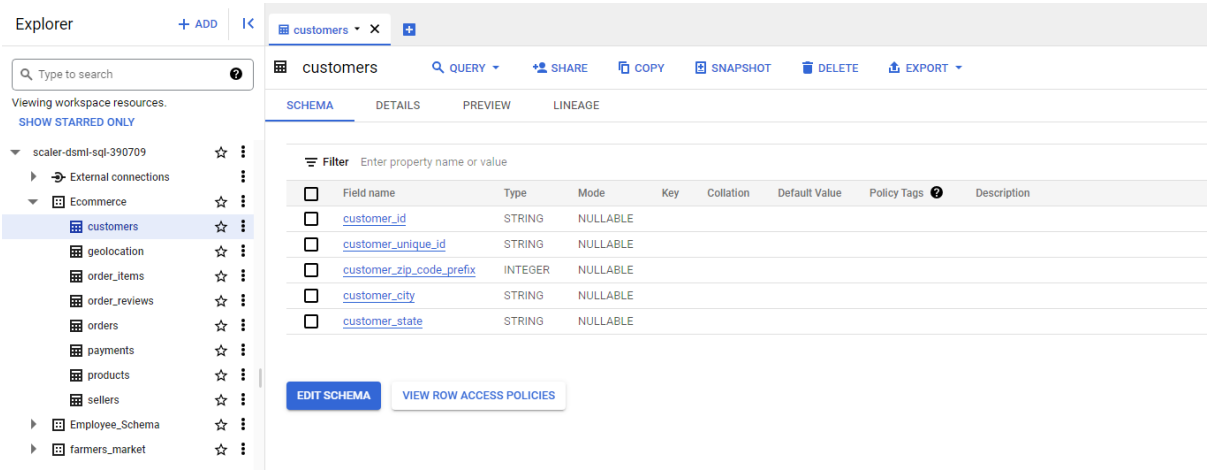
Business Case: Target SQL

Problem Statement:

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

i). Data type of all columns in the "customers" table.

customer_id – STRING
customer_unique_id - STRING
customer_zip_code_prefix - INTEGER
customer_city - STRING
customer_state - STRING



Findings:

From the above Snippet we can find the datatype of all the columns in ''customers'' table.

ii). Get the time range between which the orders were placed.

Query:

```
select min(order_purchase_timestamp) as first_order,max(order_purchase_timestamp) as last_order
from Ecommerce.orders
```

Query results

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

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

Findings:

From the above Snippet we can find the first and the last order placed, thus displaying the range in which the orders were placed.

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

Query:

```
select count(distinct customer_state) as States,
count(distinct customer_city) as Cities
from Ecommerce.orders o join
Ecommerce.customers c on o.customer_id=c.customer_id
```

Query results

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

Row	States	Cities
1	27	4119

Findings:

From the above Snippet we can find the number of different cities and states from which order was placed.

2) In-depth Exploration:

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

Query:

```
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as Total_orders
from Ecommerce.orders
group by year,month
order by year,month
```

Query results

SAVE RESU

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

Row	year	month	Total_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631

Findings:

From the above Snippet we can see that there is a growing trend in the number of orders placed over the years.

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

Query:

```
select distinct t.month, avg(t.Total_orders) over(partition by t.month order by t.month) as Average from(
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(order_id) as Total_orders
from Ecommerce.orders
group by year,month
order by year,month ) as t
```

Query results

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

Row	month	Average
1	1	4034.5
2	2	4254.0
3	3	4946.5
4	4	4671.5
5	5	5286.5
6	6	4706.0
7	7	5159.0
8	8	5421.5
9	9	1435.0
10	10	1653.0
11	11	7544.0
12	12	2837.0

Findings:

From the above Snippet we can see that there is monthly seasonality in terms of number of orders being placed and November (month 11) is having the more orders.

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

Query:

```
select count(*) as total_purchase,

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 'Mornings'
  when extract(hour from order_purchase_timestamp) between 3 and 18
    then 'Afternoon'
   else 'Night'
end as time_purchase
from Ecommerce.orders
group by time_purchase
order by total_purchase desc
```

Query results

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

Row	total_purchase	time_purchase
1	38135	Afternoon
2	28331	Night
3	27733	Mornings
4	5242	Dawn

Findings:

From the above Snippet we can see that the customers mostly placed their orders in Afternoon followed by Night, Mornings, and Dawn.

3) Evolution of E-commerce orders in the Brazil region:

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

Query:

```
select extract(month from order_purchase_timestamp) as month_of_purchase,customer_state,count(*)
as orders_placed

from Ecommerce.orders o
join Ecommerce.customers c on o.customer_id=c.customer_id
group by month_of_purchase,customer_state
order by customer_state,month_of_purchase
```

Query results

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

Row	month_of_purchase	customer_state	orders_placed
1	1	AC	8
2	2	AC	6
3	3	AC	4
4	4	AC	9
5	5	AC	10
6	6	AC	7
7	7	AC	9
8	8	AC	7
9	9	AC	5
10	10	AC	6
11	11	AC	5
12	12	AC	5
13	1	AL	39
14	2	AL	39
15	3	AL	40

Findings:

From the above Snippet we get the no. of orders placed in each state, in each month by our customers.

ii). How are the customers distributed across all the states?

Query:

```
select customer_state,count(distinct customer_id) as no_of_customers from

Ecommerce.customers
group by customer_state
order by no_of_customers
```

Query results

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

Row	customer_state	no_of_customers
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536
12	MS	715
13	MA	747
14	MT	907
15	PA	975
16	CE	1336

Findings:

From the above Snippet we get the number of unique customers in each state.

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

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

Query:

```
 select year,
ifnull(Round(((t3.sumation-t3.prev_years_value)/t3.prev_years_value)*100,2),0) as percentage_increase from
(select *,
lag(t2.sumation) over(order by year) as prev_years_value
from
(select t1.year,sum(t1.payment_value)as sumation
from
(select
extract(month from order_purchase_timestamp) as month,
extract(year from order_purchase_timestamp) as year,
*
from Ecommerce.payments p
join Ecommerce.orders o on p.order_id=o.order_id) as t1
where t1.year between 2017 and 2018  and t1.month<=8
group by t1.year
order by t1.year) as t2) as t3
```

Query results

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

Row	year	percentage_increase
1	2018	136.98
2	2017	0.0

Findings:

From the above Snippet we get 136.98% increase in the cost of orders from year 2017 to 2018 by considering months between Jan to Aug only.

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

Query:

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

Query results

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

Row	customer_state	total_order_price	average_order_price
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
10	MA	152523.02	198.86
11	MG	1872257.26	154.71
12	MS	137534.84	186.87
13	MT	187029.29	195.23
14	PA	218295.85	215.92

Findings:

From the above Snippet we get the total and average value of orders placed in each state.

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

Query:

```
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 Ecommerce.order_items oi
join Ecommerce.orders o on oi.order_id=o.order_id
join Ecommerce.customers c on c.customer_id=o.customer_id
group by c.customer_state
order by c.customer_state
```

Query results

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

Row	customer_state	total_freight_value	average_freight_valu
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
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83
15	PB	25719.73	42.72

Findings:

From the above Snippet we get the total and average value of order freight for each state.

5) Analysis based on sales, freight and delivery time.

i). 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.

Query:

```
select *,
case
when t.Extended_days>0 then 'Late Delivery Than Expected'
when t.Extended_days<0 then 'Early Delivery Than Expected'
when t.Extended_days=0 then 'Exact Day Delivered As Expected'
end as status
```

```
from
(select
order_id,customer_id,order_purchase_timestamp,order_estimated_delivery_date,order_delivered_customer_date,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as Days_Taken,
date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as Extended_days,
from Ecommerce.orders) t
```

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	order_id	customer_id	order_purchase_timestamp	order_estimated_delivery_date	order_delivered_customer_date	Days_Taken	Extended_days	status
1	770d331c84e5b214bd9dc70a...	6c57e6119369185e575b3671...	2016-10-07 14:52:30 UTC	2016-11-29 00:00:00 UTC	2016-10-14 15:07:11 UTC	7	-45	Early Delivery Than Expected
2	1950d777989f6a877539f5379...	1bccb206de9f0f25adc6871a1...	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	2018-03-21 22:03:51 UTC	30	12	Late Delivery Than Expected
3	2c45c33d2f9cb8ff8b1c86cc28...	de4caa97afa80c8eeac2ff4c8d...	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	2016-11-09 14:53:50 UTC	30	-28	Early Delivery Than Expected
4	dabf2b0e35b423f94618bf965f...	5cdec0bb8cbdf53ffc8fdc212c...	2016-10-09 00:56:52 UTC	2016-11-30 00:00:00 UTC	2016-10-16 14:36:59 UTC	7	-44	Early Delivery Than Expected
5	8beb59392e21af5eb9547ae1a...	bf609b5741f71697f65ce3852c...	2016-10-08 20:17:50 UTC	2016-11-30 00:00:00 UTC	2016-10-19 18:47:43 UTC	10	-41	Early Delivery Than Expected
6	b60b53ad0bb7dacacf2989fe2...	2f9902d85fcd930227f711cf47...	2017-05-10 14:03:27 UTC	2017-05-18 00:00:00 UTC	2017-05-23 13:12:27 UTC	12	5	Late Delivery Than Expected
7	276e9ec344d3bf029ff83a161c...	d33e520a99eb4cfc0d3ef2b6ff...	2017-04-08 21:20:24 UTC	2017-05-18 00:00:00 UTC	2017-05-22 14:11:31 UTC	43	4	Late Delivery Than Expected
8	1a0b31f08d0d7e87935b819ed...	7e769bb9acb55403ebd7ea57a...	2017-04-11 13:50:49 UTC	2017-05-18 00:00:00 UTC	2017-04-18 08:18:11 UTC	6	-29	Early Delivery Than Expected
9	cec8f5f7a13e5ab934a486ec9e...	6be61d704faaff9b97ccf47289...	2017-03-17 15:56:47 UTC	2017-05-18 00:00:00 UTC	2017-04-07 13:14:56 UTC	20	-40	Early Delivery Than Expected
10	2d846c03073b1a424c1be1a77...	5a3473648da10d2800f4783f6...	2017-05-10 15:18:16 UTC	2017-05-18 00:00:00 UTC	2017-05-25 10:49:48 UTC	14	7	Late Delivery Than Expected
11	54e1a3c2b97fb0809da548a59...	a0bc11375dd3d8bdd0e0bfcbc...	2017-04-11 19:49:45 UTC	2017-05-18 00:00:00 UTC	2017-05-22 16:18:42 UTC	40	4	Late Delivery Than Expected
12	58527ee4726911bee84a0f42c...	b7d68eb92ede54186f0385024...	2017-03-20 11:01:17 UTC	2017-05-18 00:00:00 UTC	2017-03-30 14:04:04 UTC	10	-48	Early Delivery Than Expected
13	302bb8109d097a9fc6e9cefc5...	22c0028cdec95ad1808c1fd50...	2017-04-19 22:52:59 UTC	2017-05-18 00:00:00 UTC	2017-05-23 14:19:48 UTC	33	5	Late Delivery Than Expected
14	10ed5499d1623638ee810eff1...	2bf569d940353f09136cab77b...	2017-03-21 13:38:25 UTC	2017-05-18 00:00:00 UTC	2017-04-18 13:52:43 UTC	28	-29	Early Delivery Than Expected

Findings:

From the above Snippet we get the number of days taken to deliver each order from the order's purchase date as delivery time. Also, the difference (in days) between the estimated & actual delivery date of an order.

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

Query:

```
(select c.customer_state,
round(avg(oi.freight_value),2) as average_freight_value
from Ecommerce.order_items oi
join Ecommerce.orders o on oi.order_id=o.order_id
join Ecommerce.customers c on c.customer_id=o.customer_id
group by c.customer_state
order by average_freight_value
limit 5)
union all
(select c.customer_state,
round(avg(oi.freight_value),2) as average_freight_value
from Ecommerce.order_items oi
join Ecommerce.orders o on oi.order_id=o.order_id
join Ecommerce.customers c on c.customer_id=o.customer_id
group by c.customer_state
order by average_freight_value desc
limit 5)
order by average_freight_value
```

Query results

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

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

Findings:

From the above Snippet we get the top 5 states with the lowest & highest average freight value arranged in increasing order of the average freight value.

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

Query:

```
(select c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2)as
average_days
from Ecommerce.orders o
join Ecommerce.customers c on c.customer_id=o.customer_id
group by c.customer_state
order by average_days
limit 5)
union all
(select c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2)as
average_days
from Ecommerce.orders o
join Ecommerce.customers c on c.customer_id=o.customer_id
group by c.customer_state
order by average_days desc
```

Query results

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

Row	customer_state	average_days
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Findings:

From the above Snippet we get the top 5 states with the lowest & highest average delivery time arranged in increasing order of the average delivery time.

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

Query:

```
select customer_state,
round((t.actual_average-t.expected_average),2)as Difference
from
(select c.customer_state,
round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2) as
actual_average,
round(avg(DATE_DIFF(o.order_estimated_delivery_date,o.order_purchase_timestamp,day)),2) as
expected_average
from Ecommerce.orders o
join Ecommerce.customers c on c.customer_id=o.customer_id
where o.order_delivered_customer_date is not null and o.order_status!='canceled'
group by c.customer_state) as t
order by Difference desc
limit 5
```

Query results

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

Row	customer_state	Difference
1	AL	-8.17
2	MA	-8.96
3	SE	-9.45
4	ES	-9.89
5	CE	-10.18

Findings:

From the above Snippet we get the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

6) Analysis based on the payments:

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

Query:

```
select extract(month from order_purchase_timestamp) as
month_of_purchase,customer_state,payment_type,count(*) as orders_placed
from Ecommerce.orders o
join Ecommerce.customers c on o.customer_id=c.customer_id
join Ecommerce.payments p on o.order_id=p.order_id
group by month_of_purchase,customer_state,payment_type
order by customer_state,month_of_purchase,payment_type
```

Query results

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

Row	month_of_purchase	customer_state	payment_type	orders_placed
1	1	AC	UPI	3
2	1	AC	credit_card	4
3	1	AC	voucher	2
4	2	AC	UPI	1
5	2	AC	credit_card	5
6	3	AC	UPI	1
7	3	AC	credit_card	3
8	4	AC	UPI	1
9	4	AC	credit_card	8
10	5	AC	UPI	3
11	5	AC	credit_card	7
12	5	AC	voucher	1
13	6	AC	UPI	1
14	6	AC	credit_card	6

Findings:

From the above Snippet we get the month on month number of orders placed using different payment types.

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

Query:

```
select payment_installments,count(*) as orders_placed
from Ecommerce.orders o
join Ecommerce.payments p on o.order_id=p.order_id
where payment_installments>0
group by payment_installments
order by payment_installments
```

Query results

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

Row	payment_installment	orders_placed
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
11	11	23
12	12	133

Findings:

From the above Snippet we get the number of orders placed on the basis of the payment installments that have been paid.

7) Actionable Insights & Recommendations:

- There is an increase in number of orders placed right from the first order placed.
- There is monthly seasonality in terms of number of orders being placed (i.e., 11th month has the highest number of orders placed followed by 8th,5th,3rd, and 9th month being the month with least number of orders placed.
- Time of the day has an impact on the orders being placed with Afternoon received highest number of orders and Dawn received lowest number of orders.
- Month on month number of orders placed has ups and downs across months over the same state.
- There is 136.98% increase in number of orders placed in 2018 in comparison with 2017.
- There is a delay in the orders being delivered in comparison with the estimated delivery timestamp.
- The average difference between the delivered date and the estimated delivery date in very less across the states.