project named 'scaler-dsml-sql-378016'. The main editor area contains a SQL query that has been executed. The query results are displayed in a table format with columns 'diff\_estimated\_delivery' and 'time\_to\_delivery'. The results show five rows of data, with the first row having values 155 and 20. The console also shows the query execution details and a 'Query completed' status.

```
148 group by customer_state
149 order by Sum_price desc;
150
151
152 --Q5. Analysis on sales, freight and delivery time
153 --Q5.1. Calculate days between purchasing, delivering and estimated delivery
154
155 SELECT
156   date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) diff_estimated_delivery,
157   date_diff(order_delivered_customer_date,order_purchase_timestamp,day) time_to_delivery
158 FROM Target.orders
159 ORDER BY diff_estimated_delivery desc;
160
```

Row	diff_estimated_delivery	time_to_delivery
1	155	20
2	149	9
3	146	6
4	144	not
5	144	not

```
--Q5.2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
--o time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
--o diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date
```

SELECT

```
DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,day) Days_bw_pur_estdeli,
```

```
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,day) Days_bw_pur_cus_de
```

```
FROM Target.orders
```

```
WHERE order_status != "canceled"
```

```
ORDER BY Days_bw_pur_cus_de DESC;
```

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

```
--Q5.2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
--o time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
--o diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

SELECT
DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,day) Days_bw_pur_estdeli,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,day) Days_bw_pur_cus_de
FROM Target.orders
WHERE order_status != "canceled"
ORDER BY Days_bw_pur_cus_de DESC;
```

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

Row	Days_bw_pur_cus_de	Days_bw_pur_estdeli
1	28	209
2	19	208
3	30	195
4	32	194
5	28	194

The interface also shows a sidebar with project explorer, a top navigation bar with Google Cloud logo, and a bottom status bar with system information.

--

Q5.3. Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery

SELECT

```
customer_state,
AVG(freight_value) mean_freight_value,
AVG(date_diff (order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery,
AVG(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) time_to_delivery
FROM Target.orders o
JOIN Target.order_items oi ON
o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state;
```

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

```
-- Q5.3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery
SELECT
  customer_state,
  AVG(freight_value) mean_freight_value,
  AVG(date_diff (order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery,
  AVG(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) time_to_delivery
FROM Target.orders o
JOIN Target.order_items oi ON
o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state;
```

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

Row	customer_state	mean_freight_value	diff_estimated_delivery	time_to_delivery
1	MT	28.1662943...	13.6393442...	-17.5081967...
2	MA	38.2570024...	9.10999999...	21.2037500...
3	AL	35.8436711...	7.97656079...	23.9929742...
4	SP	15.1472753...	10.2555943...	-8.25960855...

The interface also shows a sidebar with project navigation, a top navigation bar with search and settings, and a bottom status bar with system information.



--Q5.4. Sort the data to get the following:

SELECT

```
customer_state,  
AVG(freight_value) mean_freight_value,  
AVG(date_diff (order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery,  
AVG(date_diff(order_purchase_timestamp,order_delivered_customer_date,day)) time_to_delivery  
FROM Target.orders o  
JOIN Target.order_items oi ON  
o.order_id =oi.order_id  
JOIN Target.customers c  
ON c.customer_id =o.customer_id  
GROUP BY customer_state;
```

The screenshot shows the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, a search bar, and various icons. The left sidebar contains the 'Explorer' panel with a search bar and a list of resources including 'scaler-dsml-sql-378016', 'External connections', 'Saved queries (4)', and 'Project queries'. The main area displays a SQL query in the editor, which is the same query as provided in the text. Below the editor, the 'Query results' section is visible, showing a table with 4 rows and 5 columns: 'customer\_state', 'mean\_freight\_value', 'diff\_estimated\_delivery', and 'time\_to\_delivery'. The results are sorted by 'customer\_state'.

Row	customer_state	mean_freight_value	diff_estimated_delivery	time_to_delivery
1	MT	28.1662943...	13.6393442...	-17.5081967...
2	MA	38.2570024...	9.10999999...	-21.2037500...
3	AL	35.8436711...	7.97656079...	-23.9929742...
4	SP	15.1472753...	10.2555943...	-8.25960855...

--Q5.5 Top 5 states with highest average freight value - sort in desc/asc limit 5

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

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL query:

```
--Q5.5 Top 5 states with highest average freight value - sort in desc/asc limit 5
SELECT
  customer_state,
  AVG(freight_value) mean_freight_value
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY mean_freight_value DESC
LIMIT 5;
```

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

Row	customer_state	mean_freight_value
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733696...
5	PI	38.1479704...

The interface also shows a sidebar with project navigation, a top navigation bar with Google Cloud logo and search, and a bottom status bar with system information.

--Q5.5 Top 5 states with lowest average freight value - sort in desc/asc limit 5

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

The screenshot shows the Google Cloud BigQuery console interface. The query editor on the right contains the following SQL query:

```
--Q5.5 Top 5 states with lowest average freight value - sort in desc/asc limit 5
SELECT
  customer_state,
  AVG(freight_value) mean_freight_value
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY mean_freight_value ASC
LIMIT 5;
```

The query results are displayed in a table below the editor:

Row	customer_state	mean_freight_value
1	SP	15.1472753...
2	PR	20.5316515...
3	MD	20.6301668...
4	RJ	20.9609239...
5	DE	21.0413549...

The interface also shows a sidebar with project explorer, a top navigation bar with search and settings, and a bottom status bar with system information.

--Q5.6. Top 5 states with highest average time to delivery

```
SELECT
customer_state,
AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) Avg_time_to_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY Avg_time_to_delivery desc
LIMIT 5;
```

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

```
--Q5.6. Top 5 states with highest average time to delivery
SELECT
customer_state,
AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) Avg_time_to_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY Avg_time_to_delivery desc
LIMIT 5;
```

Below the query editor, the 'Query results' section displays a table with 5 rows of data. The table has two columns: 'customer\_state' and 'Avg\_time\_to\_delivery'.

Row	customer_state	Avg_time_to_delivery
1	RR	27.8260859...
2	AP	27.7530854...
3	AM	25.9631901...
4	AL	23.9929742...
5	PA	23.3017077...

The interface also includes a left sidebar with the 'Explorer' panel showing the project hierarchy, and a top navigation bar with various tools and settings.

--Q5.6. Top 5 states with lowest average time to delivery

```
SELECT
customer_state,
AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) Avg_time_to_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY Avg_time_to_delivery asc
LIMIT 5;
```

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

```
250 LIMIT 5;
251 --Q5.6. Top 5 states with lowest average time to delivery
252
253 SELECT
254 customer_state,
255 AVG(date_diff(order_delivered_customer_date,order_purchase_timestamp,day)) Avg_time_to_delivery
256 FROM Target.orders o
257 JOIN Target.order_items oi
258 ON o.order_id =oi.order_id
259 JOIN Target.customers c
260 ON c.customer_id =o.customer_id
261 GROUP BY customer_state
262 ORDER BY Avg_time_to_delivery asc
263 LIMIT 5;
```

Below the query editor, the 'Query results' section is visible, showing a table with 5 rows of data. The table has two columns: 'customer\_state' and 'Avg\_time\_to\_delivery'.

Row	customer_state	Avg_time_to_delivery
1	SP	8.25960855...
2	PR	11.4807930...
3	MD	11.6155221...
4	DE	12.5014861...
5	SC	14.5209856...

The interface also includes a left sidebar with the 'Explorer' panel showing the project structure, and a top navigation bar with various icons and a search bar.

--Q5.7. Top 5 states where delivery is really fast compared to estimated date

SELECT

customer\_state,

AVG(date\_diff(order\_estimated\_delivery\_date,order\_delivered\_customer\_date,day)) diff\_estimated\_delivery

FROM Target.orders o

JOIN Target.order\_items oi

ON o.order\_id =oi.order\_id

JOIN Target.customers c

ON c.customer\_id =o.customer\_id

GROUP BY customer\_state

ORDER BY diff\_estimated\_delivery desc

LIMIT 5;

The screenshot displays the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the project name 'Scaler-DSML-SQL', and a search bar. Below the navigation bar, the 'Explorer' panel on the left shows a tree view of the project's resources, including 'Em', 'Northwind', 'Target', 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main query editor area contains the following SQL query:

```
--Q5.7. Top 5 states where delivery is really fast compared to estimated date
SELECT
customer_state,
AVG(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY diff_estimated_delivery desc
LIMIT 5;
```

The 'Query results' panel at the bottom shows the output of the query. It includes a table with 5 rows and 2 columns: 'customer\_state' and 'diff\_estimated\_delivery'. The results are as follows:

Row	customer_state	diff_estimated_delivery
1	AC	20.0109990...
2	RO	19.0806560...
3	AM	18.9754601...
4	AP	17.4444444...
5	RR	17.4347826...

The bottom of the screen shows a Windows taskbar with the date and time '27-03-2023 01:58'.



--Q5.7. Top 5 states where delivery is really not so fast compared to estimated date

```
SELECT
customer_state,
AVG(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY diff_estimated_delivery asc
LIMIT 5;
```

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

```
--Q5.7. Top 5 states where delivery is really not so fast compared to estimated date
SELECT
customer_state,
AVG(date_diff(order_estimated_delivery_date,order_delivered_customer_date,day)) diff_estimated_delivery
FROM Target.orders o
JOIN Target.order_items oi
ON o.order_id =oi.order_id
JOIN Target.customers c
ON c.customer_id =o.customer_id
GROUP BY customer_state
ORDER BY diff_estimated_delivery asc
LIMIT 5;
```

The query results are displayed in a table below the editor:

Row	customer_state	diff_estimated_delivery
1	AL	7.97658079...
2	MA	9.10999999...
3	SE	9.16533333...
4	ES	9.76852932...
5	BA	10.1194078...

The interface also shows a sidebar with project explorer, a top navigation bar with Google Cloud logo and search, and a bottom status bar with system information.

--Q6. Payment type analysis:

--Q6.1. Month over Month count of orders for different payment types

```
SELECT *
FROM (
    SELECT
    EXTRACT (month FROM order_purchase_timestamp) Month,
    payment_type,
    COUNT(o.order_id) no_of_order
    FROM Target.orders o
    JOIN Target.payments p
    ON o.order_id =p.order_id
    GROUP BY payment_type,EXTRACT(month FROM order_purchase_timestamp)
)
ORDER BY month,payment_type,no_of_order;
```

The screenshot shows the Google Cloud BigQuery console interface. The top navigation bar includes the Google Cloud logo, the project name 'Scaler-DSML-SQL', and a search bar. The left sidebar contains the 'Explorer' panel with a search bar and a list of resources including 'Em', 'Northwind', 'Target', 'customers', 'geolocation', 'order\_items', 'order\_reviews', and 'orders'. The main panel displays a SQL query for 'Q6.1. Month over Month count of orders for different payment types'. The query is as follows:

```
294 --Q6. Payment type analysis:
295 --Q6.1. Month over Month count of orders for different payment types:
296 SELECT *
297 FROM (
298     SELECT
299     EXTRACT (month FROM order_purchase_timestamp) Month,
300     payment_type,
301     COUNT(o.order_id) no_of_order
302     FROM Target.orders o
303     JOIN Target.payments p
304     ON o.order_id =p.order_id
305     GROUP BY payment_type,EXTRACT(month FROM order_purchase_timestamp)
306 )
307 ORDER BY month,payment_type,no_of_order;
308
```

Below the query, the 'Query results' section shows a table with 3 rows and 3 columns: 'Month', 'payment\_type', and 'no\_of\_order'. The results are as follows:

Row	Month	payment_type	no_of_order
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118

The bottom of the console shows the 'PERSONAL HISTORY' and 'PROJECT HISTORY' sections, and a system tray at the very bottom with weather and time information.

--Q6.2. Count of orders based on the no. of payment installments

```
SELECT payment_installments, COUNT(DISTINCT order_id) as num_orders
FROM Target.payments
GROUP BY payment_installments
order by num_orders;
```

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

```
--Q6.2. Count of orders based on the no. of payment installments
SELECT payment_installments, COUNT(DISTINCT order_id) as num_orders
FROM Target.payments
GROUP BY payment_installments
order by num_orders;
```

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

Row	payment_installments	num_orders
1	22	1
2	23	1
3	0	2
4	21	3
5	16	5
6	17	8
7	14	15
8	13	16

The interface also shows a sidebar with project navigation, a top navigation bar with Google Cloud logo and search, and a bottom status bar with system information.