PROJECT —TARGET DATASET

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

The datatypes of all columns in the customers table are:

- customer_id — STRING
- Customer_unique_id — STRING
- Customer_zipcode_prefix — INTEGER
- Customer_city — STRING
- Customer_state — STRING

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

```
select *, x.maximum_order_time-x.minimum_order_time as
Range_of_order_time
from
(
 select
max(extract (time from order_purchase_timestamp))as
maximum_order_time,
min(extract (time from order_purchase_timestamp)) as minimum_order_time
from `target.orders`
)
as x
order by x.minimum_order_time
```

output:

Row	maximum_order_time	minimum_order_time	Range_of_order_time
1	23:59:59	00:00:00	0-0 0 23:59:59

Explanation:

In the above que its asked for the range between orders were placed, which indicates the range(highest-lowest) time of orders placed includes the time interval of orders being placed over the given data.So, we calculated the range by extracting the time from the “order_purchase_time_stamp).

1.3. Count the number of Cities and States in our dataset.

```
SELECT  count(geolocation_city) as number_of_cities,
        count(geolocation_state) as number_of_states
        FROM `target.geolocation`
```

Output:

Row	number_of_cities	number_of_states
1	1000163	1000163

Explanation:

In the above question,it is asked for the number_of cities and states ,so we used the count() on both cities and country columns and got the total number of cities and countries in the given dataset.

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

```
select  distinct extract(year from order_purchase_timestamp) as year,
 count(order_id) over(partition by extract(year from
order_purchase_timestamp)) as no_of_orders,
from `target.orders`
```

Output:

Row	year	no_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Explanation:

In the above question it is asked for the growing trend on the basis of the number of orders placed over the years. So we used the count() on the order_id in a distinct manner,we have partitioned over the year which is extracted from the column “order_purchase_times_stamp” from orders table. Hence, we can see the number of orders placed is increasing year by year in the given dataset, therefore there is a growth trend of orders placed, we can observe the same in the below mentioned picture as well.

- As we can see through the output of the resultant query:
- Yes, there is a growing trend in the number of orders placed over the past years.

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

- Monthly seasonality is defined as a regular pattern that recurs every month in data that is observed more frequently than monthly.
- In one word, Monthly seasonality means the fluctuations occur in a month compared to another month over sales,orders etc..

Information from From orders table for first 10 rows:

```
select * ,lead(x.no_of_orders,1)over( order by x.month   ) as
comparing_monthly_seasonality
from
(select  distinct extract(month from order_purchase_timestamp) as month,
count(order_id) over(partition by extract(month from
order_purchase_timestamp)) as no_of_orders,
from `target.orders`) as x
order by x.month
```

output:

Row	month	no_of_orders	comparing_monthly_seasonality
1	1	8069	8508
2	2	8508	9893
3	3	9893	9343
4	4	9343	10573
5	5	10573	9412
6	6	9412	10318
7	7	10318	10843
8	8	10843	4305
9	9	4305	4959
10	10	4959	7544

Explanation:

In The above question it is asked for the monthly seasonality on number of orders placed, we used lead() on the month column which is extracted from the “order_purchase_timestamp” from orders table and used count() on the order_id to know the number of orders were placed and ordered them by month so,we can differentiate the seasonality easily.

2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

```
with cte as(select order_id,
 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 "Evening"
        when  extract( hour from order_purchase_timestamp) between 19 and 23
        then "Night"
        end as day_part
        from `target.orders` ),

        cte2 as (select count(order_id) as no_of_orders,day_part from cte
        group by day_part)

        select day_part, max(no_of_orders) as max_order_time from cte2
        group by day_part
        order by max_order_time desc
```

Row	day_part	max_order_time
1	Evening	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Explanation:

In the above mentioned question , it askes for the number of orders placed at most in the time of dawn,night,morning,evening. So, we used case statement to set these fours “day_parts” on the extracted column of day from “order_purchase_timestamp”.Then we used a cte method to query and count() to number of orders and the max() to get the higher orders placed at which day_part.

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

```
select *
from
(select distinct extract(month from  o.order_purchase_timestamp ) as month,
c.customer_state,count(order_id)over(partition by c.customer_state) as
count_of_orders
from `target.customers` c join `target.orders` as o
on c.customer_id = o.customer_id
)as x
order by month
```

By the above query we can get the month on month number of orders placed in each state.

Row	month	customer_state	count_of_orders
1	1	AM	148
2	1	GO	2020
3	1	SP	41746
4	1	ES	2033
5	1	AC	81
6	1	DF	2140
7	1	PR	5045
8	1	RN	485
9	1	CE	1336
10	1	TO	280

Explanation:

In the above mentioned question,it is asked for the month on month orders placed for each state. So,firstly we joined the tables “orders” and “customers” on the column customer_id and extracted the month from the column “order_purchase_timestamp” and count() on the order_id and partitoned them by customer_state from customer_table to get the final output.

3.2 How are the customers distributed across all the states?

```
select customer_state,count(customer_id) as no_of_customers
from `target.customers`
group by customer_state
order by customer_state
```

By the above query we can get the customer distribution across the states.

Row	customer_state	no_of_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747

Explanation:

In the above question,its asked for the customer distribution across the states. So, we have to calculate the number of customers across each state from the customer table.We used the count() on the customer_id and grouped by customer_state, to get the resultant output.

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

You can use the "payment_value" column in the payments table to get the cost of orders.

```
with cte as (select* from
        (select o.order_id, extract(year from o.order_purchase_timestamp) as
years,p.payment_value
        from `target.orders` as o inner join `target.payments` as p
        on o.order_id = p.order_id
        where (o.order_purchase_timestamp between ("2017-01-01") and
("2017-08-31")) or
        (o.order_purchase_timestamp between ("2018-01-01") and ("2018-08-31"))
        )as x
        pivot (sum(cast(payment_value as int))for years in (2017,2018)
        )as pvt_table
        )
select (sum(_2018)-sum(_2017)*100/sum(_2017)) as inc_per from cte
```

output:

Row	increase_percentage
1	8694395.0

Explanation:

It asked for the percentage increase in the cost of the products from 2017 to 2018 in the months January to August with both years. So, firstly we have joined the tables orders and customers table to use the payments column as the cost column for the products, on the column order_id. We extracted the year from the “order_purchase_timestamp” for the years to claculate.In the where clause we filter the months from January to August and pivoted the values of cost with respected to the years and we total the cost of both years and calculated the percentage by the formula Old_price-New-price/Old_price*100.

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

```
select distinct gs.geolocation_state, round(sum(price),3) as total_price,
round(avg(price),3) as avg_price
from `target.order_items` ot inner join `target.sellers` ts
on ot.seller_id= ts.seller_id join `target.geolocation` gs
on gs.geolocation_zip_code_prefix = ts.seller_zip_code_prefix
group by gs.geolocation_state
order by gs.geolocation_state
```

Output:

Row	geolocation_state	total_price	avg_price
1	AC	43788.0	267.0
2	AM	31779.0	392.333
3	BA	23385841.45	351.609
4	CE	740073.63	246.117
5	DF	4674257.66	72.559
6	ES	3211486.26	127.309
7	GO	4444926.59	164.128
8	MA	3604486.05	89.899
9	MG	253958733.426	122.974
10	MS	724755.24	165.431

Explanation:

In the above mentioned question is asked for the total and average values of price for each state. So we have to firstly join the tables order_items and geolocations , as the two tables are not directly connected , we have to join the order_items to the sellers table and the sellers table to the geolocations table and calculate the avg() and sum() on the price of the orders and group them by the state from the geolocations table to get the required output.

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

```
select distinct gs.geolocation_state, round(sum(freight_value),3) as
total_freight_price,
round(avg(freight_value),3) as avg_freight_price
from `target.order_items` ot inner join `target.sellers` ts
on ot.seller_id= ts.seller_id join `target.geolocation` gs
on gs.geolocation_zip_code_prefix = ts.seller_zip_code_prefix
group by gs.geolocation_state
order by gs.geolocation_state
```

output:

Row	geolocation_state	total_freight_price	avg_freight_price
1	AC	5385.76	32.84
2	AM	2208.6	27.267
3	BA	1939324.41	29.158
4	CE	163715.97	54.445
5	DF	1223546.71	18.993
6	ES	724107.32	28.705
7	GO	694619.52	25.649
8	MA	1201987.71	29.978
9	MG	47130618.12	22.822
10	MS	113813.87	25.979

Explanation:

The above mentioned question is asked for the total and average freight values for each state. So we have to firstly join the tables order_items and geolocations , as the two tables are not directly connected , we have to join the order_items to the sellers table and the sellers table to the geolocations table and calculate the avg() and sum() on thefrieght_value and group them by the state from the geolocations table to get the required output.

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

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
- diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date

```
select distinct order_id,abs(extract(day from order_delivered_customer_date) - extract(day from order_purchase_timestamp)) as time_to_deliver,
 abs(extract(day from order_estimated_delivery_date) - extract(day from order_delivered_customer_date)) as diff_estimated_delivery
 from `target.orders`
```

Explanation:

In the above mentioned Question is asked for the time taken to deliver the order in the number of days and the estimated time to deliver in the days. We calculated the time taken to deliver by time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
and the Estimated delivery time
By diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date. From the orders table.

Row	order_id	time_to_deliver	diff_estimated_delivery
1	2c45c33d2f9cb8ff8b1c86cc28c11c30	0	1
2	68f47f50f04c4cb6774570cfde3a9aa7	0	2
3	304e7fc7db4a67a8ab0403ce4eb42ae0	28	11
4	c930f0fb9c6fed6ef015de48ea8dd5ea	29	12
5	8d204be4884a2307f1486df7260010f4	27	13
6	0d8f485ffe96c81fe3e282095e942c2e	29	12
7	8576190c64f6d9d9ed5055185aeefe31	27	13
8	81279a15416799e6580df60f66760a7b	0	12
9	604eadfa781354090d53180d97b11dca	0	2
10	9734c61e70431951ceea6da4c1c2a063	0	5

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

```
select *, dense_rank() over(order by x.avg_frt_value desc) as highest_val,
dense_rank() over(order by x.avg_frt_value ) as lowest_val
from(
select gs.geolocation_state,
 round(avg(ot.freight_value),2) as avg_frt_value
from `target.order_items` ot inner join `target.sellers` ts
on ot.seller_id= ts.seller_id join `target.geolocation` gs
on gs.geolocation_zip_code_prefix = ts.seller_zip_code_prefix
group by gs.geolocation_state
limit 5) as x
```

output:

Row	geolocation_state	avg_frt_value	highest_val	lowest_val
1	PI	36.94	2	4
2	RO	50.32	1	5
3	SE	29.13	4	2
4	AC	32.84	3	3
5	AM	27.27	5	1

Explanation:

The above mentioned case is to find the top 5 status with average highest and lowest freight value. We firstly joined the tables order_items and sellers and then geolocations as the prder_items and geolactions are not directly connected.Then we found the average of the freight value and used the dense_rank() to get the ranking from high to low of average values in descending order for highest avg value and dense_rank() in ascending to get the lowest avg values rank and group by state and limit 5 for top five values.

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

```
select *, dense_rank() over(order by x.avg_delivery_time desc ) as
highest_delivery_time,
dense_rank() over(order by x.avg_delivery_time) as lowest_delivery_time
from
(
select gs.geolocation_state,
round(avg(extract(day from o.order_delivered_carrier_date)),2) as avg_delivery_time
from `target.orders` as o join  `target.order_items` ot
on o.order_id = ot.order_id  join `target.sellers` ts
on ot.seller_id= ts.seller_id join `target.geolocation` gs
on gs.geolocation_zip_code_prefix = ts.seller_zip_code_prefix
group by gs.geolocation_state
) as x
Where avg_delivery_time is not null
limit 5
```

output:

Row	geolocation_state	avg_delivery_time	highest_delivery_time	lowest_delivery_time
1	DF	16.31	5	17
2	MT	14.84	18	4
3	SC	15.9	10	12
4	PI	18.08	2	20
5	BA	14.21	20	2

Explanation:

In the above case it is asked for the highest and lowest average delivery time for each state.So firstly we joined the tables orders and order_items and sellers and then geolocations as they were not directly connected, we calculated the averager() delivery_time and given it dense_rank() for lowest and highest average values grouped by states and the limit of five states, to get the required output.

5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

```
with cte as (select *
from
(
select gs.geolocation_state,
round(avg(extract(day from o.order_delivered_carrier_date)),2) as avg_delivery_time,
round(avg(extract(day from o.order_estimated_delivery_date)),2) as
avg_estimation_time
from `target.orders` as o join `target.order_items` ot
on o.order_id = ot.order_id join `target.sellers` ts
on ot.seller_id= ts.seller_id join `target.geolocation` gs
on gs.geolocation_zip_code_prefix = ts.seller_zip_code_prefix
group by gs.geolocation_state
) as x
)
select * from cte
where avg_delivery_time<avg_estimation_time
order by avg_delivery_time
limit 5
```

output:

Row	geolocation_state	avg_delivery_time	avg_estimation_time
1	AM	13.33	19.0
2	BA	14.21	16.42
3	ES	14.62	15.46
4	MT	14.84	15.16
5	MA	14.86	15.24

Explanation:

In the above case, it is asked for the average_delivery_time and the estimated_delivery_time to each state ,So firstly we joined the tables orders and geolocations, as they are not directly connected , we join them using the intermediate tables order _items and sellers , then we calculate the average of delivery_time and the delivery estimated time w.r.t each state to get the required output.

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

```
with cte as
(select distinct extract(month from o.order_purchase_timestamp) as months,
p.payment_type,
count(o.order_id) over(partition  by extract(month from
o.order_purchase_timestamp)) as no_of_orders
from `target.orders` as o join `target.payments` as p
on o.order_id = p.order_id
)
select * from cte
order by months
```

Row	months	payment_type	no_of_orders
1	1	UPI	8413
2	1	voucher	8413
3	1	credit_card	8413
4	1	debit_card	8413
5	2	credit_card	8838
6	2	voucher	8838
7	2	debit_card	8838
8	2	UPI	8838
9	3	voucher	10349
10	3	UPI	10349

Explanation:

In the above case we have to find the Payment_type and the number of orders w.r.t each month, So we join the orders and the payments table,then extracted the month from the “orderpurchase_timestamp” column and then calculated the number of orders on the each state over partition by month to get the required output.

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

```
select distinct payment_installments,count(order_id) as no_of_orders
from `target.payments`
where payment_installments >=1
group by payment_installments
```

Output:

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

Explanation:

The above is to find out the number of orders that are placed on the basis of EMI,
So we calculate the count() on the orders column as number_of_orders and select the payment_installments and the number_of_orders where the number of EMI>1 so that no direct payments reflect, to get the required output.