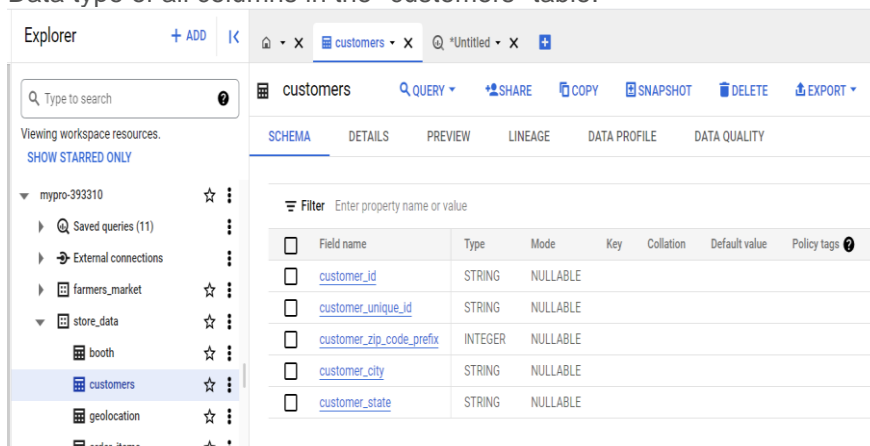


Business Case: Target SQL

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



Field name	Type	Mode	Key	Collation	Default value	Policy tags
customer_id	STRING	NULLABLE				
customer_unique_id	STRING	NULLABLE				
customer_zip_code_prefix	INTEGER	NULLABLE				
customer_city	STRING	NULLABLE				
customer_state	STRING	NULLABLE				

In the customer table data type of all the columns except zip-code is string which accepts null values as well, the column zip-code is of the time integer.

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

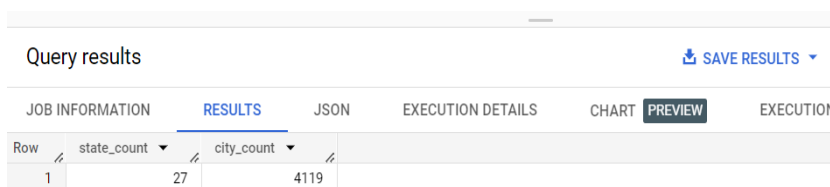
Query - `select min(order_purchase_timestamp) as first_date,max(order_purchase_timestamp) as last_date from `store_data.orders`;`

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

Inference: From the above result we can see that 1st order was placed in 2016 September and last order was placed in October 2018

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

Query - `select count(distinct c.customer_state) as state_count ,count(distinct c.customer_city) as city_count from `store_data.orders` o inner join `store_data.customers` c on o.customer_id = c.customer_id;`

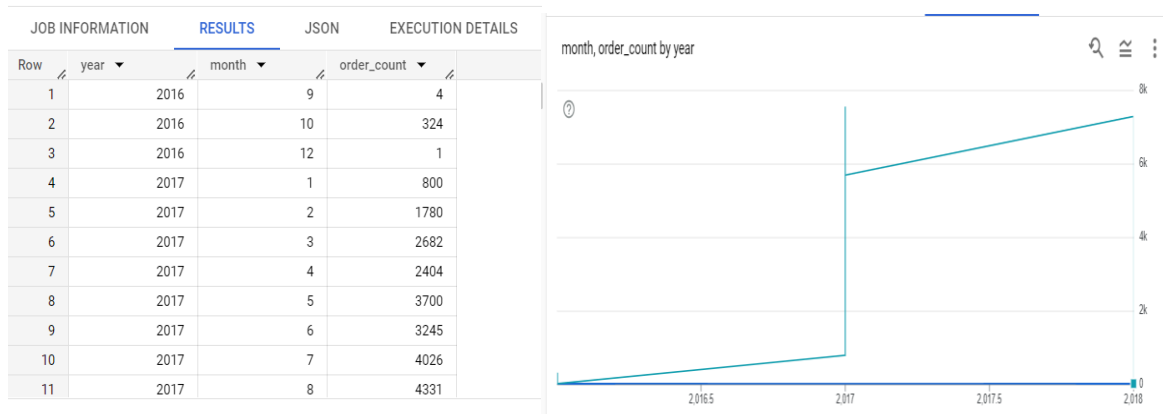


Row	state_count	city_count
1	27	4119

Inference: From the result we can say the from 27 states and 4119 cities across the states orders have been received during the given period

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

```
select extract(year from order_purchase_timestamp) as year,extract(month from
order_purchase_timestamp) as month,count(order_id) as order_count from `store_data.orders` group
by 1,2 order by 1,2;
```



In the result we can see that initially in the starting years the orders were less, in the coming 2 years 2017 & 2018 there is an increasing trend in the order count, in 2017 there is a 99% increase compared to 2016 where as there is only 54% increase the coming year. The order count increase with respect to month has a gradual growing trend but at the start of the 2017 there is a sudden spike in the order count which later on has turned to a marginal increase.

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

```
select year,month,order_count,max(order_count)over(partition by year order by month) as spike from
(select extract(year from order_purchase_timestamp) as year,extract(month from
order_purchase_timestamp) as month,count(order_id) as order_count from `store_data.orders` group
by 1,2)as t order by 1,2;
```

Row	year	month	order_count	spike
1	2016	9	4	4
2	2016	10	324	324
3	2016	12	1	324
4	2017	1	800	800
5	2017	2	1780	1780
6	2017	3	2682	2682
7	2017	4	2404	2682
8	2017	5	3700	3700
9	2017	6	3245	3700
10	2017	7	4026	4026
11	2017	8	4331	4331

We can say that initially there was a small spike in the orders in 2016 in oct, in November the greatest number of orders were placed in 2017, in January the peak number of orders was placed in 2018.

- 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

```
select time_of_day , count(time_of_day) as num_of_orders from
(
select c.customer_id,o.order_purchase_timestamp,
case when extract(hour from o.order_purchase_timestamp) between 0 and 6 then 'Dawn'
when extract(hour from o.order_purchase_timestamp) between 7 and 12 then 'Morning'
when extract(hour from o.order_purchase_timestamp) between 13 and 18 then 'Afternoon'
when extract(hour from o.order_purchase_timestamp) between 19 and 23 then 'Night'
end as time_of_day
from `store_data.customers` c inner join `store_data.orders` o on c.customer_id = o.customer_id
order by 1
) as t
group by 1
```

Inner query result

Query results					EXECUTION GRAPH
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	
Row	customer_id	order_purchase_timestamp	time_of_day		
1	00012a2ce6f8dcd20d059ce9...	2017-11-14 16:08:26 UTC	Afternoon		
2	000161a058600d5901f007fab...	2017-07-16 09:40:32 UTC	Morning		
3	0001fd6190edaa884bca3d49...	2017-02-28 11:06:43 UTC	Morning		
4	0002414f95344307404f0ace7...	2017-08-16 13:09:20 UTC	Afternoon		
5	000379cdec625522490c315e7...	2018-04-02 13:42:17 UTC	Afternoon		

Results per page:

Result

Query results			
JOB INFORMATION		RESULTS	JSON
Row	time_of_day	num_of_orders	
1	Night	28331	
2	Afternoon	38135	
3	Morning	27733	
4	Dawn	5242	

Inference: From the above result we can say that orders placed in the afternoon were the highest and at dawn least no of orders were placed where as the no of orders placed during night and morning is almost the same.

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

```
select extract(month from o.order_purchase_timestamp) as month,
c.customer_state,count(o.order_id) as order_count ,
from `store_data.orders` o join `store_data.customers` c on o.customer_id = c.customer_id
group by 1,2
order by 1,2;
```

Query results SAVE RESULTS

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	CHART PREVIEW	EXECUTION GRAPH
Row	month	customer_state	order_count		
12	1	MS	71		
13	1	MT	96		
14	1	PA	82		
15	1	PB	33		
16	1	PE	113		
17	1	PI	55		
18	1	PR	443		
19	1	RJ	990		

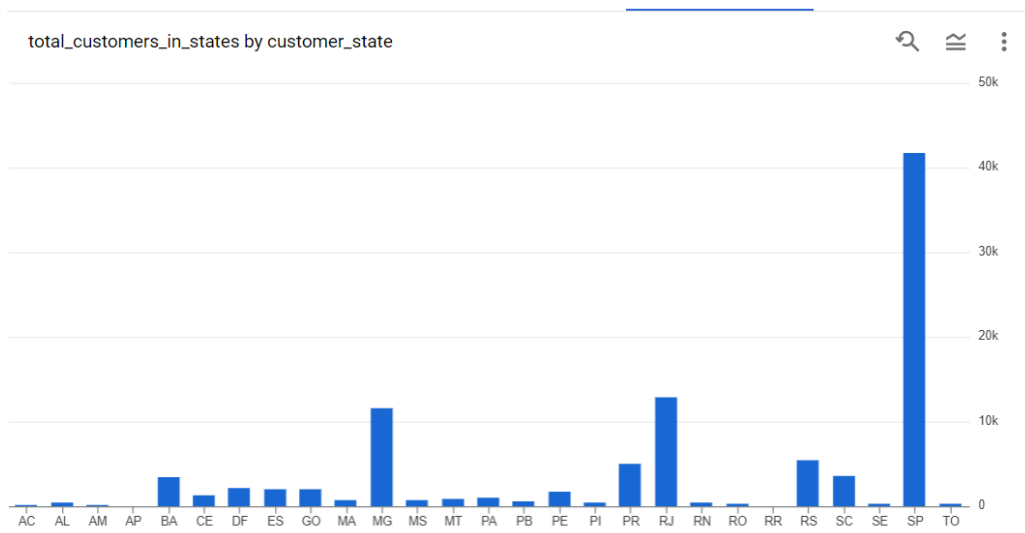


From the result we can say that in august we can see highest number of orders was from the state "SP", 4982 orders and the lowest number of orders was from RR and AP states with count of 2

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

```
select customer_state,count(customer_id) as total_customers_in_states from `store_data.customers`
group by 1 order by 1
```

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS	CHART PREVIEW
Row	customer_state	total_customers_in_states		
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		
11	MG	11635		
12	MS	715		
13	MT	907		
14	PA	975		
15	PB	536		



From the above data we can see the highest number of customers live in SP state and lowest number of customers are from RR state. Other than SP state, only MG and RJ has customers more than 10K where as the other states has customers less than 5K.

4.1. Get the % increase in the cost of orders from year 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.

```
select month,total_payments,round(((total_payments/sum(total_payments)over(order by month))*100)
as percent_increase from
(select distinct month,sum(payment_value)over(partition by month order by month) as total_payments
from
(select extract(year from o.order_purchase_timestamp) as year,extract(month from
o.order_purchase_timestamp) as month, p.payment_value from `store_data.orders`o join
`store_data.payments`p on o.order_id = p.order_id)s where month between 1 and 8 and year
between 2017 and 2018) as t
```

JOB INFORMATION		RESULTS	JSON
Row	month	total_payments	
1	1	1253492.22	
2	2	1284371.35	
3	3	1609515.72	
4	4	1578573.51	
5	5	1746900.97	
6	6	1535156.88	
7	7	1658923.67	
8	8	1696821.64	

The total payments for each month exhibit a gradual rise, reaching its peak in May, while the remaining fluctuations remain moderate.

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

```
select c.customer_state,round(sum(p.payment_value),2) as total_order_value ,
round(avg(p.payment_value),2) as average_order_value from `store_data.customers` c join
`store_data.orders` o on c.customer_id = o.customer_id
join `store_data.payments` p on o.order_id = p.order_id
group by 1
order by 1
```

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	CHART	PREVIEW
Row	customer_state	total_order_value	average_order_value			
1	AC	19680.62	234.29			
2	AL	96962.06	227.08			
3	AM	27966.93	181.6			
4	AP	16262.8	232.33			
5	BA	616645.82	170.82			
6	CE	279464.03	199.9			
7	DF	355141.08	161.13			
8	ES	325967.55	154.71			
9	GO	350092.31	165.76			

The average order value across all states hovers around 180, with the highest average originating from the state PB. The combined order value from all states amounts to under 2 million, except for state SP, which contributes roughly 6 million in total order value.

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

```
select c.customer_state,round(sum(oi.freight_value),2) as total_freight_value ,
round(avg(oi.freight_value),2) as average_freight_value from `store_data.customers` c join
`store_data.orders` o on c.customer_id = o.customer_id
join `store_data.order_items` oi on o.order_id = oi.order_id
group by 1
order by 1
```

Row	customer_state	total_freight_value	average_freight_value
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06

The state with the lowest average freight value is SP, coinciding with the highest total freight value. Conversely, the state with the highest average freight value is RR, while having the lowest total freight value.

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.

```
select order_id,
date_diff(date(order_delivered_customer_date),date(order_purchase_timestamp),day) as
time_to_deliver,
date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date),day) as
diff_estimated_delivery from `store_data.orders`
```

Row	order_id	time_to_deliver	diff_estimated_delivery
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	31	29
3	65d1e226dfaeb8cdc42f66542...	36	17
4	635c894d068ac37e6e03dc54e...	31	2
5	3b97562c3aee8bdedcb5c2e45...	33	1
6	68f47f50f04c4cb6774570cfde...	30	2
7	276e9ec344d3bf029ff83a161c...	44	-4
8	54110207b000d1548150...	41	4

The average delivery time for all orders is approximately 30 days. However, there are instances of negative values in the variance between estimated and actual delivery dates, indicating that certain deliveries did not align with the initial estimations.

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

```
(select customer_state,average_freight_value, rank()over(order by average_freight_value asc) as
rank from
(select c.customer_state, round(avg(oi.freight_value),2) as average_freight_value from
`store_data.customers` c join `store_data.orders` o on c.customer_id = o.customer_id join
`store_data.order_items` oi on o.order_id = oi.order_id group by 1 )as t
order by 3 limit 5)
union all
(select customer_state,average_freight_value, rank()over(order by average_freight_value desc) as
rank from
(select c.customer_state, round(avg(oi.freight_value),2) as average_freight_value from
`store_data.customers` c join `store_data.orders` o on c.customer_id = o.customer_id join
`store_data.order_items` oi on o.order_id = oi.order_id group by 1 )as t
order by 3 limit 5)
order by average_freight_value;
```

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

From the analysis of the top and bottom states based on average freight value provides valuable insights to understand state wise shipping cost trends, optimize logistics operations, and ensure better alignment between estimated and actual delivery times.

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

```
select customer_state,average_delivery_time from
((select customer_state,average_delivery_time, rank()over(order by average_delivery_time asc) as
rank from
(select c.customer_state,
round(avg(date_diff(date(o.order_delivered_customer_date),date(o.order_purchase_timestamp),day))
,2) as average_delivery_time from `store_data.customers` c join `store_data.orders` o on
c.customer_id = o.customer_id group by 1)as t
order by 3 limit 5)
union all
(select customer_state,average_delivery_time, rank()over(order by average_delivery_time desc) as
rank from
(select c.customer_state,
round(avg(date_diff(date(o.order_delivered_customer_date),date(o.order_purchase_timestamp),day))
,2) as average_delivery_time from `store_data.customers` c join `store_data.orders` o on
c.customer_id = o.customer_id group by 1)as t
order by 3 limit 5)
order by average_delivery_time)as w
```

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	customer_state	average_delivery_time		
1	SP	8.7		
2	PR	11.94		
3	MG	11.95		
4	DF	12.9		
5	SC	14.91		
6	PA	23.73		
7	AL	24.5		
8	AM	26.36		
9	AP	27.18		
10	RR	29.34		

The above insights can be used to manage customer expectations and to improve operational efficiency.

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.

```
select c.customer_state,
round(avg(date_diff(date(order_estimated_delivery_date),date(order_delivered_customer_date),day))
,2) as compared_delivery_time from `store_data.customers` c join `store_data.orders` o on
c.customer_id = o.customer_id group by 1 order by 2 limit 5
```

Row	customer_state	compared_delivery_time
1	AL	8.71
2	MA	9.57
3	SE	10.02
4	ES	10.5
5	BA	10.79

By studying about order delivery is really fast as compared to the estimated date of delivery, we can see where the plans and strategies are working better and those can be implemented in states where delivery time is taking longer than usual, operations aren't performing that well .

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

```
select distinct year,month,payment_type,count(order_id)over(partition by payment_type order by
month) as order_count from
(select o.order_id,extract(year from o.order_purchase_timestamp) as year,extract(month from
o.order_purchase_timestamp) as month, p.payment_type from `store_data.orders` o join
`store_data.payments` p on o.order_id = p.order_id)s
order by 1,2
```

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	CHART	PREVIEW
Row	year	month	payment_type	order_count		
1	2016	9	credit_card	62742		
2	2016	10	UPI	17115		
3	2016	10	credit_card	66520		
4	2016	10	debit_card	1395		
5	2016	10	voucher	5094		
6	2016	12	credit_card	76795		
7	2017	1	UPI	1715		
8	2017	1	credit_card	6103		
9	2017	1	debit_card	118		

From the above result we can say that the customers who ordered using the payment type credit card is highest when compared to the other payment methods used. From this we can say that offers on credit card would attract more customers .

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

```
select payment_type,payment_installments,count(order_id) as order_count from
(select distinct o.order_id,extract(year from o.order_purchase_timestamp) as year,extract(month from
o.order_purchase_timestamp) as month,p.payment_installments,p.payment_sequential,
p.payment_type from `store_data.orders` o join `store_data.payments` p on o.order_id = p.order_id
where p.payment_installments >= 1)s
group by 1,2
order by 1,2
```

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	payment_type ▼	payment_installment	order_count ▼	
1	UPI	1	19784	
2	credit_card	1	25455	
3	credit_card	2	12413	
4	credit_card	3	10461	
5	credit_card	4	7098	
6	credit_card	5	5239	
7	credit_card	6	3920	
8	credit_card	7	1626	
9	credit_card	8	4268	
10	credit_card	9	644	
11	credit_card	10	5328	

From the above we can infer that this data can be utilized to understand the customer behavior, customer segmentation, operational efficiency.