SQL PROJECT

DSML Feb23 Beginner Morning Tue

In collaboration with

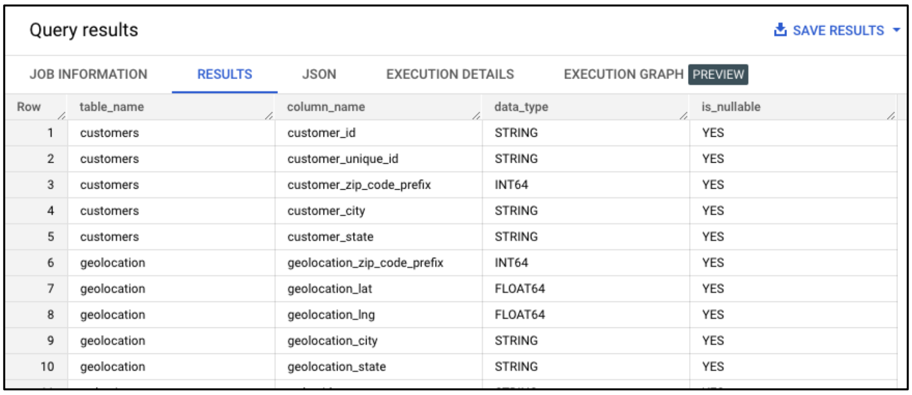
Scaler Academy & Target

Bhavesh Gawade

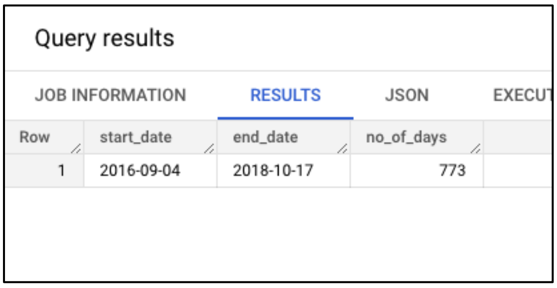
Bhavesh.gawade@gmail.com

Main question	1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
Sub question	1. Data type of columns in a table

Query	SELECT
	    table_name,
	    column_name,
	    data_type,
	    is_nullable
	FROM `scaler-target-assignment.target.INFORMATION_SCHEMA.COLUMNS`
	ORDER BY table_name, ordinal_position

Assumptions	
Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)	
Explanation	1. INFORMATION_SCHEMA contains the metadata for the objects within given project, schema.
Insights & Recommendation	NA
Images / graphs	NA

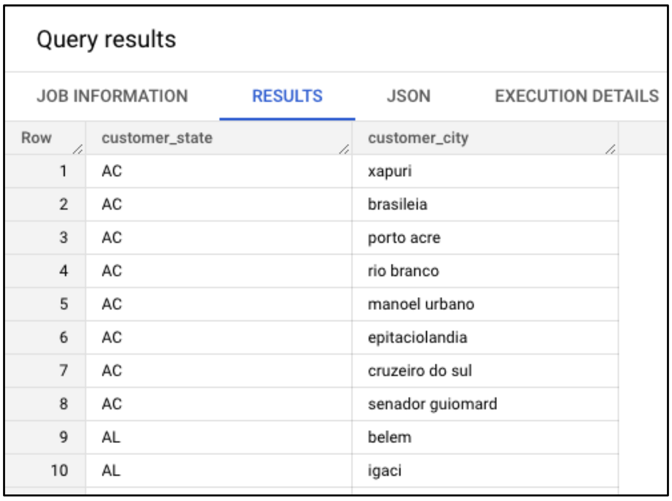
Row	table_name	column_name	data_type	is_nullable
1	customers	customer_id	STRING	YES
2	customers	customer_unique_id	STRING	YES
3	customers	customer_zip_code_prefix	INT64	YES
4	customers	customer_city	STRING	YES
5	customers	customer_state	STRING	YES
6	geolocation	geolocation_zip_code_prefix	INT64	YES
7	geolocation	geolocation_lat	FLOAT64	YES
8	geolocation	geolocation_lng	FLOAT64	YES
9	geolocation	geolocation_city	STRING	YES
10	geolocation	geolocation_state	STRING	YES

<table>
<tr><td>Main question</td><td>1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset</td></tr>
<tr><td>Sub question</td><td>2. Time period for which the data is give</td></tr>
<tr><td>Query</td><td><pre><code>SELECT
    MIN(date(order_purchase_timestamp)) AS start_date,
    MAX(date(order_purchase_timestamp)) AS end_date,
    date_diff(
                MAX(date(order_purchase_timestamp)),
                MIN(date(order_purchase_timestamp)),
                day
) no_of_days
FROM `target.orders`</code></pre></td></tr>
<tr><td>Assumptions</td><td></td></tr>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>
Query results

<table>
<tr><th>Row</th><th>start_date</th><th>end_date</th><th>no_of_days</th></tr>
<tr><td>1</td><td>2016-09-04</td><td>2018-10-17</td><td>773</td></tr>
</table></td></tr>
<tr><td>Explanation</td><td>1. The dataset belongs to the orders made by the target customer. Min & max of order_purchase_date along with date() function time will give the date range for the data.
2. date_diff function mentioned in the query will give the duration in number of days for which the data is present in dataset.
3. Dataset contains the data for 733 days.</td></tr>
<tr><td>Insights & Recommendation</td><td>NA</td></tr>
<tr><td>Images / graphs</td><td>NA</td></tr>
</table>

Main question	1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
Sub question	3. Cities and States of customers ordered during the given period

Query	`SELECT`
	`    DISTINCT`
	`        customer_state,`
	`        customer_city`
	``FROM `target.customers` ``
	`WHERE customer_id IN`
	`    (SELECT`
	
	`        DISTINCT customer_id`
	``    FROM `target.orders` ``
	`    )`
	`ORDER BY  customer_state`

Assumptions	
Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)	

Query results

Row	customer_state	customer_city
1	AC	xapuri
2	AC	brasileia
3	AC	porto acre
4	AC	rio branco
5	AC	manoel urbano
6	AC	epitaciolandia
7	AC	cruzeiro do sul
8	AC	senador guiomard
9	AL	belem
10	AL	igaci

Explanation	1. Subquery in where clause gets the distinct customer_id. As one customer can do multiple orders hence distinct keyword is used in the inner query to eliminate the duplicate customer id. This may improve the performance of outer query as it need to check for less number of customer ids. 2. Outer query uses distinct over customer_state, customer_city as one customer state & city can have multiple customers. 3. Although optional, outer query uses order by clause over customer_state to make data more readable.
Insights & Recommendation	NA
Images / graphs	NA

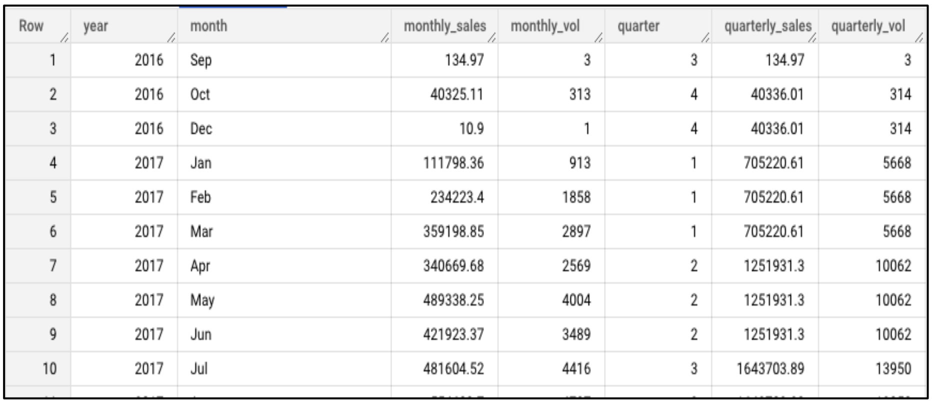
Main question	2. In-depth Exploration
Sub question	1. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?
Query	

```
with order_pattern
as (
  select
    year,
    month_num,
    month,
    round(sum(total_price),2) monthly_sales,
    count(*) monthly_vol
    from (
    select
      extract(YEAR from o.order_purchase_timestamp)
year,
      extract(MONTH from o.order_purchase_timestamp)
month_num,
      FORMAT_DATE('%b', o.order_purchase_timestamp)
month,
      price total_price
    from `target.orders` o
    join `target.order_items` oi on o.order_id =
oi.order_id
    where order_status = 'delivered'
  )
  group by year, month_num, month
)

select
  year,
  month,
  monthly_sales,
  monthly_vol,
  quarter,
  round(sum(monthly_sales) over(partition by year,
quarter),2) quarterly_sales,
  sum(monthly_vol) over(partition by year, quarter)
quarterly_vol
from (
select
  op.year,
  op.month,
  op.month_num,
  case
    when month_num in (1,2,3) then 1
    when month_num in (4,5,6) then 2
    when month_num in (7,8,9) then 3
    when month_num in (10,11,12) then 4
  end as quarter,
  op.monthly_sales,monthly_vol
from order_pattern op
)
order by year, month_num, quarter;
```

Assumptions

1. Orders which are delivered are considered for calculating volume and sales. This is because the orders cancelled won't generate any sales to the company.
2. order_item table contains the price of item in column price and expense incurred for delivering it in column freight_value. As freight_value is an expense to the company it is not considered while calculating sales.
3. Payments tables payment_value contains price and freight value hence that table & column was not considered for analysis.
4. For the improved readability, month abbreviations are used rather than month number.
5. Data is complete for the months present in the dataset. i.e. complete data is given for all the year & months present in the dataset.

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



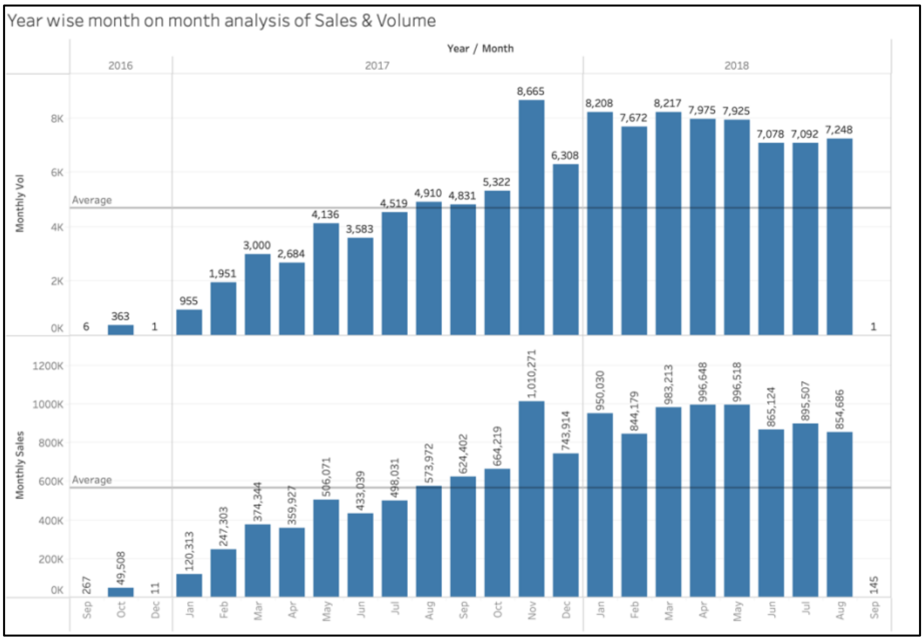
Row	year	month	monthly_sales	monthly_vol	quarter	quarterly_sales	quarterly_vol
1	2016	Sep	134.97	3	3	134.97	3
2	2016	Oct	40325.11	313	4	40336.01	314
3	2016	Dec	10.9	1	4	40336.01	314
4	2017	Jan	111798.36	913	1	705220.61	5668
5	2017	Feb	234223.4	1858	1	705220.61	5668
6	2017	Mar	359198.85	2897	1	705220.61	5668
7	2017	Apr	340669.68	2569	2	1251931.3	10062
8	2017	May	489338.25	4004	2	1251931.3	10062
9	2017	Jun	421923.37	3489	2	1251931.3	10062
10	2017	Jul	481604.52	4416	3	1643703.89	13950

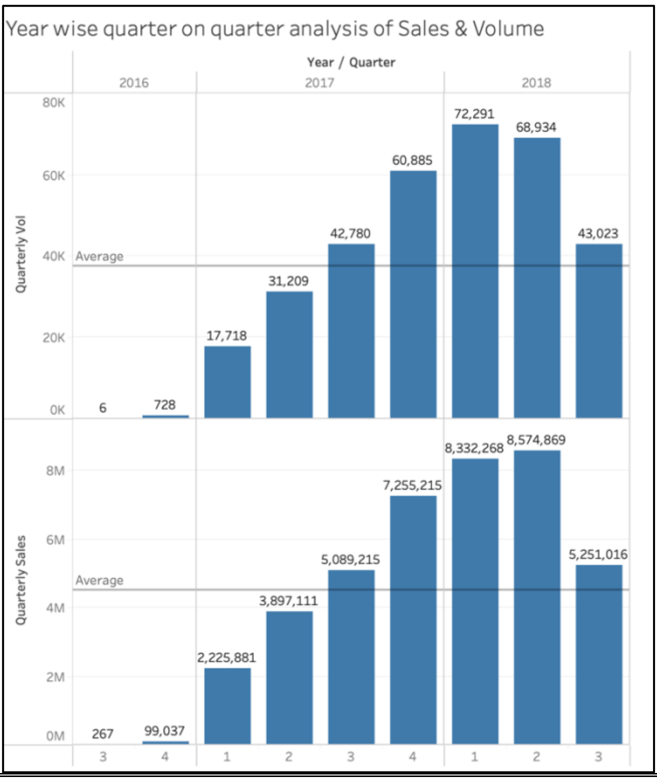
Explanation

1. order_pattern is common table expression. It used to make query simpler and readable. order_pattern inner query selects the data from orders & order_items table in target schema and inner join is performed on order_id column of both the tables. Select clause has extracts month in number & abbreviation & year into two separate columns from order_purchase_timestamp column of orders table. Then it selects prices from order_item table. Outer query in order_pattern selects year, month_num, month, sum of total price as monthly_sales, count of rows as montly volume. Here monthly volume denotes the number of items delivered with aggregation over year, month_num, month.
2. To summarize common table expression order_pattern has year, month wise count of orders and sum of prices as sales. Freight value is excluded from the sales as its an expense to the company.
3. Next select query uses order_pattern to derive necessary information. The innermost query selects year, month, month_num, monthly_sales,

	monthly_vol from order_pattern. Additionally it assigns the quarter to the month using case statement. 4. Outer query to the query described in point 4, selects year, month, monthly_sales, monthly_volume, quarter from inner query. In addition to that using window function sum it calculates quarterly sales & volume. 5. The result of outer query is ordered by year, month_num, quarter.
Insights & Recommendation	1. Month on month analysis based on chart viz "Year wise month on month analysis of Sales & Volume" a. As compared to overall data sales and volume were too low in Sep, Oct, Dec 2016. There is no data available for Nov 2016 which shows that Target wasn't operating in Nov 2016. b. Marginally upward trend in sales & volume can be seen from Jan 2017 to Oct 2017. Nov 2016 shows maximum sales & volume among all the months & years across 2016, 2017, 2018. c. Sales & volume remains below average of total sales & volume till Jul 2017. After that it remains at par or above average for all months in a year. d. In year 2018 for month Jan & Mar the sales & volumes are higher than rest of the year. The sales & volume trend remains sidewards till May. After that it shows downward trend. 2. Quarter on quarter analysis based on chart viz "Year wise quarter on quarter analysis of Sales & Volume" a. For volume the trend was upward till Q1 2018 and then volume shows downward trend. b. For sales the trend was upward till Q2 2018 and then the trend is downward. c. There is a sharp fall in sales and volume in Q3 2018.

Images / graphs (P.S. Graphs are created using Tableau. Based on data output from SQL query. No data modifications are done in Tableau.)





Main question	2. In-depth Exploration
Sub question	2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

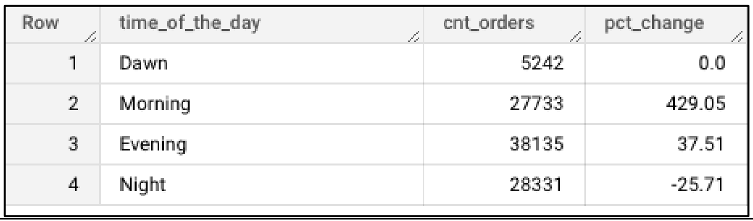
Query

```
with order_day_pattern as(
select time_of_day,
  count(order_id) cnt_orders
from(
select
  case
    when extract(HOUR from o.order_purchase_timestamp)
between 0 and 6 then 0
    when extract(HOUR from o.order_purchase_timestamp)
between 7 and 12 then 1
    when extract(HOUR from o.order_purchase_timestamp)
between 13 and 18 then 2
    when extract(HOUR from o.order_purchase_timestamp)
between 19 and 23 then 3
  end as time_of_day,
  o.order_id
from `target.orders` o
) tab
group by time_of_day
)

select
case
  when time_of_day = 0 then 'Dawn'
  when time_of_day = 1 then 'Morning'
  when time_of_day = 2 then 'Evening'
  when time_of_day = 3 then 'Night'
end as time_of_the_day,
cnt_orders,
 ifnull(
    round(
      ((cnt_orders - LAG(cnt_orders) OVER (ORDER BY
time_of_day)) / LAG(cnt_orders) OVER (ORDER BY
time_of_day)) * 100
      ,2)
    ,0) AS pct_change
 from order_day_pattern
order by time_of_day;
```

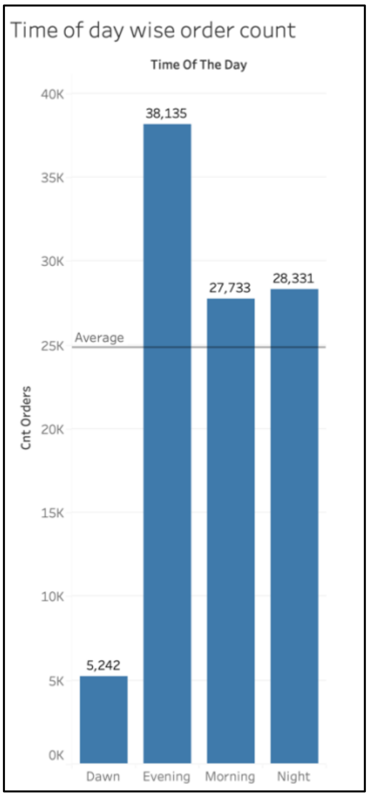
Assumptions

1. Time value in column in order_purchase_timestamp in orders table is having time as per Brazil timezone.

Result screenshot (P.S. if query returns more than 10 rows then screenshot



Row	time_of_the_day	cnt_orders	pct_change
1	Dawn	5242	0.0
2	Morning	27733	429.05
3	Evening	38135	37.51
4	Night	28331	-25.71

shows first 10 rows)	
Explanation	1. order_day_pattern is the common table expression which helps in making query readable. 2. Inner query of order_day_pattern selects order_id and using case statement on hour of order_purchase_timestamp it selects time of day in numeric form. i.e. 0 for hours between 0 to 6, 1 for hours between 7 to 12, 2 for hours between 13 to 18, 3 for hours between 19 to 23. 3. Outer query of order_day_pattern does count of orders aggregated by time of day. 4. Main select query assigns the text value to the numeric time of day using case statement. i.e. Dawn for 0, Morning for 1, Evening for 2, Night for 3 for the each row of order_day_pattern. Also it selects order count order_day_pattern and finds the % difference between previous order count ordered by numeric time of day.
Insights & Recommendation	1. Based on graph it is observed that customers tend to order more in the evening. 2. Based on graph its observed that number of orders at dawn time are below the average count of orders during rest of the times in a day. 3. Based on query outcome its observed that although the orders are highest in evening but % of change in number orders is high between Dawn & Morning. Hence it is recommended to have more human and compute resources to be allocated during this time and those can be gradually decreased during day.
Images / graphs (P.S. Graphs are created using Tableau. Based on data output from SQL query. No data modifications are done in Tableau.)	

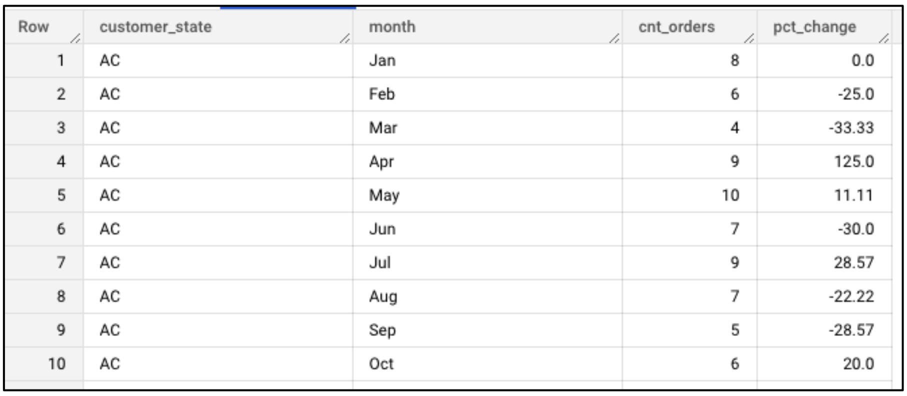
Main question	3. Evolution of E-commerce orders in the Brazil region
Sub question	1. Get month on month orders by states

Query

```
select
  customer_state,
  month,
  cnt_orders,
  ifnull(round(((cnt_orders - LAG(cnt_orders) OVER
(partition by customer_state ORDER BY customer_state,
month_num)) / LAG(cnt_orders) OVER (partition by
customer_state ORDER BY customer_state, month_num)) *
100,2),0) AS pct_change
from (
  select
    c.customer_state,
    extract(MONTH from o.order_purchase_timestamp)
month_num,
    FORMAT_DATE('%b', o.order_purchase_timestamp) month,
    count(order_id) cnt_orders
  from `target.orders` o
  join `target.customers` c on o.customer_id =
c.customer_id
  group by c.customer_state, extract(MONTH from
o.order_purchase_timestamp), FORMAT_DATE('%b',
o.order_purchase_timestamp)
)
order by customer_state, month_num
```

Assumptions

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



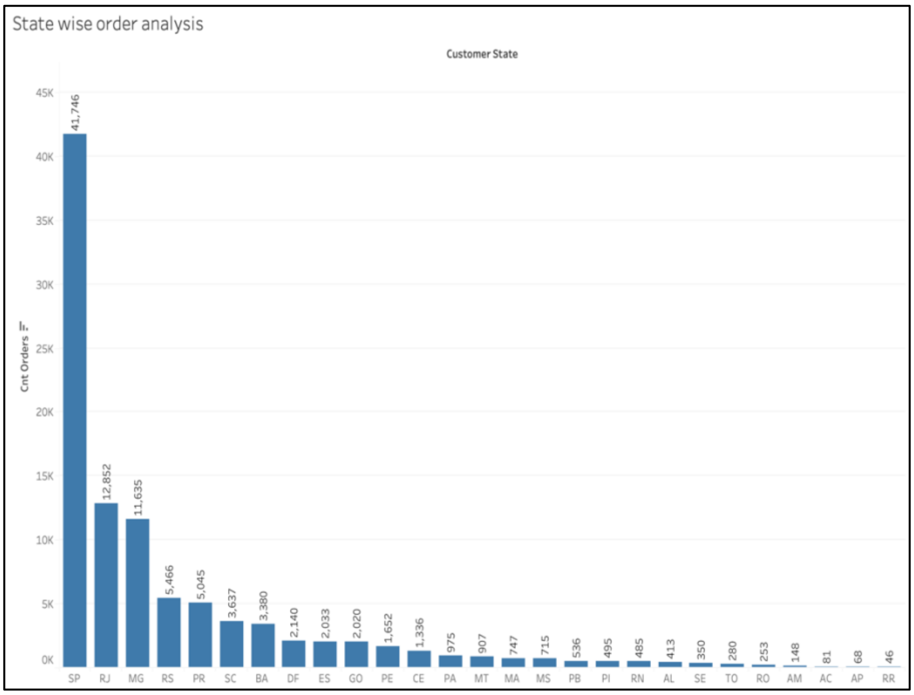
Row	customer_state	month	cnt_orders	pct_change
1	AC	Jan	8	0.0
2	AC	Feb	6	-25.0
3	AC	Mar	4	-33.33
4	AC	Apr	9	125.0
5	AC	May	10	11.11
6	AC	Jun	7	-30.0
7	AC	Jul	9	28.57
8	AC	Aug	7	-22.22
9	AC	Sep	5	-28.57
10	AC	Oct	6	20.0

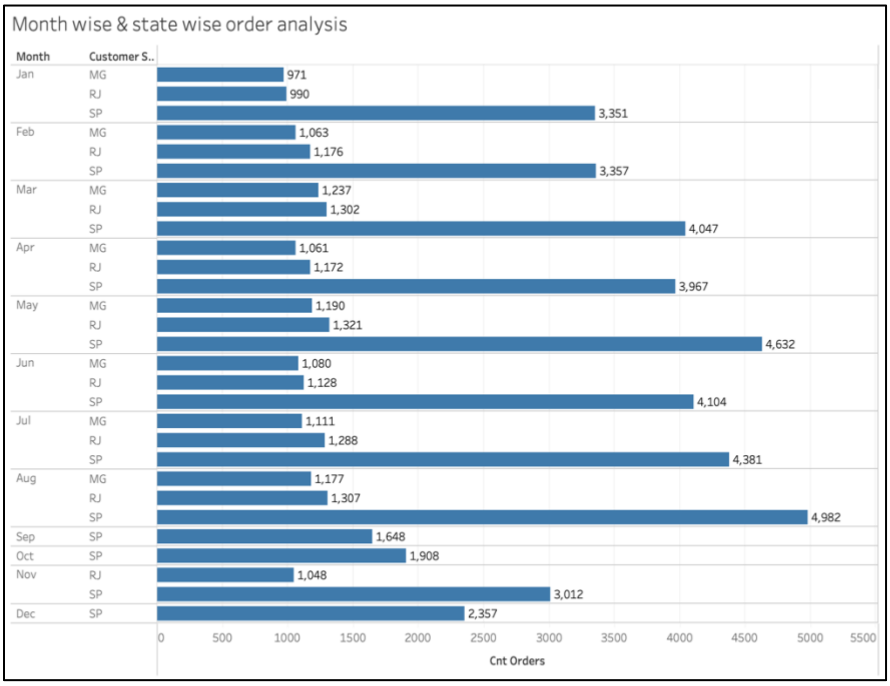
Explanation

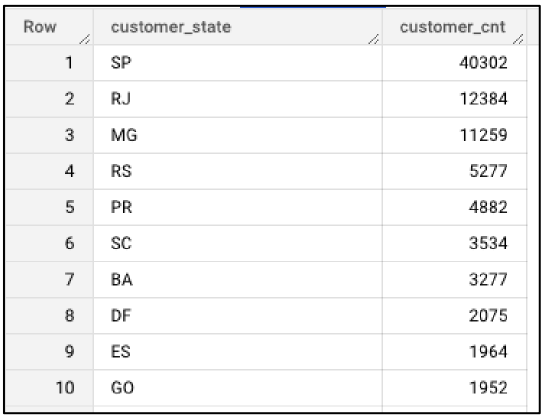
1. The inner query joins orders and customer tables on order_id column. Then it counts the number of orders based on customer_state and month (numeric and abbreviated).

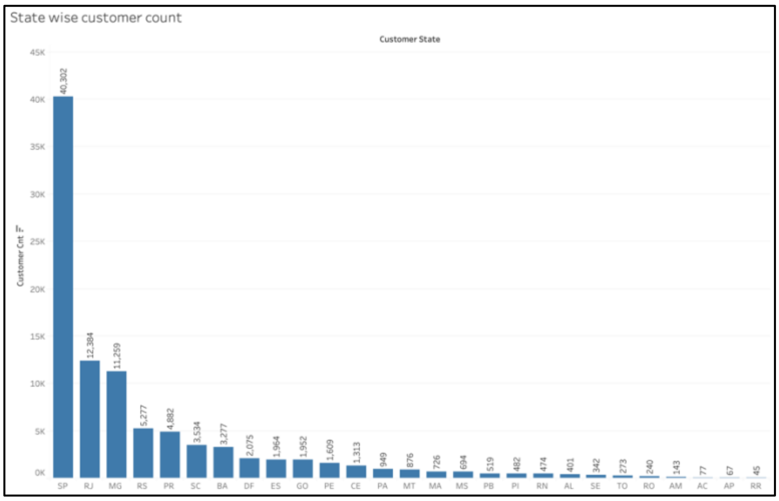
	2. Outer query selects customer_state, month (abbreviated), order count, percentage of changes of order count for each customer_state and sequenced by month. 3. Finally outer query orders the data based on customer_state, month(numeric).
Insights & Recommendation	1. Based on graph State wise order analysis, São Paulo identified by acronym SP is the state having maximum number of orders with huge margin to the next state which is Rio de Janeiro identified by acronym RJ. 2. Based on graph month wise & state wise order analysis, If monthly number of orders greater than 950 are compared for all states and months then its been observed that São Paulo has most orders compared in all the months. 3. It is recommended to focus on customer satisfaction in São Paulo as it's the state which is driving the sales for Target in Brazil. Operations in state Acre acronym identified by acronym AC, Amapá identified by acronym AP, RR identified by acronym Roraima can be shut as total number of orders in this state are less than 100.

Images / graphs (P.S. Graphs are created using Tableau. Based on data output from SQL query. No data modifications are done in Tableau.)

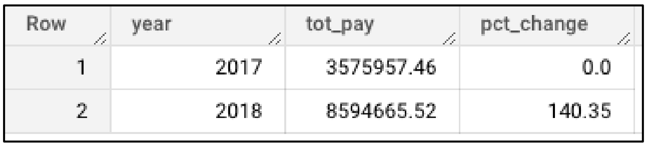

State wise order analysis


Month wise & state wise order analysis

<table>
<tr><td>Main question</td><td>3. Evolution of E-commerce orders in the Brazil region</td></tr>
<tr><td>Sub question</td><td>2. Distribution of customers across the states in Brazil</td></tr>
<tr><td>Query</td><td><pre>
select
    customer_state,
    count(distinct customer_unique_id) customer_cnt
from `target.customers`
group by customer_state
order by customer_cnt desc
</pre></td></tr>
<tr><td>Assumptions</td><td>1. In customers table both "customer_id" and "customer_unique_id" are used to identify customers, the former is used to identify a specific purchase, while the latter is used to identify the individual customer across all transactions.
2. Hence customer_unique_id is used for counting the actual customer. The distinct clause is used for counting the customers.</td></tr>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>
<table>
<tr><th>Row</th><th>customer_state</th><th>customer_cnt</th></tr>
<tr><td>1</td><td>SP</td><td>40302</td></tr>
<tr><td>2</td><td>RJ</td><td>12384</td></tr>
<tr><td>3</td><td>MG</td><td>11259</td></tr>
<tr><td>4</td><td>RS</td><td>5277</td></tr>
<tr><td>5</td><td>PR</td><td>4882</td></tr>
<tr><td>6</td><td>SC</td><td>3534</td></tr>
<tr><td>7</td><td>BA</td><td>3277</td></tr>
<tr><td>8</td><td>DF</td><td>2075</td></tr>
<tr><td>9</td><td>ES</td><td>1964</td></tr>
<tr><td>10</td><td>GO</td><td>1952</td></tr>
</table></td></tr>
<tr><td>Explanation</td><td>1. Query counts customer_id aggregated at customer_state in customers table.</td></tr>
<tr><td>Insights & Recommendation</td><td>1. Based on graph State wise order analysis, São Paulo identified by acronym SP is the state having maximum number of customers with huge margin to the next state which is Rio de Janeiro identified by acronym RJ.
2. It is recommended to focus on customer satisfaction in São Paulo as it’s the state which is driving the sales for Target in Brazil. Operations in state Acre acronym identified by acronym AC, Amapá identified by acronym AP, RR identified by acronym Roraima can be shut as total number of customers in this state are less than 100.</td></tr>
</table>

<table>
<tr>
<td>Images / graphs (P.S. Graphs are created using Tableau. Based on data output from SQL query. No data modifications are done in Tableau.)</td>
<td>


</td>
</tr>
</table>

<table>
<tr><td>Main question</td><td>4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.</td></tr>
<tr><td>Sub question</td><td>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</td></tr>
<tr><td>Query</td><td><pre><code>select
    year,
    tot_pay,
    ifnull(
            round(
                    ((tot_pay - LAG(tot_pay) OVER
(ORDER BY year)) / LAG(tot_pay) OVER (ORDER BY
year)) * 100
                    ,2)
            ,0) AS pct_change
from (
    select
        extract(year from
o.order_purchase_timestamp) year,
        round(sum(payment_value),2) tot_pay
    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
    and o.order_status not in ('canceled',
'unavailable')
    group by extract(year from
o.order_purchase_timestamp)
)
order by year</code></pre></td></tr>
<tr><td>Assumptions</td><td>1. The query excluds cancelled and unavailable order as those won’t contribute to money movement.</td></tr>
<tr><td>Result screenshot
(P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>
<table>
<tr><th>Row</th><th>year</th><th>tot_pay</th><th>pct_change</th></tr>
<tr><td>1</td><td>2017</td><td>3575957.46</td><td>0.0</td></tr>
<tr><td>2</td><td>2018</td><td>8594665.52</td><td>140.35</td></tr>
</table></td></tr>
<tr><td>Explanation</td><td>1. Inner query joins orders and payments table over order_id column. Then it filters the row for year 2017 & 2018 and for months between 1 to 8. Then</td></tr>
</table>

	query filters the records to select records not having cancelled or unavailable status. 2. Upon filtering inner query sums the payment_value aggregated over year extracted from order_purchase_timestamp from orders table. 3. Outer query selects year, sum of payment_value from inner query and finds the percentage increase in total from 2017 to 2018
Insights & Recommendation	NA
Images / graphs	NA

Main question

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

Sub question

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

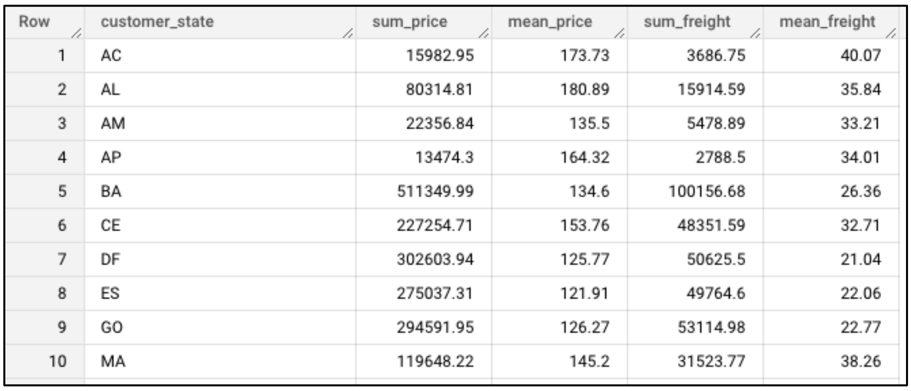
Query

```
select
  customer_state,
  round(sum(price),2) sum_price,
  round(avg(price),2) mean_price,
  round(sum(freight_value),2) sum_freight,
  round(avg(freight_value),2) mean_freight,
from `target.orders` o
join `target.order_items` oi on o.order_id = oi.order_id
join `target.customers` c on o.customer_id = c.customer_id
group by customer_state
order by customer_state
```

Assumptions

1. The query excluds cancelled and unavailable order as those won't contribute to money movement.

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



Row	customer_state	sum_price	mean_price	sum_freight	mean_freight
1	AC	15982.95	173.73	3686.75	40.07
2	AL	80314.81	180.89	15914.59	35.84
3	AM	22356.84	135.5	5478.89	33.21
4	AP	13474.3	164.32	2788.5	34.01
5	BA	511349.99	134.6	100156.68	26.36
6	CE	227254.71	153.76	48351.59	32.71
7	DF	302603.94	125.77	50625.5	21.04
8	ES	275037.31	121.91	49764.6	22.06
9	GO	294591.95	126.27	53114.98	22.77
10	MA	119648.22	145.2	31523.77	38.26

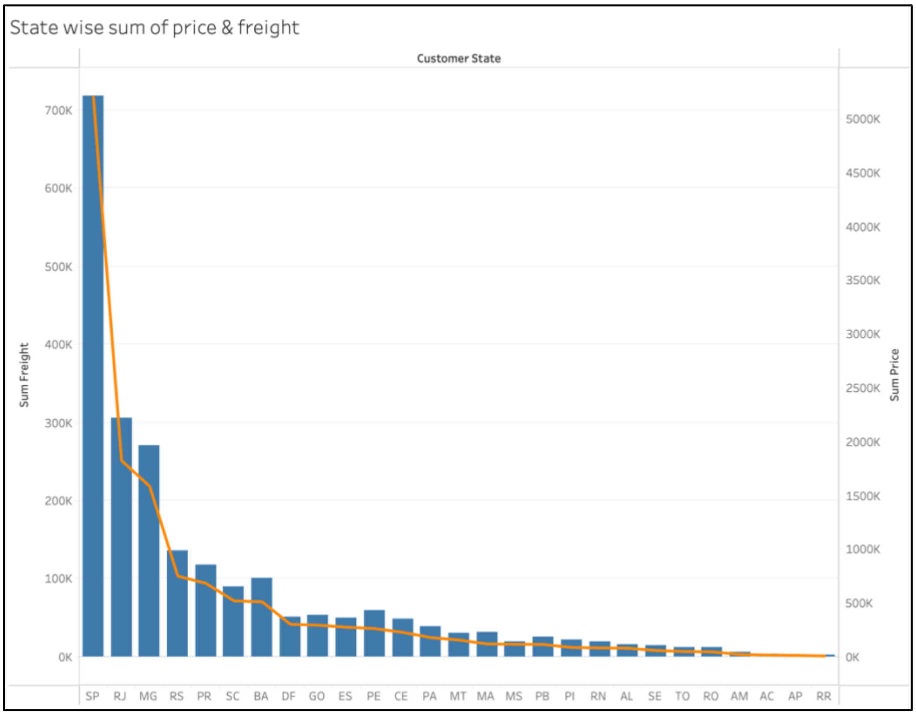
Explanation

1. Query joins orders & order_item table on order_id column. It also joins orders and customers table on customer_id column.
2. The query selects sum of price, average of price, sum of freight, average of freight from order_item table and aggregate it over customer_state.
3. Query displays the records in order of customer state.

Insights & Recommendation

1. Based on graph State wise sum of price & freight, São Paulo identified by acronym SP is the state contributed maximum to the money movement with huge margin to the next state which is Rio de Janeiro identified by acronym RJ.

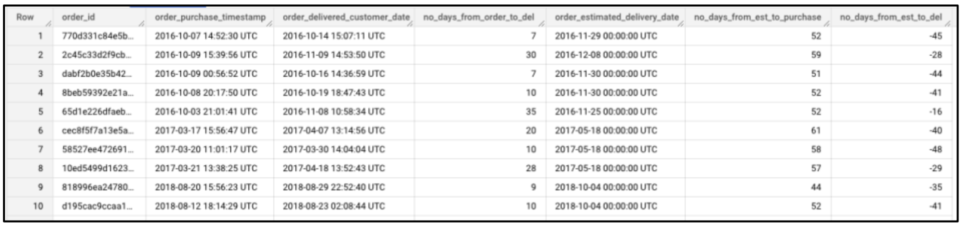
Images / graphs (P.S. Graphs are created using Tableau. Based

on data output from SQL query. No data modifications are done in Tableau.)	

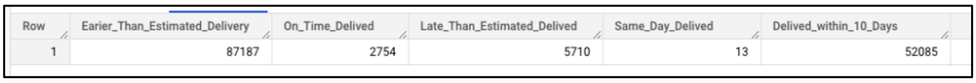
Main question	5. Analysis on sales, freight and delivery time
Sub question	1. Calculate days between purchasing, delivering and estimated delivery

Query

```
select
    order_id,
    order_purchase_timestamp,
    order_delivered_customer_date,
    date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY) no_days_from_order_to_del,
    order_estimated_delivery_date,
    date_diff(order_estimated_delivery_date,
order_purchase_timestamp, DAY)
no_days_from_est_to_purchase,

date_diff(order_delivered_customer_date,order_estimated_del
ivery_date, DAY) no_days_from_est_to_del,
from `target.orders`
where order_purchase_timestamp is not null
and order_delivered_customer_date is not null
and order_estimated_delivery_date is not null
----------------------------------------------------
select
    count(case when no_days_from_est_to_del<0 then 1 else
null end) Earier_Than_Estimated_Delivery,
    count(case when no_days_from_est_to_del=0 then 1 else
null end) On_Time_Delived,
    count(case when no_days_from_est_to_del>1 then 1 else
null end) Late_Than_Estimated_Delived,
    count(case when no_days_from_order_to_del= 0 then 1
else null end) Same_Day_Delived,
    count(case when no_days_from_order_to_del between 1 and
10 then 1 else null end) Delived_within_10_Days,
    from (
        select
            order_id,
            order_purchase_timestamp,
            order_delivered_customer_date,
            date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY) no_days_from_order_to_del,
            order_estimated_delivery_date,
            date_diff(order_estimated_delivery_date,
order_purchase_timestamp, DAY)
no_days_from_est_to_purchase,

date_diff(order_delivered_customer_date,order_estimated_del
ivery_date, DAY) no_days_from_est_to_del,
        from `target.orders`
        where order_purchase_timestamp is not null
        and order_delivered_customer_date is not null
        and order_estimated_delivery_date is not null
)
```

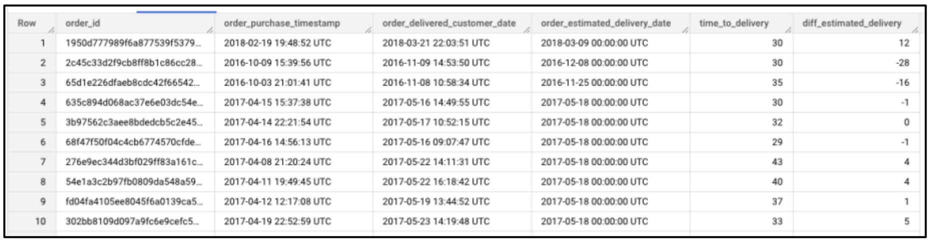
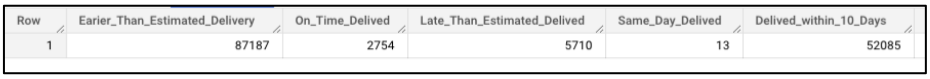
<table>
<tr><td></td><td></td></tr>
<tr><td>Assumptions</td><td>1. Orders with null values for order_purchase_timestamo, order_delivered_customer_date, order_estimated_delivery_date are ignored as those will not be considered for the subtraction.</td></tr>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>


<table>
<tr><th>Row</th><th>order_id</th><th>order_purchase_timestamp</th><th>order_delivered_customer_date</th><th>no_days_from_order_to_del</th><th>order_estimated_delivery_date</th><th>no_days_from_est_to_purchase</th><th>no_days_from_est_to_del</th></tr>
<tr><td>1</td><td>770d331c84e5b...</td><td>2016-10-07 14:52:30 UTC</td><td>2016-10-14 15:07:11 UTC</td><td>7</td><td>2016-11-29 00:00:00 UTC</td><td>52</td><td>-45</td></tr>
<tr><td>2</td><td>2c45c33d2f9cb...</td><td>2016-10-09 15:39:56 UTC</td><td>2016-11-09 14:53:50 UTC</td><td>30</td><td>2016-12-08 00:00:00 UTC</td><td>59</td><td>-28</td></tr>
<tr><td>3</td><td>dabf2b0e35b42...</td><td>2016-10-09 00:56:52 UTC</td><td>2016-10-16 14:36:59 UTC</td><td>7</td><td>2016-11-30 00:00:00 UTC</td><td>51</td><td>-44</td></tr>
<tr><td>4</td><td>8beb59392e21a...</td><td>2016-10-08 20:17:50 UTC</td><td>2016-10-19 18:47:43 UTC</td><td>10</td><td>2016-11-30 00:00:00 UTC</td><td>52</td><td>-41</td></tr>
<tr><td>5</td><td>65d1e226dfaeb...</td><td>2016-10-03 21:01:41 UTC</td><td>2016-11-08 10:58:34 UTC</td><td>35</td><td>2016-11-25 00:00:00 UTC</td><td>52</td><td>-16</td></tr>
<tr><td>6</td><td>cec8f5f7a13e5a...</td><td>2017-03-17 15:56:47 UTC</td><td>2017-04-07 13:14:56 UTC</td><td>20</td><td>2017-05-18 00:00:00 UTC</td><td>61</td><td>-40</td></tr>
<tr><td>7</td><td>58527ee472691...</td><td>2017-03-20 11:01:17 UTC</td><td>2017-03-30 14:04:04 UTC</td><td>10</td><td>2017-05-18 00:00:00 UTC</td><td>58</td><td>-48</td></tr>
<tr><td>8</td><td>10ed5499d1623...</td><td>2017-03-21 13:38:25 UTC</td><td>2017-04-18 13:52:43 UTC</td><td>28</td><td>2017-05-18 00:00:00 UTC</td><td>57</td><td>-29</td></tr>
<tr><td>9</td><td>818996ea24780...</td><td>2018-08-20 15:56:23 UTC</td><td>2018-08-29 22:52:40 UTC</td><td>9</td><td>2018-10-04 00:00:00 UTC</td><td>44</td><td>-35</td></tr>
<tr><td>10</td><td>d195cac9ccaa1...</td><td>2018-08-12 18:14:29 UTC</td><td>2018-08-23 02:08:44 UTC</td><td>10</td><td>2018-10-04 00:00:00 UTC</td><td>52</td><td>-41</td></tr>
</table>



<table>
<tr><th>Row</th><th>Earier_Than_Estimated_Delivery</th><th>On_Time_Delived</th><th>Late_Than_Estimated_Delived</th><th>Same_Day_Delived</th><th>Delived_within_10_Days</th></tr>
<tr><td>1</td><td>87187</td><td>2754</td><td>5710</td><td>13</td><td>52085</td></tr>
</table>
</td></tr>
<tr><td>Explanation</td><td>1. Query selects the record from orders table.
2. Query selects order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date.
3. Query takes difference of order_delivered_customer_date and order_purchase_timestamp as no_days_from_order_to_del.
4. Query takes difference of order_estimated_delivery_date and order_purchase_timestamp as no_days_from_est_to_purchase
5. Query takes difference of order_delivered_customer_date and order_estimated_delivery_date as no_days_from_est_to_del
6. Negative value in no_days_from_est_to_del indicates the delivery is done earlier than estimated time.</td></tr>
<tr><td>Insights & Recommendation</td><td>1. Based on 2nd aggregation query it can be identified that significant number of the orders are getting delivered earlier than estimated delivery date. This indicates that logic required to calculate estimated delivery date needs to be revisited.
2. Based on 2nd aggregation query it can be identified that more than 50% of orders are getting fulfilled between 0-10 days.
3. Based on 2nd aggregation query it is recommended that Target need to look at increasing same day delivery to compete with traditional retail stores.</td></tr>
<tr><td>Images / graphs</td><td>NA</td></tr>
</table>

<table>
<tr><td>Main question</td><td>5. Analysis on sales, freight and delivery time</td></tr>
<tr><td>Sub question</td><td>2. Calculate days between purchasing, delivering and estimated delivery
• time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
• diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_dat</td></tr>
<tr><td>Query</td><td><pre>
select
  order_id,
  order_purchase_timestamp,
  order_delivered_customer_date,
  order_estimated_delivery_date,
  date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY) time_to_delivery,
  date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)
diff_estimated_delivery,
from `target.orders`
where order_purchase_timestamp is not null
and order_delivered_customer_date is not null
and order_estimated_delivery_date is not null
----
select
    count(case when diff_estimated_delivery<0 then 1 else
null end) Earier_Than_Estimated_Delivery,
    count(case when diff_estimated_delivery=0 then 1 else
null end) On_Time_Delived,
    count(case when diff_estimated_delivery>1 then 1 else
null end) Late_Than_Estimated_Delived,
    count(case when time_to_delivery= 0 then 1 else null
end) Same_Day_Delived,
    count(case when time_to_delivery between 1 and 10 then
1 else null end) Delived_within_10_Days,
    from (
      select
        order_id,
        order_purchase_timestamp,
        order_delivered_customer_date,
        order_estimated_delivery_date,
        date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY) time_to_delivery,
        date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)
diff_estimated_delivery,
      from `target.orders`
      where order_purchase_timestamp is not null
      and order_delivered_customer_date is not null
      and order_estimated_delivery_date is not null
)
</pre></td></tr>
</table>

<table>
<tr><td></td><td></td></tr>
<tr><td>Assumptions</td><td>1. Orders with null values for order_purchase_timestamo, order_delivered_customer_date, order_estimated_delivery_date are ignored as those will not be considered for the subtraction.</td></tr>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>

<table>
<tr><th>Row</th><th>order_id</th><th>order_purchase_timestamp</th><th>order_delivered_customer_date</th><th>order_estimated_delivery_date</th><th>time_to_delivery</th><th>diff_estimated_delivery</th></tr>
<tr><td>1</td><td>1950d777989f6a877539f5379...</td><td>2018-02-19 19:48:52 UTC</td><td>2018-03-21 22:03:51 UTC</td><td>2018-03-09 00:00:00 UTC</td><td>30</td><td>12</td></tr>
<tr><td>2</td><td>2c45c33d2f9cb8ff8b1c86cc28...</td><td>2016-10-09 15:39:56 UTC</td><td>2016-11-09 14:53:50 UTC</td><td>2016-12-08 00:00:00 UTC</td><td>30</td><td>-28</td></tr>
<tr><td>3</td><td>65d1e226dfaeb8cdc42f66542...</td><td>2016-10-03 21:01:41 UTC</td><td>2016-11-08 10:58:34 UTC</td><td>2016-11-25 00:00:00 UTC</td><td>35</td><td>-16</td></tr>
<tr><td>4</td><td>635c894d068ac37e6e03dc54e...</td><td>2017-04-15 15:37:38 UTC</td><td>2017-05-16 14:49:55 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>30</td><td>-1</td></tr>
<tr><td>5</td><td>3b97562c3aee8bdedcb5c2e45...</td><td>2017-04-14 22:21:54 UTC</td><td>2017-05-17 10:52:15 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>32</td><td>0</td></tr>
<tr><td>6</td><td>68f47f50f04c4cb6774570cfde...</td><td>2017-04-16 14:56:13 UTC</td><td>2017-05-16 09:07:47 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>29</td><td>-1</td></tr>
<tr><td>7</td><td>276e9ec344d3bf029ff83a161c...</td><td>2017-04-08 21:20:24 UTC</td><td>2017-05-22 14:11:31 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>43</td><td>4</td></tr>
<tr><td>8</td><td>54e1a3c2b97fb0809da548a59...</td><td>2017-04-11 19:49:45 UTC</td><td>2017-05-22 16:18:42 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>40</td><td>4</td></tr>
<tr><td>9</td><td>fd04fa4105ee8045f6a0139ca5...</td><td>2017-04-12 12:17:08 UTC</td><td>2017-05-19 13:44:52 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>37</td><td>1</td></tr>
<tr><td>10</td><td>302bb8109d097a9fc6e9cefc5...</td><td>2017-04-19 22:52:59 UTC</td><td>2017-05-23 14:19:48 UTC</td><td>2017-05-18 00:00:00 UTC</td><td>33</td><td>5</td></tr>
</table>

<table>
<tr><th>Row</th><th>Earier_Than_Estimated_Delivery</th><th>On_Time_Delived</th><th>Late_Than_Estimated_Delived</th><th>Same_Day_Delived</th><th>Delived_within_10_Days</th></tr>
<tr><td>1</td><td>87187</td><td>2754</td><td>5710</td><td>13</td><td>52085</td></tr>
</table>
</td></tr>
<tr><td>Explanation</td><td>1. Query selects the record from orders table.
2. Query selects order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date.
3. Query takes difference of order_delivered_customer_date and order_purchase_timestamp as no_days_from_order_to_del.
4. Query takes difference of order_estimated_delivery_date and order_purchase_timestamp as no_days_from_est_to_purchase
5. Negative value in diff_estimated_delivery indicates the delivery is done earlier than estimated time.</td></tr>
<tr><td>Insights & Recommendation</td><td>1. Based on 2nd aggregation query it can be identified that significant number of the orders are getting delivered earlier than estimated delivery date. This indicates that logic required to calculate estimated delivery date needs to be revisited.
2. Based on 2nd aggregation query it can be identified that more than 50% of orders are getting fulfilled between 0-10 days.
3. Based on 2nd aggregation query it is recommended that Target need to look at increasing same day delivery to compete with traditional retail stores.</td></tr>
<tr><td>Images / graphs</td><td>NA</td></tr>
</table>

<table>
<tr><td>Main question</td><td>5. Analysis on sales, freight and delivery time</td></tr>
<tr><td>Sub question</td><td>3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery</td></tr>
<tr><td>Query</td><td>

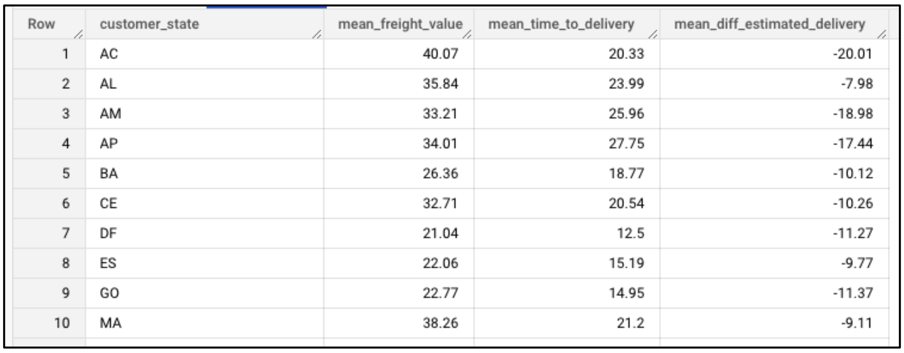
```
select
    c.customer_state,
    round(avg(freight_value),2) mean_freight_value,
    round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2) mean_time_to_delivery,
    round(avg(date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)),2)
mean_diff_estimated_delivery
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 c.customer_state
order by c.customer_state
----------------------------------------------------
select
    customer_state,
    mean_freight_value
from (
    select
        customer_state,
        mean_freight_value,
        dense_rank() over(order by
mean_freight_value desc) as
desc_mean_freight_value_rank,
        dense_rank() over(order by
mean_freight_value) as asc_mean_freight_value_rank
    from (
        select
            c.customer_state,
            round(avg(freight_value),2)
mean_freight_value,

round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2)
mean_time_to_delivery,

round(avg(date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)),2)
mean_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 c.customer_state
    )
```

</td></tr>
</table>

```
)
where desc_mean_freight_value_rank <= 3 or
asc_mean_freight_value_rank <= 3
order by mean_freight_value
--------------------------------------------------
select
    customer_state,
    mean_time_to_delivery
from (
    select
        customer_state,
        mean_time_to_delivery,
        dense_rank() over(order by
mean_time_to_delivery desc) as
desc_mean_time_to_delivery_rank,
        dense_rank() over(order by
mean_time_to_delivery) as
asc_mean_time_to_delivery_rank
    from (
        select
            c.customer_state,
            round(avg(freight_value),2)
mean_freight_value,

round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2)
mean_time_to_delivery,

round(avg(date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)),2)
mean_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 c.customer_state
    )
)
where desc_mean_time_to_delivery_rank <= 3 or
asc_mean_time_to_delivery_rank <= 3
order by mean_time_to_delivery
-------------------------------------------------
select
    customer_state,
    mean_diff_estimated_delivery
from (
    select
        customer_state,
        mean_diff_estimated_delivery,
        dense_rank() over(order by
mean_diff_estimated_delivery desc) as
desc_mean_diff_estimated_delivery_rank,
        dense_rank() over(order by
mean_diff_estimated_delivery) as
asc_mean_diff_estimated_delivery_rank
    from (
        select
            c.customer_state,
            round(avg(freight_value),2)
mean_freight_value,

round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2)
mean_time_to_delivery,

round(avg(date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)),2)
mean_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 c.customer_state
    )
)
where desc_mean_diff_estimated_delivery_rank <= 3 or
asc_mean_diff_estimated_delivery_rank <= 3
order by mean_diff_estimated_delivery
```

Assumptions

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



Row	customer_state	mean_freight_value	mean_time_to_delivery	mean_diff_estimated_delivery
1	AC	40.07	20.33	-20.01
2	AL	35.84	23.99	-7.98
3	AM	33.21	25.96	-18.98
4	AP	34.01	27.75	-17.44
5	BA	26.36	18.77	-10.12
6	CE	32.71	20.54	-10.26
7	DF	21.04	12.5	-11.27
8	ES	22.06	15.19	-9.77
9	GO	22.77	14.95	-11.37
10	MA	38.26	21.2	-9.11

Explanation

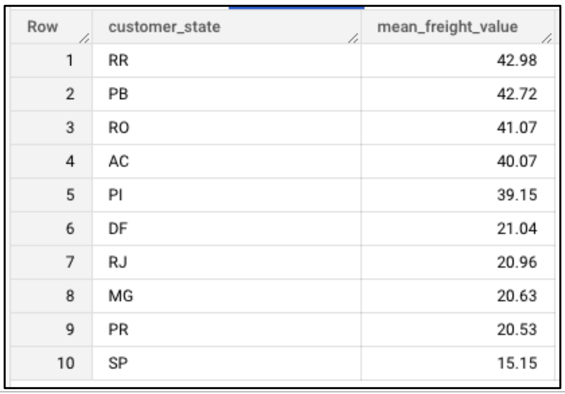
1. Query selects average of freight value, average time to delivery, average time between estimated and delivered date to the customer for each state.
2. Data is ordered by customer state

Insights & Recommendation	1. Using 2nd query its it can be identified that Roraima (RR), Paraíba (PB), Rondônia (RO) are top 3 states and São Paulo, Paraná, Minas Gerais are buttom 3 states for mean freight value. 2. Using 3rd query its can be identified that São Paulo (SP), Paraná(PR), Minas Gerais (MG) are bottom 3 and Amazonas (AM), Amapá(AP), Roraima(RR) are top 3 states for mean time to delivery. 3. Using 4th query its can be identified that Acre(AC), Rondônia (RO), Amazonas (AM) are bottom 3 and Sergipe (SE), Maranhão (MA), Alagoas (AL) are top 3 stated for mean time difference between estimation and actual delivery.
Images / graphs	

<table>
<tr><td>Main question</td><td>5. Analysis on sales, freight and delivery time</td></tr>
<tr><td>Sub question</td><td>5. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5</td></tr>
<tr><td>Query</td><td>

```
with cust_ord_info
as (
    select
        c.customer_state,
        round(avg(freight_value),2)
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 c.customer_state
)

select
    customer_state,
    mean_freight_value
from (
    select customer_state,
    mean_freight_value,
    row_number() over(order by mean_freight_value)
asc_row_num,
    row_number() over(order by mean_freight_value
desc) desc_row_num
    from cust_ord_info x
)
where desc_row_num <= 5 OR asc_row_num <= 5;
```

</td></tr>
<tr><td>Assumptions</td><td>1. Row_number is used to get only top 5 rows even if there are duplicates in top 5 rows.</td></tr>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>



Row	customer_state	mean_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	DF	21.04
7	RJ	20.96
8	MG	20.63
9	PR	20.53
10	SP	15.15

</td></tr>
<tr><td>Explanation</td><td>1. The cust_order_info is the common table expression which has select query. This query joins orders and order_items table on order_id and orders and customers table on customer_id. It calculates & lists customer state wise average freight value.</td></tr>
</table>

	2. Inner query of select query uses cust_order_info to find row_number by average freight value using ascending and decending order in separate columns. It selects customer state and average freight value. Row_number is used to get only top 5 rows even if there are duplicates in top 5 rows. 3. Outer query of select query selects the rows having descending rank or ascending rank less than 5
Insights & Recommendation	As mentioned in the output.
Images / graphs	NA

Main question

5. Analysis on sales, freight and delivery time

Sub question

6. Top 5 states with highest/lowest average time to delivery

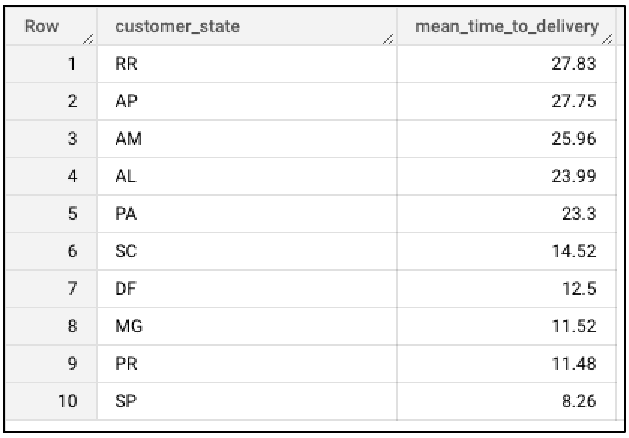
Query

```
with cust_ord_info
as (
    select
        c.customer_state,

round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2)
mean_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 c.customer_state
)

select
    customer_state,
    mean_time_to_delivery
from (
    select
        customer_state,
        mean_time_to_delivery,
        row_number() over(order by
mean_time_to_delivery) asc_row_num,
        row_number() over(order by
mean_time_to_delivery desc) desc_row_num
    from cust_ord_info x
)
where desc_row_num <= 5 OR asc_row_num <= 5;
```

Assumptions

1. Row_number is used to get only top 5 rows even if there are duplicates in top 5 rows.

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



Row	customer_state	mean_time_to_delivery
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3
6	SC	14.52
7	DF	12.5
8	MG	11.52
9	PR	11.48
10	SP	8.26

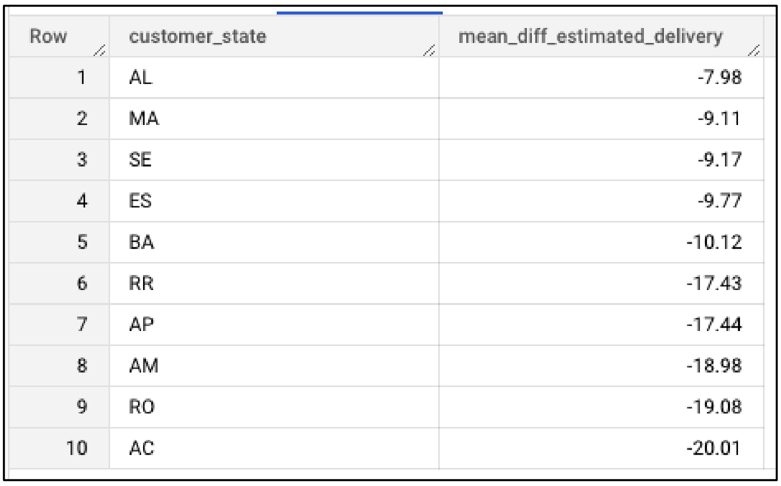
Explanation	1. The cust_order_info is the common table expression which has select query. This query joins orders and order_items table on order_id and orders and customers table on customer_id. It calculates & lists customer state wise average time to delivery. 2. Inner query of select query uses cust_order_info to find row_number by average time to delivery using ascending and decending order in separate columns. It selects customer state and average time to delivry. 3. Outer query of selects query selects the rows having descending rank or ascending rank less than 5
Insights & Recommendation	As per output
Images / graphs	NA

Main question	5. Analysis on sales, freight and delivery time
Sub question	7. Top 5 states where delivery is really fast/ not so fast compared to estimated date

Query

```
with cust_ord_info
as (
    select
        c.customer_state,

round(avg(date_diff(order_delivered_customer_date,
order_estimated_delivery_date, DAY)),2)
mean_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 c.customer_state
)

select customer_state,
mean_diff_estimated_delivery
from (
    select
        customer_state,
        mean_diff_estimated_delivery,
        row_number() over(order by
mean_diff_estimated_delivery) asc_row_num,
        row_number() over(order by
mean_diff_estimated_delivery desc) desc_row_num
    from cust_ord_info x
)
where desc_row_num <= 5 OR asc_row_num <= 5;
```

Assumptions	1. Row_number is used to get only top 5 rows even if there are duplicates in top 5 rows.

<table>
<tr><td>Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)</td><td>
<table>
<tr><th>Row</th><th>customer_state</th><th>mean_diff_estimated_delivery</th></tr>
<tr><td>1</td><td>AL</td><td>-7.98</td></tr>
<tr><td>2</td><td>MA</td><td>-9.11</td></tr>
<tr><td>3</td><td>SE</td><td>-9.17</td></tr>
<tr><td>4</td><td>ES</td><td>-9.77</td></tr>
<tr><td>5</td><td>BA</td><td>-10.12</td></tr>
<tr><td>6</td><td>RR</td><td>-17.43</td></tr>
<tr><td>7</td><td>AP</td><td>-17.44</td></tr>
<tr><td>8</td><td>AM</td><td>-18.98</td></tr>
<tr><td>9</td><td>RO</td><td>-19.08</td></tr>
<tr><td>10</td><td>AC</td><td>-20.01</td></tr>
</table></td></tr>
<tr><td>Explanation</td><td>1. The cust_order_info is the common table expression which has select query. This query joins orders and order_items table on order_id and orders and customers table on customer_id. It calculates & lists customer state wise average days between estimated and delivery date.
2. Inner query of select query uses cust_order_info to find row_number by average average days between estimated and delivery date using ascending and decending order in separate columns. It selects customer state and average time to delivry.
3. Outer query of selects query selects the rows having descending rank or ascending rank less than 5</td></tr>
<tr><td>Insights & Recommendation</td><td>As per query output</td></tr>
<tr><td>Images / graphs</td><td></td></tr>
</table>

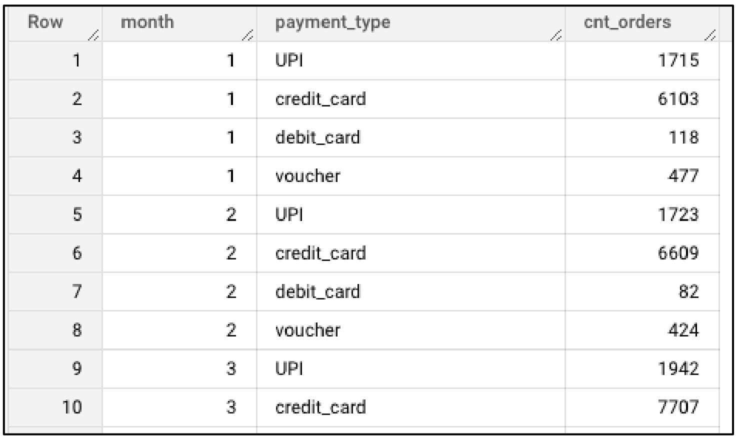
Main question	6. Payment type analysis
Sub question	1. Month over Month count of orders for different payment types

Query

```
select
    month,
    payment_type,count(order_id) cnt_orders
from (
    select
        extract(MONTH from
o.order_purchase_timestamp) month_num,
        FORMAT_DATE('%b',
o.order_purchase_timestamp) month,
        payment_type,
        o.order_id
    from `target.payments` p
    join `target.orders` o on p.order_id =
o.order_id
)
group by month,month_num,payment_type
order by month_num,payment_type
```

Assumptions

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



Row	month	payment_type	cnt_orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707

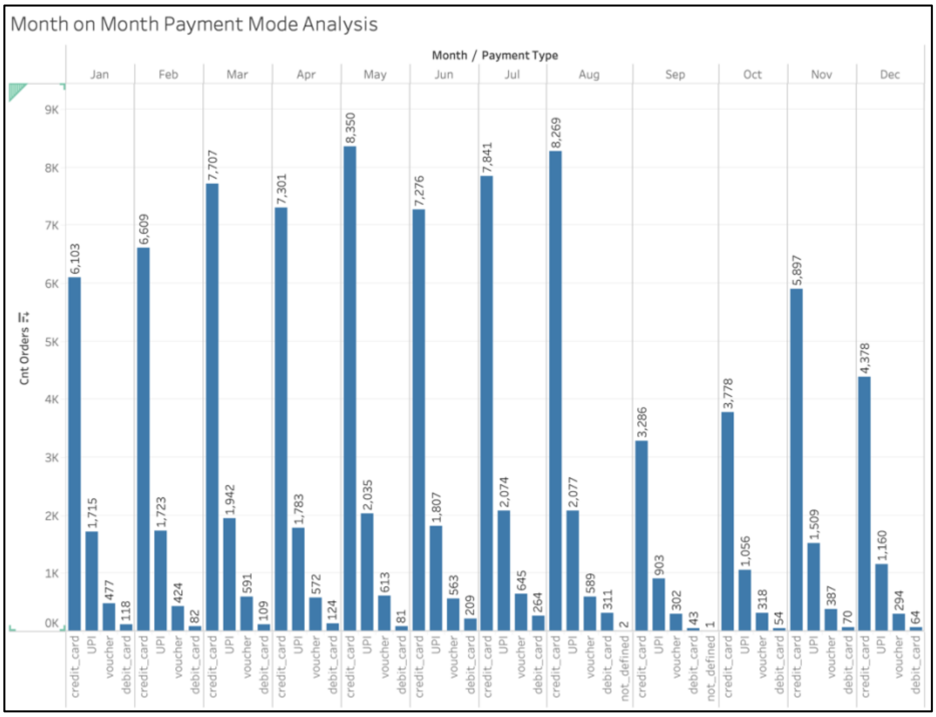
Explanation

1. Inner query joins payments & orders table over order_id.
2. Inner query select month in numeric and abbreviation format, payment_type, order_id.
3. Outer query displays the month in abbreviation format and count of orders group by month & payment type and ordered by month and payment_type.

Insights & Recommendation

1. Credit card is most preferred method of payment.
2. Overall payments are declining every year after Aug.

Images / graphs (P.S. Graphs are created using Tableau. Based on data output from SQL query. No data modifications are done in Tableau.)

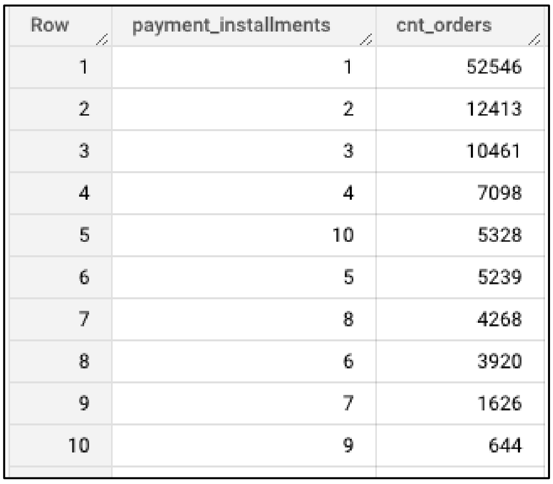


Main question	6. Payment type analysis
Sub question	2. Count of orders based on the no. of payment installments

Query	`select`
	`    payment_installments,`
	`    count(o.order_id) cnt_orders`
	``from `target.payments` p``
	``join `target.orders` o on p.order_id = o.order_id``
	`group by payment_installments`
	`order by count(o.order_id) desc`

Assumptions	

Result screenshot (P.S. if query returns more than 10 rows then screenshot shows first 10 rows)



Row	payment_installments	cnt_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644

Explanation	1. Query selects count of order_id aggregated over payment_installments from payments table joined to orders table over order_id. 2. The result is ordered by count of order_ids
Insights & Recommendation	1. Significant amount of customers prefer to pay in single instalment.
Images / graphs	NA