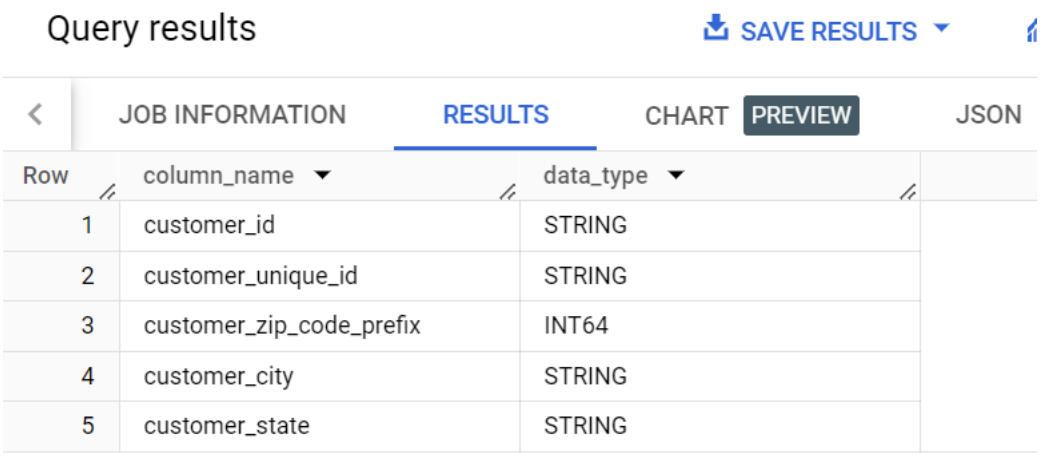
Business Case – Target SQL – Answers

I. 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.

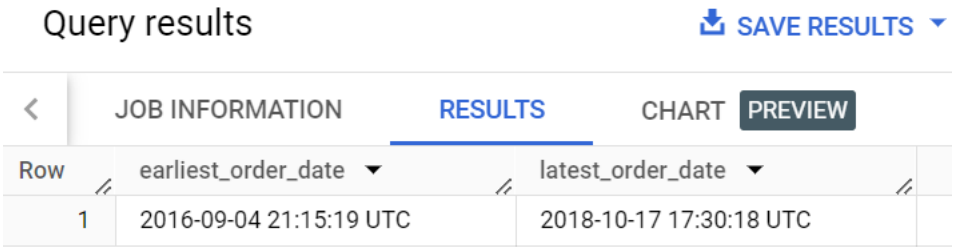
```
SELECT column_name, data_type
FROM `Business_case_scaler.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```



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 earliest_order_date,
  MAX(order_purchase_timestamp) AS latest_order_date
FROM `scaler-sql-dsml-402220.Business_case_scaler.orders`
```



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

Insights: Orders were placed by customers between Sep 2016 9:15 PM to October 2018 5:30 PM UTC.

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

```
SELECT
  COUNT(c.customer_city) AS city_count,
  COUNT(c.customer_state) AS state_count
FROM
  `scaler-sql-dsml-402220.Business_case_scaler.orders` o
```

```
INNER JOIN
  `Business_case_scaler.customers` c
ON
  o.customer_id = c.customer_id
WHERE
  o.order_purchase_timestamp BETWEEN (
    SELECT
      MIN(order_purchase_timestamp)
    FROM
      `scaler-sql-dsml-402220.Business_case_scaler.orders`
  ) AND (
    SELECT
      MAX(order_purchase_timestamp)
    FROM
      `scaler-sql-dsml-402220.Business_case_scaler.orders`
  )
GROUP BY
  c.customer_city,
  c.customer_state
ORDER BY
  c.customer_city,
  c.customer_state
LIMIT 10;
```

JOB INFORMATION | RESULTS

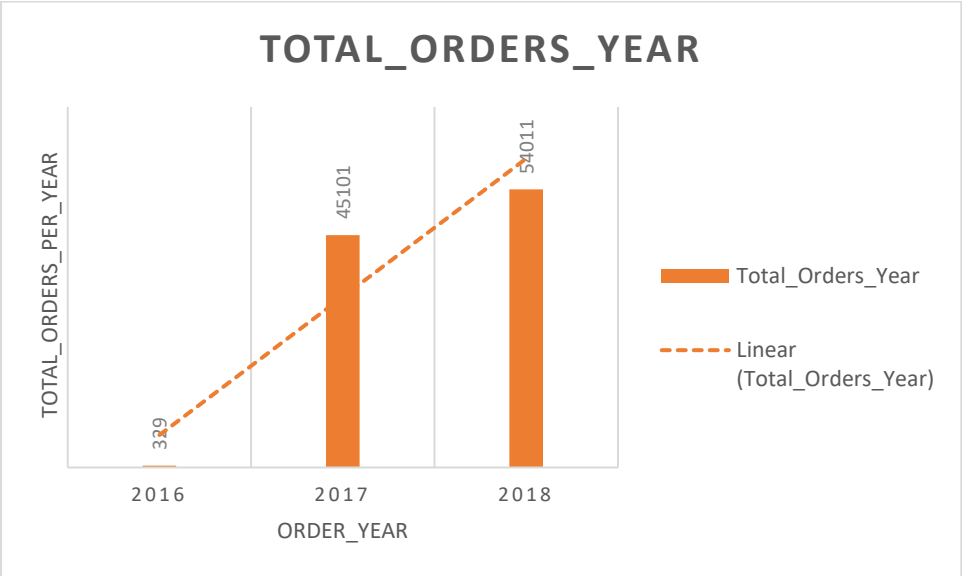
Row	city_count	state_count
1	3	3
2	1	1
3	12	12
4	11	11
5	2	2
6	2	2
7	2	2
8	3	3
9	1	1
10	6	6

II. In-depth Exploration:

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

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  COUNT(order_id) AS order_count
FROM
  `Business_case_scaler.orders`
GROUP BY
  order_year
ORDER BY
  order_year
```

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



Insights: Yearly Growth: The order count has been increasing steadily from 2016 to 2018, indicating consistent growth in sales over these years.

Recommendation: This suggests that the business has been successful in attracting more customers or retaining existing ones.

Insights: There's a significant spike in orders in 2017 & 2018 compared to the previous year.

Recommendation: This might indicate successful marketing campaigns, improved product offerings, or increased customer satisfaction during that year.

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

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
  FORMAT_DATE('%B', DATE_TRUNC(order_purchase_timestamp, MONTH)) AS order_month_mm,
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  COUNT(order_id) AS order_count
FROM
  `Business_case_scaler.orders`
GROUP BY
  order_month, order_month_mm, order_year
ORDER BY
  order_month,order_month_mm,  order_year
```

Insights:

In Year **2017** the Orders growth was linear and inclined growth. Whereas in **2018** the orders growth were declining graph.

In 2017 the orders peaked up in November and December months. However the orders growth was consistent from July to October month.

In 2018 the orders peak was in initial 3months and stayed consistent growth from April to August however the orders started declined were at very low during September and October months.

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

```
SELECT
  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
'Mornings'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
```

```
    ELSE 'Night'
  END AS Order_hours,
  COUNT(order_id) AS order_count
FROM `Business_case_scaler.orders`
WHERE order_purchase_timestamp BETWEEN (
    SELECT
      MIN(order_purchase_timestamp)
    FROM
      `Business_case_scaler.orders`
  ) AND (
    SELECT
      MAX(order_purchase_timestamp)
    FROM
      `Business_case_scaler.orders`
  )
GROUP BY Order_hours
ORDER BY Order_count;
```

Row	Order_hours	order_count
1	Dawn	5242
2	Mornings	27733
3	Night	28331
4	Afternoon	38135

Insights:

Max number of orders were placed during **Afternoon** hours.

Minimum and low orders were placed by Brazilian customers during **Dawn** hours.

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

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

```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
  FORMAT_DATE('%B', DATE_TRUNC(order_purchase_timestamp, MONTH)) AS order_month_mm,
  EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
  c.customer_state AS state,
  COUNT(o.order_id) AS order_count
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.customers` c
ON
  o.customer_id = c.customer_id
WHERE
  order_purchase_timestamp BETWEEN (
    SELECT
      MIN(order_purchase_timestamp)
```

```
    FROM
      `Business_case_scaler.orders`
  ) AND (
    SELECT
      MAX(order_purchase_timestamp)
    FROM
      `Business_case_scaler.orders`
  )
GROUP BY
  order_year,
  order_month,
  order_month_mm,
  state
ORDER BY
  order_year,
  order_month,
  order_month_mm,
  state;

SELECT
  customer_state AS state,
  COUNT(DISTINCT customer_id) AS customer_count
FROM
  `Business_case_scaler.customers`
GROUP BY
  state
ORDER BY
  state;
```

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

```
SELECT
  customer_state AS state,
  COUNT(DISTINCT customer_id) AS customer_count
FROM
  `Business_case_scaler.customers`
GROUP BY
  state
ORDER BY
  state;

  SELECT
  customer_state AS state,
  COUNT(DISTINCT customer_id) AS customer_count
FROM
  `Business_case_scaler.customers`
GROUP BY
  state
ORDER BY
  state;
```

Row	state	customer_count
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
10	MA	747

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

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

```
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(p.payment_value) AS total_payment_value,
    SUM(oi.price) AS total_price_value
  FROM
    `Business_case_scaler.payments` p
  JOIN
    `Business_case_scaler.orders` o ON p.order_id = o.order_id
  JOIN
    `Business_case_scaler.order_items` oi ON oi.order_id = o.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) BETWEEN 2017 AND 2018
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
  GROUP BY
    order_year,
    order_month
)
SELECT
  ((t2.cost_2018 - t1.cost_2017) / t1.cost_2017) * 100 AS
cost_increase_percentage
FROM (
  SELECT
    SUM(total_payment_value + total_price_value) AS cost_2017
  FROM
    YearlyCost
  WHERE
    order_year = 2017
) AS t1,
(
```

```
  SELECT
    SUM(total_payment_value + total_price_value) AS cost_2018
  FROM
    YearlyCost
  WHERE
    order_year = 2018
) AS t2
```

Row	cost_increase_perce
1	140.0896211729...

Insights:
140% increase in the cost of orders from year **2017 to 2018 (include months between Jan to Aug only).**

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

```
SELECT
  c.customer_state AS state,
  ROUND(SUM(oi.price),2) AS total_order_price,
  ROUND(AVG(oi.price),2) AS average_order_price
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.order_items` oi ON o.order_id = oi.order_id
JOIN
  `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY state, total_order_price, average_order_price;
```

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	state	total_order_price	average_order_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

Insights:

- This shows variations in total and average order prices across different states, highlighting regional differences in purchasing behaviors.
- States with higher total order prices might indicate regions where customers tend to place larger or more frequent orders.

Recommendations:

- Target marketing campaign to Identify states with higher total order prices for potential targeting of specialized products or services that align with the spending preferences in those areas.
- Customer Engagement: Explore reasons behind variations in spending behaviors among states. Engage with customers in regions with lower spending to understand their needs and preferences better.

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

```
SELECT
  c.customer_state AS state,
  ROUND(SUM(oi.freight_value),2) AS total_freight_value,
  ROUND(AVG(oi.freight_value),2) AS average_freight_value
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.order_items` oi ON o.order_id = oi.order_id
JOIN
  `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY state, total_freight_value, average_freight_value;
```

Row	state	total_freight_value	average_freight_valu
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	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

Insights:

- The result provides total and average freight values for each state, highlighting the overall spending on freight services per state.
- Variation in freight costs indicates variations in total and average freight costs among different states, showcasing potential discrepancies in shipping expenses across regions.

Recommendations:

Logistics Optimization: States with higher total freight costs might benefit from logistics optimization strategies. Explore options to streamline shipping routes, negotiate better carrier contracts, or leverage bulk shipment strategies to reduce overall freight expenses.

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

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

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_deliver_Days,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY) AS diff_estimated_delivery_Days
FROM
  `Business_case_scaler.orders`
WHERE
  order_delivered_customer_date IS NOT NULL
  AND order_estimated_delivery_date IS NOT NULL
LIMIT 10
```

Row	order_id	time_to_deliver_Days	diff_estimated_delivery_Days
1	770d331c84e5b214bd9dc70a1...	7	45
2	1950d777989f6a877539f5379...	30	-12
3	2c45c33d2f9cb8ff8b1c86cc28...	30	28
4	dabf2b0e35b423f94618bf965f...	7	44
5	8beb59392e21af5eb9547ae1a...	10	41
6	65d1e226dfaeb8cdc42f66542...	35	16
7	c158e9806f85a33877bdfd4f60...	23	9
8	b60b53ad0bb7dacacf2989fe2...	12	-5
9	c830f223aae08493ebecb52f2...	12	12
10	a8aa2cd070eeac7e4368cae3d...	7	1

Insights:

Delivery Timeframe: time_to_deliver_days indicates the number of days taken for an order to be delivered from the purchase date. It provides insights into the actual delivery duration for completed orders.

Recommendations:

Optimizing Delivery Speed: Analyze the factors influencing longer delivery times (time_to_deliver_Days) to identify bottlenecks in the delivery process. Streamlining logistics, optimizing routes, or enhancing fulfillment processes can reduce delivery times and enhance customer satisfaction.

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

Highest Average freight Values:

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

Row	order_id	customer_state	average_freight_valu
1	a77e1550db865202c56b19ddc...	PI	409.68
2	3fde74c28a3d5d618c00f26d5...	SC	375.28
3	076d1555fb53a89b0ef4d529e...	PR	375.28
4	9f49bd16053df810384e79338...	SP	339.59
5	264a7e199467906c0727394df...	MT	338.3

Lowest Average freight values:

```
Select * FROM

(SELECT
  o.order_id, c.customer_state,
  ROUND(AVG(oi.freight_value),2) AS average_freight_value
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.order_items` oi ON o.order_id = oi.order_id
JOIN
  `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
```

```
GROUP BY
   o.order_id, c.customer_state
)t
WHERE t.average_freight_value != 0
ORDER BY average_freight_value
LIMIT 5;
```

Row	order_id	customer_state	average_freight_valu
1	9ef13efd6949e4573a18964dd...	GO	1.2
2	ca3625898fbd48669d50701ab...	SP	1.9
3	090ece27665c1cc89eda287f8...	SP	1.98
4	9c4f3693a36ba481e0d9da739...	SP	2.24
5	895dce73de79888b14581541c...	PR	2.31

Insights:

Non-Zero Freight Charges: Filtering for average_freight_value != 0 ensures a focus on orders where freight costs were incurred, excluding cases where freight charges were null or zero.

Recommendations:

Individual Order Analysis: Investigate the specific details of the identified orders (top 5) with higher average freight values. Understand the nature of products, shipping distances, or any exceptional shipping requirements associated with these orders.

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

   Highest Average Delivery Time:

```
SELECT
  c.customer_state AS state,
  AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)) AS avg_delivery_time
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
WHERE
  o.order_delivered_customer_date IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time DESC
LIMIT 5
```

Row	state	avg_delivery_time
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...

Lowest Average Delivery Time:

```
SELECT
  c.customer_state AS state,
  AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY)) AS avg_delivery_time
FROM
  `Business_case_scaler.orders` o
JOIN
  `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
WHERE
  o.order_delivered_customer_date IS NOT NULL
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time
LIMIT 5
```

Row	state	avg_delivery_time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...

Insights

Timely Order Fulfillment: Focuses on regions where the average duration between order placement (order_purchase_timestamp) and delivery (order_delivered_customer_date) is comparatively shorter.

4. 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.

```
WITH DeliveryTime AS (
  SELECT
    c.customer_state AS state,
    AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)) AS avg_actual_delivery_time,
```

```
    AVG(DATE_DIFF(o.order_estimated_delivery_date,
o.order_purchase_timestamp, DAY)) AS avg_estimated_delivery_time
  FROM
    `Business_case_scaler.orders` o
  JOIN
    `Business_case_scaler.customers` c ON o.customer_id = c.customer_id
  WHERE
    o.order_delivered_customer_date IS NOT NULL
    AND o.order_estimated_delivery_date IS NOT NULL
  GROUP BY
    c.customer_state
)
SELECT
  state,
  (avg_estimated_delivery_time - avg_actual_delivery_time) AS
delivery_speed_difference
FROM
  DeliveryTime
ORDER BY
  delivery_speed_difference DESC
LIMIT 5
```

Row	state	delivery_speed_diffe
1	AC	20.08749999999...
2	RO	19.47325102880...
3	AP	19.13432835820...
4	AM	18.93793103448...
5	RR	16.65853658536...

Insights:

Delivery Speed Differential: By subtracting the average estimated time from the average actual time, the query quantifies the difference, highlighting states where deliveries consistently outperform expectations.

Recommendations:

Logistics Optimization: Assess transportation and logistics networks in regions with faster delivery times. Explore opportunities to replicate successful strategies or optimize routes to enhance delivery speed in other states.

VI. Analysis based on the payments:

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

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

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

Row	order_year	order_month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	debit_card	2
3	2016	10	voucher	23
4	2016	10	credit_card	254
5	2016	10	UPI	63
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	voucher	61
9	2017	1	credit_card	583
10	2017	1	debit_card	9

Insights:

Monthly Order Trends: The query provides a breakdown of order counts per month and year, indicating fluctuations in purchase volumes over time.

Recommendations:
Payment Method Promotion: Identify popular payment types across different months/years. Consider promoting or offering incentives for the usage of preferred payment methods to encourage their adoption.

2. 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 orders_count
FROM
  `Business_case_scaler.payments`
  WHERE payment_value !=0
GROUP BY
  payment_installments
```

Row	payment_installment	orders_count
1	0	2
2	1	49057
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644

Insights:

Payment Installment Analysis: The query categorizes orders based on the number of payment installments used, providing insights into how customers prefer to split their payments.

Recommendations:

Customer Behavior Understanding: Analyze patterns to understand why customers choose specific payment installment options. This insight can help tailor payment plans or offerings to align with customer preferences.