Author : Priyesh Yadav

# Table of Contents

Author : Priyesh Yadav

# TARGET SQL – BUSINESS CASE SOLUTION

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

- Download all(8) CSV files from Google Drive.

- Create a *New Dataset* called **Target** in Google BigQuery.

- Create tables under the Target dataset by uploading each CSV one by one, using **Upload** & **Auto Detect** schema options.
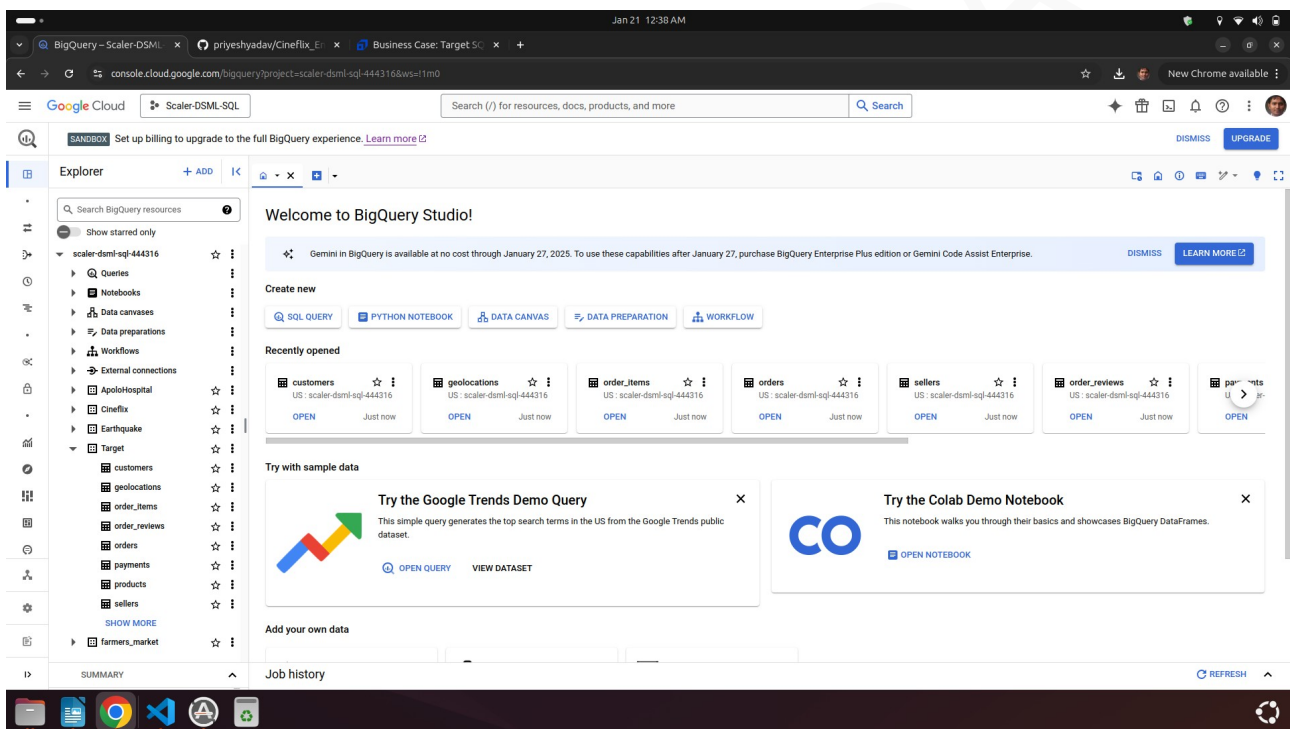


## Data type of all columns in the "customers" table.

- Google BigQuery provides a convenient way to check all columns of a table & their data type.

- Click on Custome Table  & Big Query will open a tab with Table Schema Details.

- Another way is to use *Information_Schema.Columns*

- ***We observe that there are total 5 columns, 4 have string data type while 1 has timestamp***

```
 7
 8
 9
10
11   select Column_Name, Data_type
12   from `Target.INFORMATION_SCHEMA.COLUMNS`
13   where table_name = 'customers'
14
15
16
```

## 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 | |

Author : Priyesh Yadav

## Get the time range between which the orders were placed

- We are looking for Oldest & Recent order datetime, Calculating difference between both of them as well.

- Since the same record is updated in the orders table, we do not filter on order_status

- *We Observe that We have 772 days of Range.*



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

- If we inner join both *Customers* & *Orders* table, than we get only customers who has placed a order, & it will automatically fall into timeframe of Orders.

- Here also we can provide data in 2 ways
    - Individual count of distinct cities & states.
    - Count of cities per state

- *We observe that all customer who ordered something belong to 27 different states which has 4119 distinct cities, We also identified number of cities per state.*

Author : Priyesh Yadav



```
1  select
2  count( distinct C.customer_state) as State_Count,
3  count( distinct C.customer_city) as City_Count
4  from `Target.customers` C
5  join `Target.orders` O on C.customer_id = O.customer_id;
6
7
8
```

## Query results

JOB INFORMATION    RESULTS    CHART    JSON    EXECUTION DETAILS    E

| Row | State_Count | City_Count |
|---|---|---|
| 1 | 27 | 4119 |

```
13
14  select
15  C.customer_state,
16  count( distinct C.customer_city) as City_Count
17  from `Target.customers` C
18  join `Target.orders` O on C.customer_id = O.customer_id
19  group by 1 order by 2 desc;
20
```

Press Alt+

## Query results

SAVE RESULTS

JOB INFORMATION    RESULTS    CHART    JSON    EXECUTION DETAILS    EXECUTION GRAPH

| Row | customer_state | City_Count |
|---|---|---|
| 1 | MG | 745 |
| 2 | SP | 629 |
| 3 | RS | 379 |
| 4 | PR | 364 |
| 5 | BA | 353 |
| 6 | SC | 240 |
| 7 | GO | 178 |
| 8 | CE | 161 |
| 9 | PE | 152 |
| 10 | RJ | 149 |
| 11 | MA | 122 |
| 12 | MT | 101 |
| 13 | ES | 95 |
| 14 | PB | 92 |
| 15 | RN | 90 |
| 16 | PA | 80 |
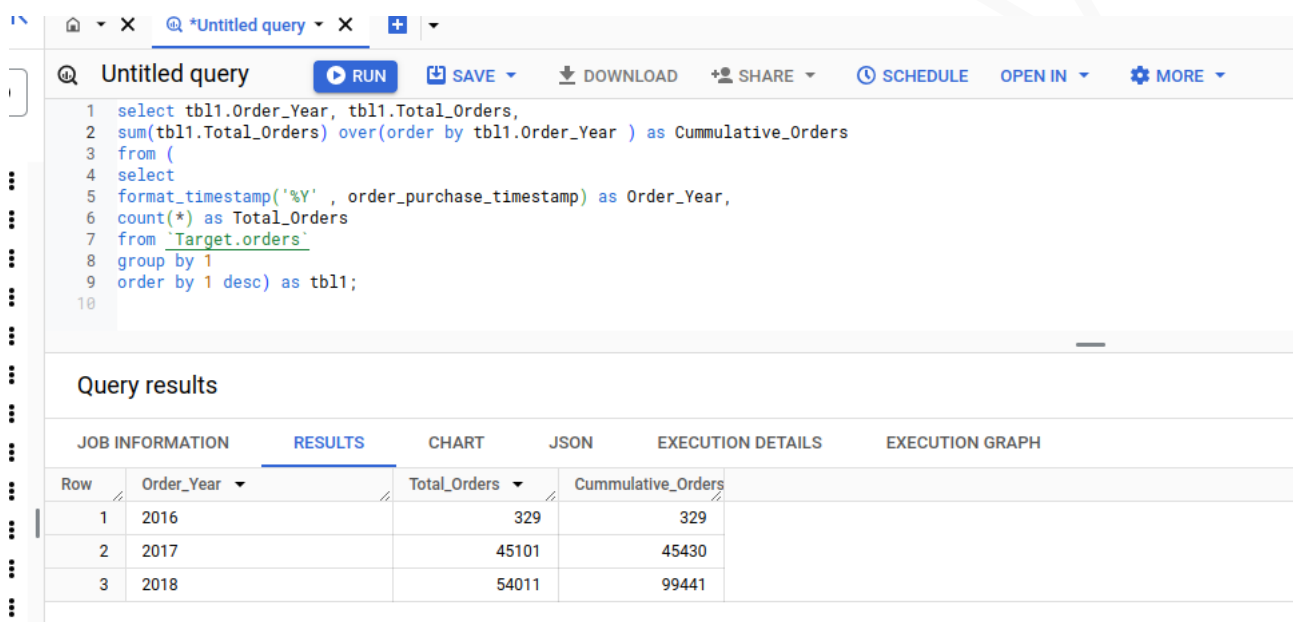
Results per page: 50   1 – 27 of 27

Author : Priyesh Yadav

# In-depth Exploration

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

- To identify yearly tread we should get the number of orders *per year* or *per month per year.*

- Sort the Orders & calculate thier cumulative sum year on year

- Since we are considering trend of order placed , we should even consider canceled(625) and unavailable(609) orders

- ***We observe a Posotive Yearly Orders count pattern.***

Author : Priyesh Yadav

```
   Untitled query      ● RUN    ⊟ SAVE ▼    ⬇ DOWNLOAD   ⊕ SHARE ▼    ⏱ SCHEDULE   OPEN IN ▼   ⚙ MORE ▼
1  select tbl1.Order_Year_Month, tbl1.Total_Orders,
2  sum(tbl1.Total_Orders) over(order by tbl1.Order_Year_Month ) as Cummulative_Orders
3  from (
4  select
5  format_timestamp('%Y-%m' , order_purchase_timestamp) as Order_Year_Month,
6  count(*) as Total_Orders
7  from `Target.orders`
8  group by 1
9  order by 1 desc) as tbl1;
                                                                                          Press A
```

**Query results**                                                                    ⊡ SAVE RESULTS ▼

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

| Row | Order_Year_Month ▼ | Total_Orders ▼ | Cummulative_Orders |
|-----|--------------------|----------------|--------------------|
| 1 | 2016-09 | 4 | 4 |
| 2 | 2016-10 | 324 | 328 |
| 3 | 2016-12 | 1 | 329 |
| 4 | 2017-01 | 800 | 1129 |
| 5 | 2017-02 | 1780 | 2909 |
| 6 | 2017-03 | 2682 | 5591 |
| 7 | 2017-04 | 2404 | 7995 |
| 8 | 2017-05 | 3700 | 11695 |
| 9 | 2017-06 | 3245 | 14940 |
| 10 | 2017-07 | 4026 | 18966 |
| 11 | 2017-08 | 4331 | 23297 |
| 12 | 2017-09 | 4285 | 27582 |
| 13 | 2017-10 | 4631 | 32213 |
| 14 | 2017-11 | 7544 | 39757 |

Results per page:    50 ▼    1 – 25 of 25

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

- To identify monthly seasonality , We should group order bases on month.

- Since seasons ( e.g. winter/summer ) are based on months only , we should not consider Year-Month combination.

- Again we are considering all order irrespective of order_status

- *We observe that Monthly Seasonality*

   ○ *Months May, June, July & August are best in terms of orders placed.*

   ○ *Month Sepember, October, November  & December are worst in terms of orders placed.*

   ○ *Orders decline to lowest in November, While we receive most orders in August.*

```
   Untitled query      ● RUN    ⊟ SAVE ▼    ⬇ DOWNLOAD   ⊕ SHARE ▼    ⏱ SCHEDULE   OPEN IN ▼   ⚙ MORE ▼
1  select tbl1.Order_Year_Month, tbl1.Total_Orders,
2  sum(tbl1.Total_Orders) over(order by tbl1.Order_Year_Month ) as Cummulative_Orders
3  from (
4  select
5  format_timestamp('%Y-%m' , order_purchase_timestamp) as Order_Year_Month,
6  count(*) as Total_Orders
7  from `Target.orders`
8  group by 1
9  order by 1 desc) as tbl1;
```

Author : Priyesh Yadav



```
2   tbl1.Order_Month, tbl1.Total_Orders,
3   concat(round(ifnull(100*(tbl1.Total_Orders - lag(tbl1.Total_Orders) over(Order by Order_Month asc ))/tbl1.Total_Orders,0),2),'%') as Prev_Montly_Comparision
4   from
5   (select
6   format_timestamp('%m' , order_purchase_timestamp) as Order_Month,
7   count(*) as Total_Orders,
8   from `Target.orders`
9   group by 1
10  ) as tbl1
11  order by 1 asc
12
```

**Query results**

| Row | Order_Month | Total_Orders ▾ | Prev_Montly_Comparision ▾ |
|---|---|---|---|
| 1 | 01 | 8069 | 0% |
| 2 | 02 | 8508 | 5.16% |
| 3 | 03 | 9893 | 14% |
| 4 | 04 | 9343 | -5.89% |
| 5 | 05 | 10573 | 11.63% |
| 6 | 06 | 9412 | -12.34% |
| 7 | 07 | 10318 | 8.78% |
| 8 | 08 | 10843 | 4.84% |
| 9 | 09 | 4305 | -151.87% |
| 10 | 10 | 4959 | 13.19% |
| 11 | 11 | 7544 | 34.27% |
| 12 | 12 | 5674 | -32.96% |

## During what time of the day, do the Brazilian customers mostly place their orders

- Since the whole data is for Brazilian customer, There is no need to identify Brazilian Customer.

- First we need to extract the Hour part form the Order_Purchase_Timestamp.

- Next, Create categories with Case statement with given range of hours.

- Next, We count order placed in each category.

- As our final output requirement is 1 row, we limit 1 rows after sorting data based on order placed in each category in descending order.

- *We observe that Brazilian customer mostly place thier orders in Afternoon.*

```
4  select
5  case
6  when extract(HOUR from order_purchase_timestamp) between 0 and 6 then 'Dawn'
7  when extract(HOUR from order_purchase_timestamp) between 7 and 12 then 'Mornings'
8  when cast(format_timestamp('%H', order_purchase_timestamp) as INT64) between 13 and 18 then 'Afternoon'
9  else 'Night'
10 end as Time_of_the_Day,
11 count(*) as No_Of_Ordes
12 from `Target.orders`
13 Group by 1 order by 2 desc limit 1;
14
15
16
```

Query results

JOB INFORMATION     RESULTS     CHART     JSON     EXECUTION DETAILS     EXECUTION GRAPH

| Row | Time_of_the_Day ▼ | No_Of_Ordes ▼ |
|-----|-------------------|---------------|
| 1   | Afternoon         | 38135         |

# Evolution of E-commerce orders in the Brazil region

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

- Orders table do not have *State* column.

- We should *left join* with *customers* table , so that we get State details, also retain data for orders.

- We can work with null State column separately. (*Curretly we dont have such data.)*

- We can represent this data in two format.

- ***We are able to observe the month on month orders placed in each state, however First solution provides a good pivotal view.***

Author : Priyesh Yadav



```sql
select
C.customer_state,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 1 then 1 else 0 end ) as Jan,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 2 then 1 else 0 end ) as Feb,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 3 then 1 else 0 end ) as Mar,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 4 then 1 else 0 end ) as Apr,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 5 then 1 else 0 end ) as May,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 6 then 1 else 0 end ) as Jun,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 7 then 1 else 0 end ) as Jul,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 8 then 1 else 0 end ) as Aug,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 9 then 1 else 0 end ) as Sep,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 10 then 1 else 0 end ) as Oct,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 11 then 1 else 0 end ) as Nov,
sum(case when extract(MONTH from O.order_purchase_timestamp) = 12 then 1 else 0 end ) as Dec
from `Target.orders` O
left join `Target.customers` C on O.customer_id = C.customer_id
group by 1;
```

Query results

| Row | customer_state | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | RJ | 990 | 1176 | 1302 | 1172 | 1321 | 1128 | 1288 | 1307 | 612 | 725 | |
| 2 | RS | 427 | 473 | 569 | 488 | 559 | 526 | 565 | 599 | 279 | 276 | |
| 3 | SP | 3351 | 3357 | 4047 | 3967 | 4632 | 4104 | 4381 | 4982 | 1648 | 1908 | |
| 4 | DF | 151 | 196 | 207 | 183 | 208 | 220 | 243 | 232 | 97 | 104 | |
| 5 | PR | 443 | 460 | 504 | 500 | 524 | 478 | 523 | 556 | 183 | 225 | |
| 6 | MT | 96 | 84 | 71 | 92 | 104 | 83 | 85 | 78 | 35 | 55 | |
| 7 | MA | 66 | 67 | 77 | 73 | 65 | 59 | 79 | 70 | 42 | 52 | |

Results per page: 50 ▼  1 – 27 of 27

```sql
select C.customer_state, extract(MONTH from O.order_purchase_timestamp) as Order_Month,
count(*) as Orders
from `Target.orders` O
left join `Target.customers` C on O.customer_id = C.customer_id
group by 1,2
order by 1, 2 ASC;
```

Query results
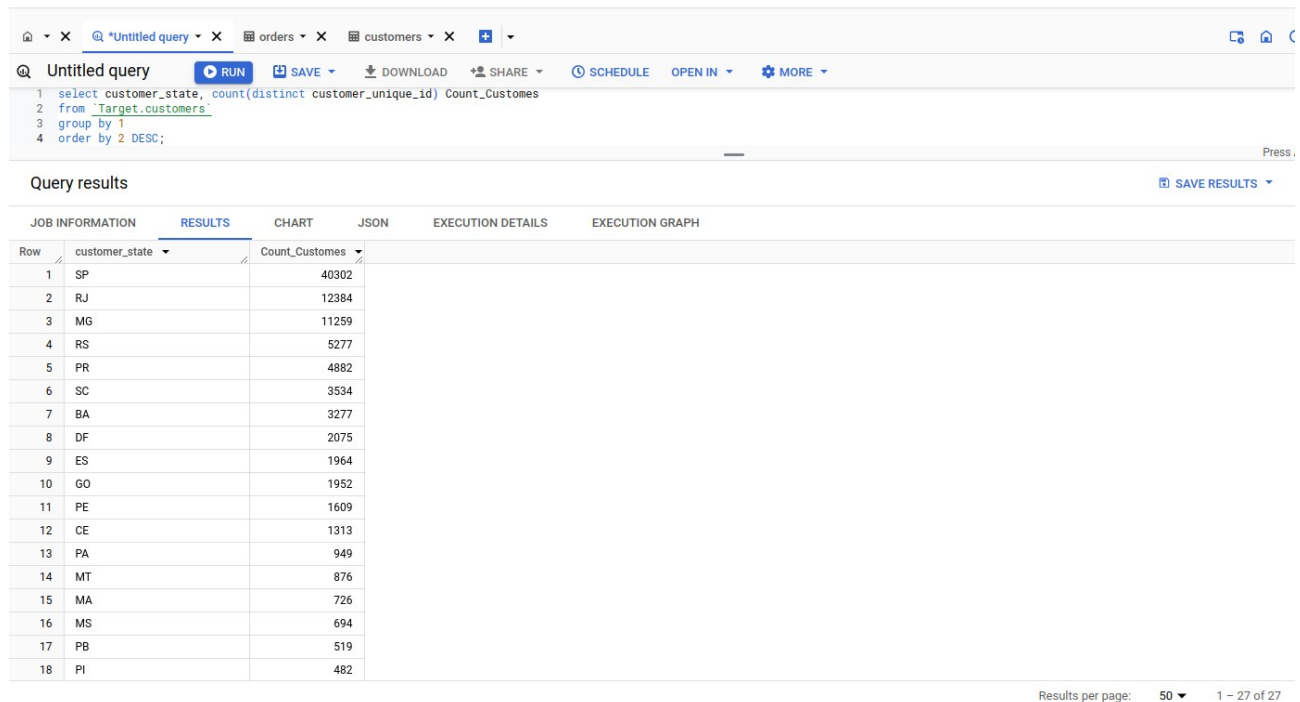
| Row | customer_state | Order_Month | Orders |
|---|---|---|---|
| 1 | AC | 1 | 8 |
| 2 | AC | 2 | 6 |
| 3 | AC | 3 | 4 |
| 4 | AC | 4 | 9 |
| 5 | AC | 5 | 10 |
| 6 | AC | 6 | 7 |
| 7 | AC | 7 | 9 |
| 8 | AC | 8 | 7 |
| 9 | AC | 9 | 5 |
| 10 | AC | 10 | 6 |
| 11 | AC | 11 | 5 |
| 12 | AC | 12 | 5 |
| 13 | AL | 1 | 39 |
| 14 | AL | 2 | 39 |
| 15 | AL | 3 | 40 |
| 16 | AL | 4 | 51 |

Results per page: 50 ▼  1 – 50 of 322

## How are the customers distributed across all the states

- There are two columns, *customer_id* and *customer_unique_id,* But upon checking with *customers* & *orders*, we identify that each customer_unique_id has multiple customer_id which is directly related to order_id.

- Hence taking the count of customer_unique_id group by will provide the correct result.

Author : Priyesh Yadav
- ***We Observe Customer distribution across state. SP , RJ & MG are top 3 contributing states.***



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

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

- We identify that Orders & Purchase table has 1 to many relationshiop.

- Purchase table do not have any date column, we can get purchase date details by combining the Orders & Purchase table over order_id.

- We aggregate the purchase_amount over given year & month commbination in a inner query.

Author : Priyesh Yadav
- We use lag function to get next value & calculate differnce. Since we are dealing with only 2 year, One of the output for lag will be null.

- We limit our Rows to only 1 to showcase percentage increase between 2017 & 2018.

- *We observe Percentage Increase for Cost of Orders , Between January to Auguest months of 2017 & 2018 is almost 138%.*



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

- To get order price statewirse, we need to join Payment, Orders & Customers.

- Since one order can have multiple payment, Hence calculating average using payments table directly is not a good options.

- First we create a inner query to calculate Order value per order. Combine it with Statedata & then calculate Statewise Total & Average.

- *We observe total & average order price for each state.*

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

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

- Freight_value is presnet in order_items table, which has multiple rows for a single order, So we should first create a total freight_value for each order.

- We further join it with orders & customer table to get to State for each order.

- Finally we can aggregate required values at state level.

- *We observe total & average order freight value per state.*

```sql
1  select C.customer_state, round(sum(tbl_Freight.Freight_by_Order),2) as Total_Freight, round(avg(tbl_Freight.Freight_by_Order),2) as Average_Freigh
2  from `Target.customers` C
3  join `Target.orders` O on C.customer_id = O.customer_id
4  join (
5  select order_id, round(sum(freight_value),2) as Freight_by_Order
6  from `Target.order_items`
7  group by 1 ) as tbl_Freight
8  on O.order_id = tbl_Freight.order_id
9  group by 1 order by 1
```

| Row | customer_state | Total_Freight | Average_Freigh |
|-----|----------------|---------------|----------------|
| 1 | AC | 3686.75 | 45.52 |
| 2 | AL | 15914.59 | 38.72 |
| 3 | AM | 5478.89 | 37.27 |
| 4 | AP | 2788.5 | 41.01 |
| 5 | BA | 100156.68 | 29.83 |
| 6 | CE | 48351.59 | 36.44 |
| 7 | DF | 50625.5 | 23.82 |
| 8 | ES | 49764.6 | 24.58 |
| 9 | GO | 53114.98 | 26.46 |
| 10 | MA | 31523.77 | 42.6 |
| 11 | MG | 270853.46 | 23.46 |
| 12 | MS | 19144.03 | 27.0 |
| 13 | MT | 29715.43 | 32.91 |
| 14 | PA | 38699.3 | 39.9 |

Results per page: 50    1 – 27 of 27

# Analysis based on sales, freight and delivery time

**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.**

- Intial inspection of *Orders* table suggest that customer delivery date is only applicable for order with either *delivered* or *canceled* status.

- We are considering *6 canceled* order which has customer delivery date populated for analysis.

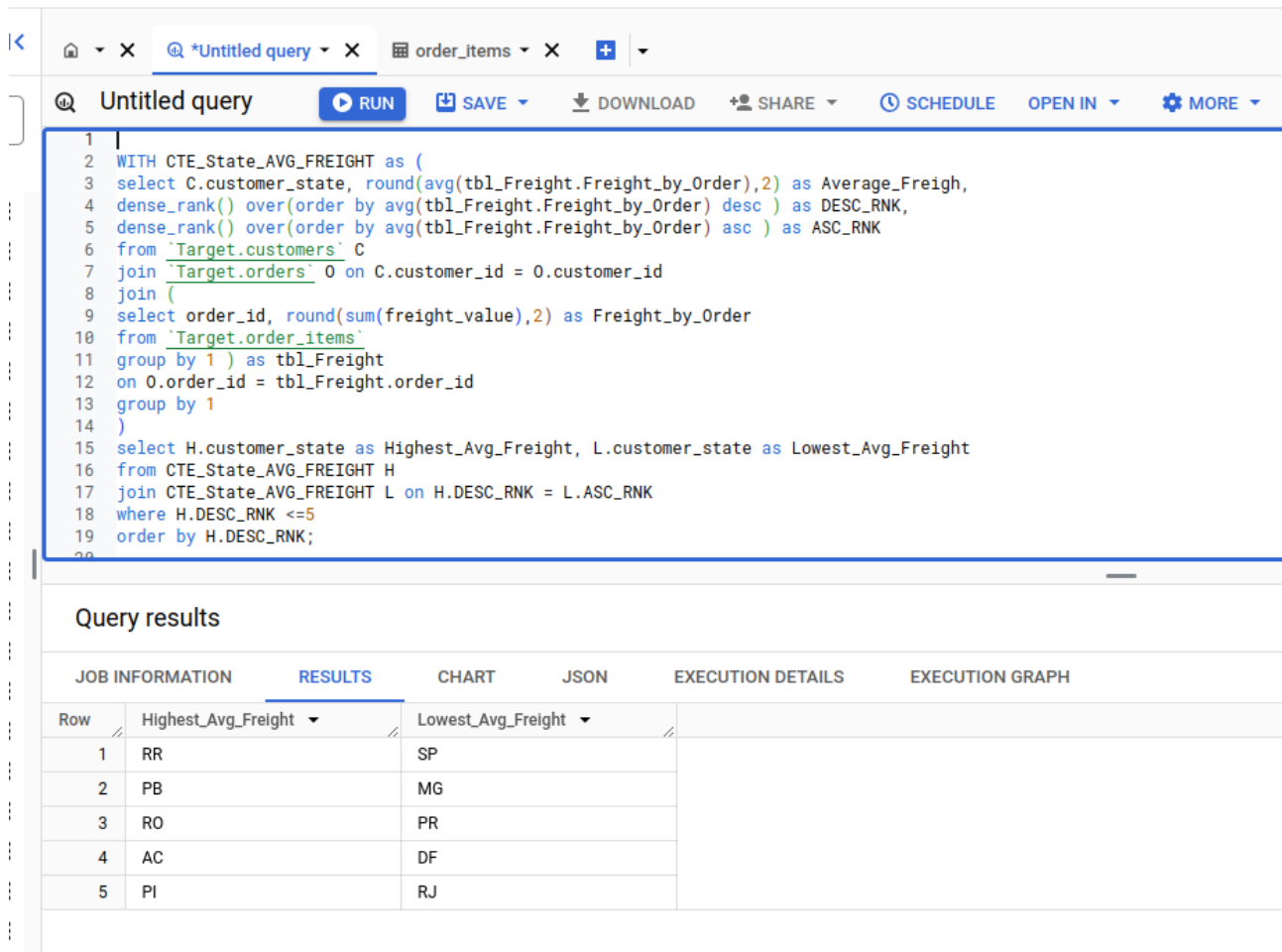- *We observe that a lot of the orders has been delivered very late.*

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

- We had already calculated statewise average freight ealrier, we can extend same query to identify top 5 & low 5.

- However since we want to display both in adjacement column, we can first calculate state rank over average freight value in both orders & self join to display data in adjancemt columns.

- *We observe Top 5 & Low states as per average frieght rate.*

```
1
2  WITH CTE_State_AVG_FREIGHT as (
3  select C.customer_state, round(avg(tbl_Freight.Freight_by_Order),2) as Average_Freigh,
4  dense_rank() over(order by avg(tbl_Freight.Freight_by_Order) desc ) as DESC_RNK,
5  dense_rank() over(order by avg(tbl_Freight.Freight_by_Order) asc ) as ASC_RNK
6  from `Target.customers` C
7  join `Target.orders` O on C.customer_id = O.customer_id
8  join (
9  select order_id, round(sum(freight_value),2) as Freight_by_Order
10 from `Target.order_items`
11 group by 1 ) as tbl_Freight
12 on O.order_id = tbl_Freight.order_id
13 group by 1
14 )
15 select H.customer_state as Highest_Avg_Freight, L.customer_state as Lowest_Avg_Freight
16 from CTE_State_AVG_FREIGHT H
17 join CTE_State_AVG_FREIGHT L on H.DESC_RNK = L.ASC_RNK
18 where H.DESC_RNK <=5
19 order by H.DESC_RNK;
```

Query results

| Row | Highest_Avg_Freight ▾ | Lowest_Avg_Freight ▾ |
|-----|------------------------|----------------------|
| 1 | RR | SP |
| 2 | PB | MG |
| 3 | RO | PR |
| 4 | AC | DF |
| 5 | PI | RJ |

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

- We use the same logic that we have used for average freight value per state.

- *We observe top 5 & low 5 state with average delivery time.*

Author : Priyesh Yadav



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

Author : Priyesh Yadav



```
1   With CTE_FAST_DEL as
2   ( select C.customer_state as State, round(avg(date_diff(O.order_estimated_delivery_date,O.order_delivered_customer_date, DAY)),2) as Fast_del
3   from `Target.orders` O
4   join `Target.customers` C on O.customer_id = C.customer_id
5   where O.order_delivered_customer_date is not null
6   group by 1 ),
7   CTE_FAST_DEL_RNK as
8   (
9     select State, Fast_del ,
10    dense_rank() over(order by Fast_del asc) as H_RANK,
11    from CTE_FAST_DEL
12  )
13  select State from CTE_FAST_DEL_RNK
14  where H_rank <=5
15  order by H_Rank
16  ;
```

**Query results**

JOB INFORMATION     RESULTS     CHART     JSON     EXECUTION DETAILS     EXECUTION GRAPH

| Row | State |
| --- | --- |
| 1 | AL |
| 2 | MA |
| 3 | SE |
| 4 | ES |
| 5 | BA |

# Analysis based on the payments

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

- We analyse that one order can have multiple payments & payment type.

- We cnsider unique combination of order_id, Order_date & payment_type for further analysis.

- Order with multiple payment type ( card & voucher ) will be counted in both payment type.

- We are consider only order where payment_value > 0

- *We Observe positive month on month trend.*

Author : Priyesh Yadav



```sql
1  WITH CTE_Order_Payment_type_combination as (
2  select distinct O.order_id,
3  format_timestamp('%Y-%m',order_purchase_timestamp) as order_purchase,
4  P.payment_type
5  from `Target.orders` O
6  join `Target.payments` P on O.order_id = P.order_id
7  where P.payment_value > 0
8  ),
9  CTE_Date_payment_type_count as (
10   select order_purchase, payment_type, count(*) as No_of_Order
11   from CTE_Order_Payment_type_combination
12   group by 1, 2
13   order by 1, 2
14 )
15 select payment_type, order_purchase, No_of_Order,
16 sum(No_of_Order) over (partition by payment_type order by order_purchase) as month_on_month_Order
17 from CTE_Date_payment_type_count
18 order by 1, 2
```

| Row | payment_type | order_purchase | No_of_Order | month_on_month_Or |
|-----|--------------|----------------|-------------|-------------------|
| 1 | UPI | 2016-10 | 63 | 63 |
| 2 | UPI | 2017-01 | 197 | 260 |
| 3 | UPI | 2017-02 | 398 | 658 |
| 4 | UPI | 2017-03 | 590 | 1248 |
| 5 | UPI | 2017-04 | 496 | 1744 |
| 6 | UPI | 2017-05 | 772 | 2516 |
| 7 | UPI | 2017-06 | 707 | 3223 |
| 8 | UPI | 2017-07 | 845 | 4068 |
| 9 | UPI | 2017-08 | 938 | 5006 |

Results per page: 50 ▼   1 – 50 of 88

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

- Order where installments are opted by customer has payment type as *Credit Card*.

- We need to only include playment that has *payment_value > 0, Payment_installments > 1*

- *Atleast 2 payment* enries are present.

- **We observe orders on installement where atleast 1 emi has been paid.**



```sql
1  select payment_installments, count(distinct order_id) as order_count from
2  `Target.payments`
3  where
4  payment_type = 'credit_card'
5  and
6  payment_installments > 1 and payment_value > 0
7  group by 1
8  having count(order_id) > 1;
```

| Row | payment_installment | order_count |
|-----|---------------------|-------------|
| 1 | 2 | 12389 |
| 2 | 3 | 10443 |
| 3 | 4 | 7088 |
| 4 | 5 | 5234 |
| 5 | 6 | 3916 |
| 6 | 7 | 1623 |
| 7 | 8 | 4253 |
| 8 | 9 | 644 |
| 9 | 10 | 5315 |
| 10 | 11 | 23 |
| 11 | 12 | 133 |
| 12 | 13 | 16 |
| 13 | 14 | 15 |
| 14 | 15 | 74 |
| 15 | 16 | 5 |

Results per page: 50 ▼   1 – 20 of 20

Author : Priyesh Yadav

# Actionable Recommendation

1) Target should try to increase city center in state where overall city center count is less than 100. 12 State has more than 100 city centers while 15 State has less than 100 city centers.

2) Target should try to identify cause of rapid drop in customer order starting from September till December.

3) Most Suitable time for Ad compaign & new launches seems be between May & August.

4) Since Orders placed in Dawn period are very few (5.27%), It could be utilized for Production issue fixes & maintenance activities.

5) Target should get rid of Customer_Id column in Customer table, With Each new order a New Customer_Id is generated which is tagged to Customer_Unique_id. Because of this the size of customer table will continue to grow parallel to new orders. Unless its fulfilling some other purpose which we have not discovered based on this Case Study.

6) Target should try to increase customer base in state where customer count is less than 2000, Currenlty 19 State cout of 27 have less than 2000 customers.

7) Target should improvise on delivering the orders faster, 63 order out of 96476 were delivered in over 100 days.

8) Target should improve on providing the estimated delivery date, Only 2754 orders were delivery on estimated delivery date,

9) Target should improve average delivery time against expected delivery time in 5 states where it is more than 15 days.