Target Brazil Sales Analysis

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

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

```
--This query will return a list of columns and their respective data types
--for the customers table in the Target_CaseStudy database.
SELECT
  column_name,
  data_type
FROM Target_CaseStudy.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'customers'
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

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

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

```
--This query will give you the range of time between which the orders were placed,
--showing the earliest and latest order_purchase_timestamp.
SELECT
  MIN(order_purchase_timestamp) first_order_date,
  MAX(order_purchase_timestamp) last_order_date
FROM `Target_CaseStudy.orders`
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EX

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

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

```sql
---This will give you the count of unique states and cities
--where customers who placed orders are located.
SELECT
  COUNT( DISTINCT customer_state) count_of_unique_states,
  COUNT( DISTINCT customer_city) count_of_unique_cities
FROM `Target_CaseStudy.customers` c JOIN `Target_CaseStudy.orders` o
    ON c.customer_id = o.customer_id
WHERE
  o.order_purchase_timestamp
    BETWEEN
  (SELECT MIN(order_purchase_timestamp) FROM `Target_CaseStudy.orders`)
    AND
  (SELECT MAX(order_purchase_timestamp) FROM `Target_CaseStudy.orders`)
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	count_of_unique_states	count_of_unique_cities
1	27	4119

II. In-depth Exploration:

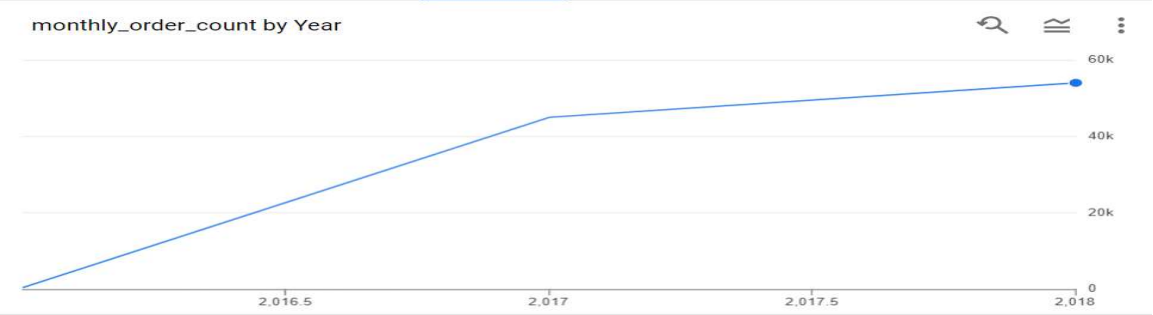
A. Is there a growing trend in the no. of orders placed over the past years?

```sql
--The query provides a clear view of the total number of orders placed each year.
--This can help in understanding how the volume of orders has changed over time.
SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) as Year,
COUNT(order_id) as monthly_order_count
FROM `Target_CaseStudy.orders`
GROUP BY Year
ORDER BY Year
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EX

Row	Year	monthly_order_coun
1	2016	329
2	2017	45101
3	2018	54011



Insight: The count in 2016 is less because of 3 months of data. Sales increased by 19.75% from 2017 to 2018.

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

1st Query

```
1  --This query will help you see if the number of orders has increased gradually each month
2  --over the past years by calculating the total number of orders placed each month for each year.
3  SELECT
4  EXTRACT(YEAR FROM order_purchase_timestamp) as Year,
5  FORMAT_TIMESTAMP('%b', order_purchase_timestamp) as Month,
6  EXTRACT(MONTH FROM order_purchase_timestamp) as Month_Number,
7  COUNT(order_id) as Order_Count
8  FROM `Target_CaseStudy.orders`
9  GROUP BY Year,Month, Month_Number
10 ORDER BY Year, Month_Number;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	Year	Month	Month_Number	Order_Count
1	2016	Sep	9	4
2	2016	Oct	10	324
3	2016	Dec	12	1
4	2017	Jan	1	800
5	2017	Feb	2	1780
6	2017	Mar	3	2682
7	2017	Apr	4	2404
8	2017	May	5	3700
9	2017	Jun	6	3245
10	2017	Jul	7	4026
11	2017	Aug	8	4331
12	2017	Sep	9	4285
13	2017	Oct	10	4631
14	2017	Nov	11	7544
15	2017	Dec	12	5673
16	2018	Jan	1	7269
17	2018	Feb	2	6728
18	2018	Mar	3	7211
19	2018	Apr	4	6939
20	2018	May	5	6873
21	2018	Jun	6	6167
22	2018	Jul	7	6292
23	2018	Aug	8	6512
24	2018	Sep	9	16
25	2018	Oct	10	4

2nd Query

```
SELECT
  FORMAT_TIMESTAMP("%b", order_purchase_timestamp) Month,
  EXTRACT(MONTH FROM order_purchase_timestamp) Month_Number,
  COUNT(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2016 THEN order_id END) order_in_2016,
  COUNT(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2017 THEN order_id END) order_in_2017,
  COUNT(CASE WHEN EXTRACT(YEAR FROM order_purchase_timestamp) = 2018 THEN order_id END) order_in_2018
FROM `Target_CaseStudy.orders`
GROUP BY Month, Month_Number
ORDER BY Month_Number
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | E

Row	Month	Month_Number	order_in_2016	order_in_2017	order_in_2018
1	Jan	1	0	800	7269
2	Feb	2	0	1780	6728
3	Mar	3	0	2682	7211
4	Apr	4	0	2404	6939
5	May	5	0	3700	6873
6	Jun	6	0	3245	6167
7	Jul	7	0	4026	6292
8	Aug	8	0	4331	6512
9	Sep	9	4	4285	16
10	Oct	10	324	4631	4
11	Nov	11	0	7544	0
12	Dec	12	1	5673	0

Insights: Looking at the counts across the three years, 2016 had low orders from people who might have taken time to become aware of this platform. Then, in 2017, the number of orders per month increased until the end of the year. However, the order count from Sep for 2018 has decreased.

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

```sql
-- It categorizes the order times into four intervals: Dawn, Morning, Afternoon, and Night,
--based on the hour of the day extracted from the order_purchase_timestamp field.
SELECT
  COUNT(order_id) Order_Count,
  CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN "Dawn"
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN "Morning"
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN "Afternoon"
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN "Night"
  ELSE "Other" END AS Order_time_category
FROM `Target_CaseStudy.orders`
GROUP BY order_time_category
ORDER BY Order_Count
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GI

Row	Order_Count	Order_time_category
1	5242	Dawn
2	27733	Morning
3	28331	Night
4	38135	Afternoon

Bar chart

Insights: The highest order count is during the Afternoon (13-18 HRS) when Brazilian customers place the most orders.

Recommendations: Schedule marketing campaigns, promotions, and advertisements during peak ordering times to maximize engagement and sales. Allocate more customer service and support resources during peak order times to handle the increased volume effectively.

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

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

```sql
--This query gives month-on-month number of orders placed in each state
--with month names as columns and states as rows.
SELECT
  customer_state State,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 1 THEN 1 END) AS Jan,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 2 THEN 1 END) AS Feb,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 3 THEN 1 END) AS Mar,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 4 THEN 1 END) AS Apr,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 5 THEN 1 END) AS May,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 6 THEN 1 END) AS Jun,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 7 THEN 1 END) AS Jul,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 8 THEN 1 END) AS Aug,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 9 THEN 1 END) AS Sep,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 10 THEN 1 END) AS Oct,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 11 THEN 1 END) AS Nov,
  SUM(CASE WHEN EXTRACT(MONTH FROM order_purchase_timestamp) = 12 THEN 1 END) AS Dec
FROM
  `Target_CaseStudy.orders` o JOIN `Target_CaseStudy.customers` c
    ON o.customer_id = c.customer_id
GROUP BY
  customer_state
ORDER BY
  customer_state;
```

Query results

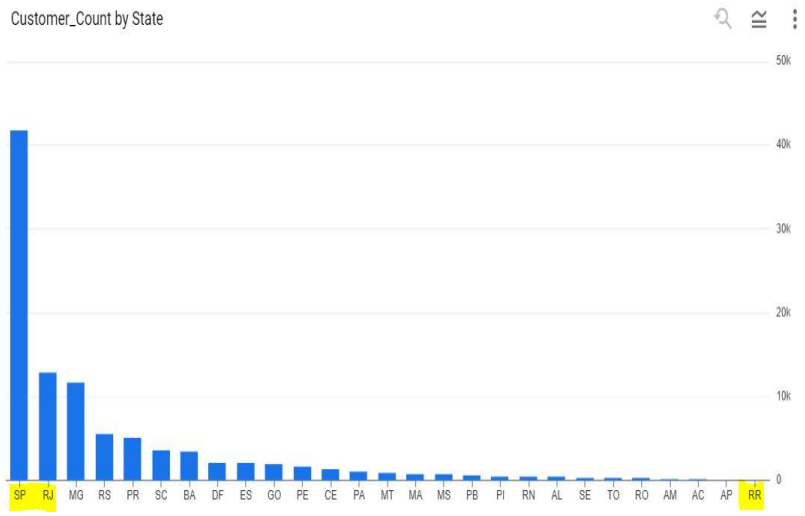
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	State	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	AC	8	6	4	9	10	7	9	7	5	6	5	5
2	AL	39	39	40	51	46	34	40	34	20	30	26	14
3	AM	12	16	14	19	19	8	23	9	9	3	10	6
4	AP	11	4	8	5	11	4	7	5	2	3	4	4
5	BA	264	273	340	318	368	307	405	323	170	170	250	192
6	CE	99	101	126	143	136	121	140	130	77	74	108	81
7	DF	151	196	207	183	208	220	243	232	97	104	168	131
8	ES	159	186	182	188	228	204	206	200	93	104	170	113
9	GO	164	176	199	177	226	184	192	213	88	117	157	127
10	MA	66	67	77	73	65	59	79	70	42	52	56	41
11	MG	971	1063	1237	1061	1190	1080	1111	1177	511	600	943	691
12	MS	71	75	79	58	74	76	74	59	33	34	46	36
13	MT	96	84	71	92	104	83	85	78	35	55	74	50
14	PA	82	83	109	107	75	92	96	104	41	58	70	58
15	PB	33	47	55	51	47	51	79	46	29	31	30	37
16	PE	113	146	153	154	174	140	210	170	76	87	126	103
17	PI	55	46	48	50	56	43	52	43	23	25	31	23
18	PR	443	460	504	500	524	478	523	556	183	225	378	271
19	RJ	990	1176	1302	1172	1321	1128	1288	1307	612	725	1048	783
20	RN	51	31	52	42	39	49	56	40	24	27	44	30
21	RO	23	25	29	20	26	22	27	23	16	14	17	11
22	RR	2	7	8	4	3	8	6	null	2	4	2	null
23	RS	427	473	569	488	559	526	565	599	279	276	422	283
24	SC	345	316	362	351	379	321	356	365	157	189	303	193
25	SE	24	27	43	27	19	37	42	43	16	25	27	20
26	SP	3351	3357	4047	3967	4632	4104	4381	4982	1648	1908	3012	2357
27	TO	19	28	28	33	34	26	23	28	17	13	17	14

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

```
SELECT
customer_state State,
COUNT(*) Customer_Count,
FROM `Target_CaseStudy.customers`
GROUP BY customer_state
ORDER BY Customer_Count DESC
```

Row	State	Customer_Count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336
13	PA	975
14	MT	907
15	MA	747
16	MS	715
17	PB	536
18	PI	495
19	RN	485
20	AL	413
21	SE	350
22	TO	280
23	RO	253
24	AM	148
25	AC	81
26	AP	68
27	RR	46



<u>Insights for both A and B</u>:

High-Volume States: **States like São Paulo (SP) and Rio de Janeiro (RJ) have a significantly higher number of unique customers and orders. The high order volume in SP and RJ might be due to a larger population, better internet penetration, and higher economic activity.**

Low-Volume States: States like AP and RR have the fewest customer orders. The lower order volume might be due to a smaller population, lower economic activity, or logistical challenges.

<u>Recommendations</u>:

Expand in Less-Volume States: **While SP and RJ are essential markets, Target may explore opportunities for expansion and growth in states with lower customer counts, such as Roraima (RR), Amapá (AP). Investigate the reasons behind the lower customer counts in these regions and develop strategies to increase market penetration.**

IV. Impact on the 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 the year 2017 to 2018 *(include months between Jan to Aug only).*

```sql
WITH year_wise AS
(SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) Month,
  FORMAT_TIMESTAMP("%b", order_purchase_timestamp) Month_name,
  ROUND(SUM(payment_value),2) Cost_of_Orders
FROM `Target_CaseStudy.payments` p JOIN `Target_CaseStudy.orders` o
  ON p.order_id = o.order_id
WHERE EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY Year, Month, Month_name),
  y2017 AS
  (SELECT
    Year,
    Month,
    Month_name,
    Cost_of_Orders AS Cost_2017
  FROM year_wise
  WHERE Year = 2017
  ORDER BY Month),
  y2018 AS
  (SELECT
    Year,
    Month,
    Month_name,
    Cost_of_Orders AS Cost_2018
  FROM year_wise
  WHERE Year = 2018
  ORDER BY Month)

--Final SELECT--

  SELECT
    y2017.Month_name as Month,
    y2017.Cost_2017,
    y2018.Cost_2018,
    ROUND((y2018.Cost_2018 - y2017.Cost_2017)/y2017.Cost_2017 * 100,2) Percentage_Difference
  FROM y2017 JOIN y2018
    ON y2017.Month = y2018.Month
  ORDER BY y2017.Month
```

Row	Month	Cost_2017	Cost_2018	Percentage_Difference
1	Jan	138488.04	1115004.18	705.13
2	Feb	291908.01	992463.34	239.99
3	Mar	449863.6	1159652.12	157.78
4	Apr	417788.03	1160785.48	177.84
5	May	592918.82	1153982.15	94.63
6	Jun	511276.38	1023880.5	100.26
7	Jul	592382.92	1066540.75	80.04
8	Aug	674396.32	1022425.32	51.61

Insights: **The cost of orders from 2017 to 2018 increased for all months. January witnessed the highest percentage increment, while August was the least.**

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

```sql
SELECT
  customer_state AS State,
  ROUND(SUM(price),2) Total_Order_Price,
  ROUND(AVG(price),2) Avg_Order_Price,
FROM `Target_CaseStudy.order_items` oi JOIN `Target_CaseStudy.orders` o
    ON oi.order_id = o.order_id
    JOIN `Target_CaseStudy.customers` c
    ON o.customer_id = c.customer_id
GROUP BY State
ORDER BY Total_Order_Price Desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JS

Row	State	Total_Order_Price	Avg_Order_Price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76
13	PA	178947.81	165.69
14	MT	156453.53	148.3
15	MA	119648.22	145.2
16	MS	116812.64	142.63
17	PB	115268.08	191.48
18	PI	86914.08	160.36
19	RN	83034.98	156.97
20	AL	80314.81	180.89
21	SE	58920.85	153.04
22	TO	49621.74	157.53
23	RO	46140.64	165.97
24	AM	22356.84	135.5
25	AC	15982.95	173.73
26	AP	13474.3	164.32
27	RR	7829.43	150.57

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

```
SELECT
  customer_state AS State,
  ROUND(SUM(freight_value),2) Total_Freight_Value,
  ROUND(AVG(freight_value),2) Avg_Freight_Value,
FROM `Target_CaseStudy.order_items` oi JOIN `Target_CaseStudy.orders` o
    ON oi.order_id = o.order_id
    JOIN `Target_CaseStudy.customers` c
    ON o.customer_id = c.customer_id
GROUP BY State
ORDER BY Total_Freight_Value Desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSO

Row	State	Total_Freight_Value	Avg_Freight_Value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71
13	PA	38699.3	35.83
14	MA	31523.77	38.26
15	MT	29715.43	28.17
16	PB	25719.73	42.72
17	PI	21218.2	39.15
18	MS	19144.03	23.37
19	RN	18860.1	35.65
20	AL	15914.59	35.84
21	SE	14111.47	36.65
22	TO	11732.68	37.25
23	RO	11417.38	41.07
24	AM	5478.89	33.21
25	AC	3686.75	40.07
26	AP	2788.5	34.01
27	RR	2235.19	42.98

Insights: **Evidently, the state 'SP' has the maximum Order Price and Freight Value. However, its Average Order Price and Freight Value are the lowest among all, maybe because of its large population. The state 'RR' has the lowest Total Price and Freight Value, but its Average Order Price and Freight Value are neither the highest nor the lowest.**

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

A. 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.

```sql
SELECT
  order_id,
  DATE(order_purchase_timestamp) as Purchase_Date,
  DATE(order_delivered_customer_date) as Delivery_Date,
  DATE(order_estimated_delivery_date) as Est_Delivery_Date,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS time_to_deliver,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS diff_estimated_delivery
FROM `Target_CaseStudy.orders`
WHERE
  order_status = "delivered"
ORDER BY
  order_purchase_timestamp
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	order_id	Purchase_Date	Delivery_Date	Est_Delivery_Date	time_to_deliver	diff_estimated_delivery
1	bfbd0f9bdef84302105ad712d...	2016-09-15	2016-11-09	2016-10-04	54	-36
2	3b697a20d9e427646d925679...	2016-10-03	2016-10-26	2016-10-27	23	0
3	be5bc2f0da14d8071e2d45451...	2016-10-03	2016-10-27	2016-11-07	24	10
4	a41c8759fbe7aab36ea07e038...	2016-10-03	2016-11-03	2016-11-29	30	25
5	d207cc272675637bfed0062ed...	2016-10-03	2016-10-31	2016-11-23	27	22
6	cd3b8574c82b42fc8129f6d50...	2016-10-03	2016-10-14	2016-11-23	10	39
7	ae8a60e4b03c5a4ba9ca0672c...	2016-10-03	2016-11-03	2016-12-01	30	27
8	ef1b29b591d31d57c0d733746...	2016-10-03	2016-11-01	2016-11-25	28	23
9	0a0837a5eee9e7a9ce2b1fa83...	2016-10-04	2016-10-22	2016-11-24	18	32
10	1ff217aa612f6cd7c4255c9bfe...	2016-10-04	2016-10-24	2016-11-24	20	30
11	ed8c7b1b3eb256c70ce0c7423...	2016-10-04	2016-11-18	2016-11-24	44	5
12	c3d9e402b6a0fbe2a5f7fc5b41...	2016-10-04	2016-11-08	2016-12-08	35	29
13	46046adea0e222a29259bad3d...	2016-10-04	2016-10-24	2016-11-28	20	34
14	80606b26965c5ed21e85a085e...	2016-10-04	2016-11-17	2016-12-08	44	20
15	79ffdd52a918bbe867895a4b1...	2016-10-04	2016-10-26	2016-11-24	22	28
16	c4b41c36dd589e901f6879f25...	2016-10-04	2016-11-09	2016-11-24	36	14
17	36989eb07a0de2d3d3129eea3...	2016-10-04	2016-10-11	2016-12-06	7	55
18	3f72d2b757e725cd48a4726f8...	2016-10-04	2016-10-17	2016-11-28	12	41
19	f3f12fc90564a9b036680a887c...	2016-10-04	2016-11-01	2016-11-24	28	22
20	5b1376fe61863fe3508011db3...	2016-10-04	2016-10-17	2016-12-08	13	51

Insights:

"time_to_deliver" to assess the overall delivery performance and identify any delays. A shorter delivery time generally indicates better efficiency in the logistics and supply chain.

"diff_estimated_delivery" helps in understanding the accuracy of the estimated delivery dates. Negative values indicate late deliveries, while positive values indicate early deliveries.

Recommendations:

Dates with negative:

a. **Root cause analysis:** Conduct a thorough analysis of the reasons behind delays. Is it due to shipping partner issues, internal processing inefficiencies, or other factors? Identifying the root causes can help you address the underlying issues effectively. Use this data to set improvement targets and track progress over time.

Days more than a week:

a. **Optimize Inventory Management:** Maintain sufficient stock levels of popular products to avoid stockouts and minimize the need for backorders, which can lead to extended delivery times.

b. **Improve order processing Speed:** Invest in order processing automation and technology to reduce the time it takes to process and fulfill customer orders.

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

Highest State Average Freight Value

```
WITH
state_freight_values AS
  (SELECT
    customer_state AS State,
    ROUND(AVG(freight_value),2) AS Avg_freight_value
  FROM
    `Target_CaseStudy.order_items` oi
  JOIN
    `Target_CaseStudy.orders` o
  ON
    oi.order_id = o.order_id
  JOIN
    `Target_CaseStudy.customers` c
  ON
    o.customer_id = c.customer_id
  GROUP BY customer_state),
top_5_state AS
  (SELECT
    State,
    Avg_freight_value
  FROM state_freight_values
  ORDER BY avg_freight_value DESC
  LIMIT 5)

  SELECT
    State AS Highest_state,
    Avg_freight_value AS Highest_freight_value
  FROM top_5_state
  ORDER BY highest_freight_value
```

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

Lowest State Average Freight Value

```sql
WITH
state_freight_values AS
  (SELECT
    customer_state AS State,
    ROUND(AVG(freight_value),2) AS Avg_freight_value
  FROM
    `Target_CaseStudy.order_items` oi
  JOIN
    `Target_CaseStudy.orders` o
  ON
    oi.order_id = o.order_id
  JOIN
    `Target_CaseStudy.customers` c
  ON
    o.customer_id = c.customer_id
  GROUP BY customer_state),
bottom_5_state AS
  (SELECT
    State,
    Avg_freight_value
  FROM state_freight_values
  ORDER BY avg_freight_value
  LIMIT 5)

  SELECT
    State AS Lowest_state,
    Avg_freight_value AS Lowest_freight_value
  FROM bottom_5_state
  ORDER BY Lowest_freight_value
```

Row	Lowest_state	Lowest_freight_value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

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

Highest Average delivery time.

```sql
WITH
state_deliver_time AS
  (SELECT
    customer_state AS State,
    ROUND(AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY)),2) AS time_to_deliver
  FROM
    `Target_CaseStudy.orders` o
  JOIN
    `Target_CaseStudy.customers` c
  ON
    o.customer_id = c.customer_id
  WHERE order_status = 'delivered'
  GROUP BY customer_state),

top_5_state AS
  (SELECT
    State,
    time_to_deliver
  FROM state_deliver_time
  ORDER BY time_to_deliver DESC
  LIMIT 5 )

  SELECT
    State,
    time_to_deliver AS highest_time_to_deliver
  FROM top_5_state
  ORDER BY highest_time_to_deliver
```

Row	State	highest_time_to_deliver
1	PA	23.32
2	AL	24.04
3	AM	25.99
4	AP	26.73
5	RR	28.98

Lowest Average delivery time.

```sql
WITH
state_deliver_time AS
  (SELECT
    customer_state AS State,
    ROUND(AVG(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY)),2) AS time_to_deliver
  FROM
    `Target_CaseStudy.orders` o
  JOIN
    `Target_CaseStudy.customers` c
  ON
    o.customer_id = c.customer_id
  WHERE order_status = 'delivered'
  GROUP BY customer_state),

bottom_5_state AS
  (SELECT
    State,
    time_to_deliver
  FROM state_deliver_time
  ORDER BY time_to_deliver
  LIMIT 5 )

  SELECT
    State,
    time_to_deliver AS lowest_time_to_deliver
  FROM bottom_5_state
  ORDER BY lowest_time_to_deliver
```

Row	State	lowest_time_to_deliver
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Insights:

1. The highest average delivery times are significantly above 20 days. There is a relatively small range between the top state (RR) and the fifth state (PA), suggesting consistent delays across these states.
2. The lowest average delivery time for the state (SP) is just 8.3 days. The range of delivery times is wider among the lowest states compared to the highest states, indicating some variability in delivery efficiency.

Recommendations:

1. For States with the Highest Average Delivery Time, conduct a detailed analysis to identify the root causes of delays in the states with the highest average delivery times (RR, AP, AM, AL, PA). Look into factors such as transportation issues, warehouse inefficiencies, or logistical challenges.
2. For States with the Lowest Average Delivery Time, analyze the strategies and practices used in these states and try to replicate them in states with higher delivery times. Identify key success factors and apply them where possible.

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

```sql
WITH state_delivery_time AS
(SELECT
  customer_state AS State,
  ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)),2) Avg_delivery_time,
  ROUND(AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)),2) AS Avg_est_delivery_time
FROM
  `Target_CaseStudy.orders` o
JOIN
  `Target_CaseStudy.customers` c
ON
  o.customer_id = c.customer_id
WHERE order_status = 'delivered'
GROUP BY customer_state
),
delivery_difference AS
(
  SELECT
    State,
    Avg_delivery_time,
    Avg_est_delivery_time,
    ROUND((Avg_est_delivery_time - Avg_delivery_time),2) AS delivery_diff
  FROM
    state_delivery_time
  ORDER BY delivery_diff
  LIMIT 5)
SELECT
  State,
  delivery_diff AS Delivery_speed
FROM delivery_difference
ORDER BY Delivery_speed
```

Row	State	Delivery_speed
1	AL	-16.09
2	RR	-12.57
3	MA	-12.35
4	SE	-11.86
5	CE	-10.86

Insight: Negative values indicate that every state had more late deliveries than early ones. Hence, the average fastest delivery value for each state is negative.

Recommendations: In this situation, since orders are arriving before the estimated delivery time, we can either reduce the estimated time and make the customer feel more valued or reduce the freight charges for these states to match the actual delivery time with the estimated delivery time, thereby reducing the total cost.

VI. Analysis based on the payments:

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

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  FORMAT_TIMESTAMP('%B', order_purchase_timestamp) AS month_name,
COUNT(CASE WHEN p.payment_type  = 'credit_card' THEN o.order_id END) AS credit_card,
COUNT(CASE WHEN p.payment_type  = 'debit_card' THEN o.order_id END) AS debit_card,
COUNT(CASE WHEN p.payment_type  = 'UPI' THEN o.order_id END) AS UPI,
COUNT(CASE WHEN p.payment_type  = 'voucher' THEN o.order_id END) AS voucher,
COUNT(CASE WHEN p.payment_type  = 'not_defined' THEN o.order_id END) AS not_defined
FROM
  `Target_CaseStudy.orders` o
JOIN
  `Target_CaseStudy.payments` p
ON
  o.order_id = p.order_id
GROUP BY
  year, month, month_name
ORDER BY
  year, month, month_name
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTIOI

Row	year	month	month_name	credit_card	debit_card	UPI	voucher	not_defined
1	2016	9	September	3	0	0	0	0
2	2016	10	October	254	2	63	23	0
3	2016	12	December	1	0	0	0	0
4	2017	1	January	583	9	197	61	0
5	2017	2	February	1356	13	398	119	0
6	2017	3	March	2016	31	590	200	0
7	2017	4	April	1846	27	496	202	0
8	2017	5	May	2853	30	772	289	0
9	2017	6	June	2463	27	707	239	0
10	2017	7	July	3086	22	845	364	0
11	2017	8	August	3284	34	938	294	0
12	2017	9	September	3283	43	903	287	0
13	2017	10	October	3524	52	993	291	0
14	2017	11	November	5897	70	1509	387	0
15	2017	12	December	4377	64	1160	294	0
16	2018	1	January	5520	109	1518	416	0
17	2018	2	February	5253	69	1325	305	0
18	2018	3	March	5691	78	1352	391	0
19	2018	4	April	5455	97	1287	370	0
20	2018	5	May	5497	51	1263	324	0
21	2018	6	June	4813	182	1100	324	0
22	2018	7	July	4755	242	1229	281	0
23	2018	8	August	4985	277	1139	295	2
24	2018	9	September	0	0	0	15	1
25	2018	10	October	0	0	0	4	0

Insights: The number of orders increased clearly from 2016 to 2018. Per output, credit card is the most used payment method, which is nowadays the easiest and most seamless for customers.

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

```sql
SELECT
  payment_installments,
  COUNT(DISTINCT order_id) AS num_orders
FROM
  `Target_CaseStudy.payments`
WHERE
  payment_installments != 0  -- This ensures we are counting orders with at least one installment
GROUP BY
  payment_installments
ORDER BY
  payment_installments;
```

JOB INFORMATION | RESULTS | CH

Row	payment_installments	num_orders
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315
11	11	23
12	12	133
13	13	16
14	14	15
15	15	74
16	16	5
17	17	8
18	18	27
19	20	17
20	21	3
21	22	1
22	23	1

Insights: Most of the orders are placed in single payment/installment.