Target SQL Business Case

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

Output:

SCHEMA | DETAILS | PREVIEW | LINEAGE | DATA PROFILE

Filter Enter property name or value

Field name	Type	Mode	Key	Co
customer_id	STRING	NULLABLE	-	-
customer_unique_id	STRING	NULLABLE	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-
customer_city	STRING	NULLABLE	-	-
customer_state	STRING	NULLABLE	-	-

Inference:
Except Zip code all other columns are of type string

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

Query:

```
select min(order_purchase_timestamp) as orders_start_date,
max((order_purchase_timestamp)) as orders_end_date,
count(order_id)/(timestamp_diff(max(order_purchase_timestamp),
min(order_purchase_timestamp), hour)) as OrdersPerHour
from `SQL_Target.orders`
```

Output:

ow	orders_start_date	orders_end_date	OrdersPerHour
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	5.361278844080...

Inference:
The dataset comprises of orders placed between September 4, 2016, at 21:15:19 UTC, and October 17, 2018, at 17:30:18 UTC. On average, approximately 5 orders have been placed every hour during this period.

#1.3.1 Count the Cities of customers who ordered during the given period.

Query:

```
select c.customer_city, count(distinct o.customer_id) as No_of_customers,
round(100 * count(distinct o.customer_id) / sum(count(distinct
o.customer_id)) over(), 2) as customer_distribution
from `SQL_Target.orders` o
```

```
join `SQL_Target.customers` c
on o.customer_id = c.customer_id
group by c.customer_city
order by No_of_customers desc
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	customer_city	No_of_customers	customer_distributio
1	sao paulo	15540	15.63
2	rio de janeiro	6882	6.92
3	belo horizonte	2773	2.79
4	brasilia	2131	2.14
5	curitiba	1521	1.53
6	campinas	1444	1.45
7	porto alegre	1379	1.39

Inference:
During the given period, orders were received from customers in a total of 4,119 cities. The majority of customers are from Sao Paulo (15.6% of total orders) and Rio de Janeiro (6.9% of total orders). Interestingly, nearly 3,155 cities have single-digit customers, with fewer than 10 orders each.

#1.3.2 Count the states of customers who ordered during the given period.

Query:

```
select customer_state,  T.No_of_customers, round(100 * T.No_of_customers
/ T.total_customers, 2) as customer_distribution
from (select c.customer_state, count(distinct o.customer_id) as
No_of_customers, sum(count(distinct o.customer_id)) over()  as
total_customers
from `SQL_Target.orders` o
join `SQL_Target.customers` c
on o.customer_id = c.customer_id
group by c.customer_state) T
order by T.No_of_customers desc
```

Output:

Row	customer_state	No_of_customers	customer_distributio
1	SP	41746	41.98
2	RJ	12852	12.92
3	MG	11635	11.7
4	RS	5466	5.5
5	PR	5045	5.07
6	SC	3637	3.66
7	BA	3380	3.4
8	DF	2140	2.15
9	ES	2033	2.04
10	GO	2020	2.03

Inference:

During the given period, orders were placed by customers from a total of 27 states. Most customers are from SP accounting for 42% of the orders, followed by RJ with 13%, and MG with 11.7%. Conversely, the least number of customers are from AC, AP, and RR with less than 100 customers each.

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

Query:

```
select t.year, t.month, count(*) as monthly_no_of_orders,
(100 * (count(*) - lag(count(*)) over(order by year, month)))/lag(count(*))
over(order by year, month) as MoMPercentChange
from (select *, Extract(month from order_purchase_timestamp) as month,
Extract(year from order_purchase_timestamp) as year
from `SQL_Target.orders`) t
group by t.year, t.month
order by t.year, t.month
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUT

Row	year	month	monthly_no_of_order	MoMChange
5	2017	2	1780	122.5
6	2017	3	2682	50.67415730337...
7	2017	4	2404	-10.3653989560...
8	2017	5	3700	53.91014975041...
9	2017	6	3245	-12.2972972972...
10	2017	7	4026	24.06779661016...
11	2017	8	4331	7.575757575757...
12	2017	9	4285	-1.06211036712...
13	2017	10	4631	8.074679113185...
14	2017	11	7544	62.90218095443...
15	2017	12	5673	-24.8011664899...
16	2018	1	7269	28.13326282390...
17	2018	2	6728	-7.44256431421...
18	2018	3	7211	7.178953626634...
19	2018	4	6939	-3.77201497711...

Inference:

While there are no distinct patterns observed during 2016 and end of 2018 sales, there is a noticeable upward trend in monthly orders throughout the year 2017. This trend continues into 2018, reaching its peak before stabilizing.

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

Query:

```
with cte1 as
(select *, Extract(month from order_purchase_timestamp) as month,
Extract(year from order_purchase_timestamp) as year
from `SQL_Target.orders`),
cte2 as
(select year, month, count(*) as monthly_orders, (100 * (count(*) -
lag(count(*)) over(order by year, month)))/lag(count(*)) over(order by year,
month) as MoMChange
from cte1
group by year, month
order by year, month
)
```

```
select l.month, l.monthly_orders as orders2017, l.MoMChange as
MoMChange2017, r.monthly_orders as orders2018, r.MoMChange as
MoMChange2018
from (select *
from cte2
where year = 2017) l
join (select *
from cte2
where year = 2018) r
on l.month = r.month
order by l.month
```

Output:

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	month	orders2017	MoMChange2017	orders2018	MoMChange2018
1	1	800	79900.0	7269	28.13326282390...
2	2	1780	122.5	6728	-7.44256431421...
3	3	2682	50.67415730337...	7211	7.178953626634...
4	4	2404	-10.3653989560...	6939	-3.77201497711...
5	5	3700	53.91014975041...	6873	-0.95114569822...
6	6	3245	-12.2972972972...	6167	-10.2720791502...
7	7	4026	24.06779661016...	6292	2.026917463920...
8	8	4331	7.575757575757...	6512	3.496503496503...
9	9	4285	-1.06211036712...	16	-99.7542997542...
10	10	4631	8.074679113185...	4	-75.0

Inference:

In comparison between 2017 and 2018, notable patterns emerge: a surge in sales is observed during January, February, July, and August, showcasing significant growth compared to preceding months. Conversely, a decline in sales is evident in April, June, and September when compared to their respective preceding months.

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

Query:

```
select t.order_time, count(*) as no_of_orders
from (select time(order_purchase_timestamp),
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 'Afternoon'
```

```
when Extract(hour from order_purchase_timestamp) between 19 and 23
then 'Night'
end as order_time from `SQL_Target.orders`) t
group by t.order_time
order by no_of_orders desc
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW

ow	order_time	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Inference:

The majority of orders are typically placed in the afternoon, with the lowest number of orders occurring during dawn

#2.3 Detailed analysis

Query:

```
select t.order_hour, count(*) as no_of_orders , sum(count(*)) over(order by
t.order_hour rows between 5 preceding and current row) as six_hour_orders
from (select time(order_purchase_timestamp),
Extract(hour from order_purchase_timestamp) as order_hour
from `SQL_Target.orders`) t
group by t.order_hour
order by t.order_hour
```

Output:

11	10	6177	15850
12	11	6578	22240
13	12	5995	27733
14	13	6518	33020
15	14	6569	36622
16	15	6454	38291
17	16	6675	38789
18	17	6150	38361
19	18	5769	38135

Inference:

More specifically, during the six-hour window from 11:00:00 to 16:59:59 (UTC-4), the maximum number of orders is observed. Conversely, the period from 01:00:00 to 06:59:59 (UTC-4) sees the minimum number of orders.

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

Query:

```
with cte as (select distinct c.customer_state, t.year, t.month, count(*)
over(partition by c.customer_state, t.year, t.month) as monthly_orders
from (select *, Extract(month from order_purchase_timestamp) as month,
Extract(year from order_purchase_timestamp) as year
from `SQL_Target.orders`) t
join `SQL_Target.customers` c
on t.customer_id = c.customer_id
order by c.customer_state, t.year, t.month)

select *, 100 * (monthly_orders - (lag(monthly_orders) over(order by
customer_state, year, month)))/lag(monthly_orders) over(order by
customer_state, year, month) as percent_change
from cte
order by customer_state, year, month
```

Output:

Row	customer_state	year	month	monthly_orders	percent_change
1	AC	2017	1	2	*null*
2	AC	2017	2	3	50.0
3	AC	2017	3	2	-33.33333333...
4	AC	2017	4	5	150.0
5	AC	2017	5	8	60.0
6	AC	2017	6	4	-50.0
7	AC	2017	7	5	25.0
8	AC	2017	8	4	-20.0
9	AC	2017	9	5	25.0
10	AC	2017	10	6	20.0
11	AC	2017	11	5	-16.66666666...

Inference:

Just a data extraction, no much insights

#3.1 Detailed Analysis

Query:

```
with cte as (select distinct c.customer_state, t.year, t.month, count(*)
over(partition by c.customer_state, t.year, t.month) as monthly_orders
from (select *, Extract(month from order_purchase_timestamp) as month,
Extract(year from order_purchase_timestamp) as year
from `SQL_Target.orders`) t
join `SQL_Target.customers` c
on t.customer_id = c.customer_id
order by c.customer_state, t.year, t.month),
cte2 as (
```

```
select *, 100 * (monthly_orders - (lag(monthly_orders) over(order by
customer_state, year, month)))/lag(monthly_orders) over(order by
customer_state, year, month) as percent_change,
nth_value(month,1) over(partition by customer_state order by
monthly_orders desc) as max_mon,
nth_value(year,1) over(partition by customer_state order by monthly_orders
desc) as max_year,
from cte
order by customer_state, year, month)

select distinct customer_state, max_year, max_mon
from cte2
order by customer_state
```

Output:

Row	customer_state	max_year	max_mon
1	AL	2018	1
2	AP	2018	1
3	MA	2018	1
4	MS	2018	1
5	MT	2018	1
6	PI	2018	1
7	RN	2018	1
8	RO	2018	1
9	SC	2018	1
10	DF	2018	2
11	PA	2018	3

Inference:

Among the 27 states, 9 states demonstrate their highest sales figures in January 2018. Additionally, 6 states each record their peak sales in November 2017 and July 2018

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

Query:

```
select customer_state, count(*) as num_customer,
count(*)*100/(sum(count(*)) over()) as percentage_distribution
from `SQL_Target.customers`
group by customer_state
order by percentage_distribution desc
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | E

Row	customer_state	num_customer	percentage_distribut
1	SP	41746	41.98067195623...
2	RJ	12852	12.92424653814...
3	MG	11635	11.70040526543...
4	RS	5466	5.496726702265...
5	PR	5045	5.073360082863...
6	SC	3637	3.657445118210...
7	BA	3380	3.399000412304...
8	DF	2140	2.152029846843...
9	ES	2033	2.044428354501...
10	GO	2020	2.031355275992...

Inference:

Customers are not evenly distributed across all states, with approximately 70% of customers originating from São Paulo (SP), Rio de Janeiro (RJ), and Minas Gerais (MG) alone

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

Query:

```
with cte as
(select l.year, l.month, sum(payment_value) as Totalorderprice
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_value
from `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id) l
where l.year>=2017 and l.month between 1 and 8
group by l.year, l.month)

select l.month, l.Totalorderprice as OrderPrice2017, r.Totalorderprice as
OrderPrice2018,
100*(r.Totalorderprice - l.Totalorderprice) / l.Totalorderprice as YoYGrowth
from (select * from cte where year = 2017) l
join (select * from cte where year = 2018) r
on l.month = r.month
order by l.month
```

Output:

Row	month	OrderPrice2017	OrderPrice2018	YoYGrowth
1	1	138488.0399999...	1115004.180000...	705.1266954171...
2	2	291908.0099999...	992463.3400000...	239.9918145445...
3	3	449863.6000000...	1159652.119999...	157.7786066709...
4	4	417788.0300000...	1160785.479999...	177.8407701149...
5	5	592918.8200000...	1153982.149999...	94.62734375677...
6	6	511276.3800000...	1023880.499999...	100.2596912456...
7	7	592382.9200000...	1066540.750000...	80.04245463390...
8	8	674396.3200000...	1022425.320000...	51.60600520477...

Inference:
The year-on-year growth of monthly sales from 2017 to 2018 is more than 100% for each month, except for August

#4.1 Detailed Analysis 1
Query:

```
with cte as
(select l.year, sum(payment_value)/count(order_id) as price_per_order
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_value
from `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id) l
where l.year>=2017 and l.month between 1 and 8
group by l.year)

select year, price_per_order, 100*((select price_per_order from cte where year
= 2018) - (select price_per_order from cte where year = 2017)) / (select
price_per_order from cte where year = 2017) as PercentChange2017To2018
from cte
order by year
```

Output:

Row	year	price_per_order	PercentChange2017To2018
1	2017	150.4252437374...	3.2253393398267556
2	2018	155.2769683007...	3.2253393398267556

Inference:
A 3.25% increase is observed in the price per order from January to August 2017 to the same period in 2018

#4.1 Detailed Analysis 2

Query:

```
with cte as
(select l.year, count(order_id) as NoOfOrders, sum(payment_value) as
CostOfOrders, sum(payment_value)/count(order_id) as price_per_order
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_value
from `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id) l
where l.year>=2017 and l.month between 1 and 8
group by l.year)

select year, NoOfOrders,
case when year= 2018 then 100*((select NoOfOrders from cte where year =
2018) - (select NoOfOrders from cte where year = 2017)) / (select
NoOfOrders from cte where year = 2017) end as
OrderPercentChng2017To2018,
CostOfOrders as TotalOrderPrice,
case when year= 2018 then 100*((select CostOfOrders from cte where year =
2018) - (select CostOfOrders from cte where year = 2017)) / (select
CostOfOrders from cte where year = 2017) end as
CostPercentChange2017To2018
from cte
order by year
```

Output:

Row	year	NoOfOrders	OrderPercentChng2017To2018	TotalOrderPrice	CostPercentChange2017To2018
1	2017	24391	*null*	3669022.119999...	*null*
2	2018	55995	129.57238325611905	8694733.839999...	136.97687164665652

Inference:

A notable increase of 137% is observed in the total order price for the defined period, accompanied by a significant 130% increase in the number of orders placed

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

Query:

```
with cte as
(select c.customer_state, avg(p.payment_value) as AvgSaleValue,
sum(p.payment_value) as TotalSaleValue
from `SQL_Target.orders` o
join `SQL_Target.customers` c
on o.customer_id = c.customer_id
join `SQL_Target.payments` p
on o.order_id = p.order_id
group by c.customer_state)
```

```
select *, 100*TotalSaleValue/(sum(TotalSaleValue) over()) as
TotalSaleValue_Distribution
from cte
order by TotalSaleValue desc
```

Output:

Row	customer_state	AvgSaleValue	TotalSaleValue	TotalSaleValue_Distr
1	SP	137.5046297739...	5998226.959999...	37.46814213417...
2	RJ	158.5258882235...	2144379.689999...	13.39494546477...
3	MG	154.7064336473...	1872257.260000...	11.69512284167...
4	RS	157.1804057868...	890898.5399999...	5.565030024113...
5	PR	154.1536259977...	811156.3799999...	5.066917731116...
6	SC	165.9793367075...	623086.4299999...	3.892131971131...
7	BA	170.8160166204...	616645.8200000...	3.851900467301...
8	DF	161.1347912885...	355141.0800000...	2.218401629658...
9	GO	165.7634043560...	350092.3100000...	2.186864304841...
10	ES	154.7069530137...	325967.55	2.036168117007...
11	PE	187.9921527777...	324850.4400000...	2.029190048898...
12	CE	199.9027396280...	279464.0299999...	1.745682193631...

Inference:

Sales from SP, RJ, and MG states collectively contribute to the highest revenue, accounting for over 60% of the total revenue generated during the period

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

Query:

```
with cte as
(select c.customer_state, avg(oi.freight_value) as AvgFreightValue,
sum(oi.freight_value) as TotalFreightValue
from `SQL_Target.orders` o
join `SQL_Target.customers` c
on o.customer_id = c.customer_id
join `SQL_Target.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state)


select *, 100*TotalFreightValue/(sum(TotalFreightValue) over()) as
TotalFreightValue_Distribution
from cte
order by TotalFreightValue desc
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION

Row	customer_s	AvgFreightValue	TotalFreightValue	TotalFreightValue_Distribution
1	SP	15.14727539041...	718723.0699999...	31.91616080635222
2	RJ	20.96092393168...	305589.3100000...	13.570230267775397
3	MG	20.63016680630...	270853.4600000...	12.027723813453218
4	RS	21.73580433039...	135522.7400000...	6.0181253994777579
5	PR	20.53165156794...	117851.6800000...	5.2334109299968565
6	BA	26.36395893656...	100156.6799999...	4.44763336275041
7	SC	21.47036877394...	89660.26000000...	3.9815213891762649
8	PE	32.91786267995...	59449.65999999...	2.6399666125132257

Inference:

The top3 states with highest freight value are SP, RJ and MG. All 3 are collectively accounting for over 50% of the total freight value compared to other states.

#5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

Query:

```
select order_id, timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, day) as NoOfDaysToDeliver,
timestamp_diff(order_delivered_customer_date,
order_estimated_delivery_date, day) as EstVsActual
from `SQL_Target.orders`
order by NoOfDaysToDeliver desc
```

Output:

Row	order_id	NoOfDaysToDeliver	EstVsActual
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	0f4519c5f1c541ddec9f21b3bd...	194	161
5	285ab9426d6982034523a855f...	194	166
6	2fb597c2f772eca01b1f5c561b...	194	155
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	437222e3fd1b07396f1d9ba8c...	187	144

Inference:

Orders within the dataset exhibit a diverse range of delivery periods, spanning from same-day delivery to delivery windows of up to 6 months.

#5.1. Detailed Analysis

Query:

```
create or replace view `SQL_Target.OrderDelivery` as (select order_id,
customer_id, timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, day) as NoOfDaysToDeliver,
timestamp_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) as EstVsActual
from `SQL_Target.orders`);

select avg(NoOfDaysToDeliver) as AvgNoOfDaysToDeliver,
(select count(*) from `SQL_Target.OrderDelivery` where (EstVsActual<0)) as
LateDeliveries,
(select count(*) from `SQL_Target.OrderDelivery` where (EstVsActual>0)) as
EarlyDeliveries,
(select count(*) from `SQL_Target.OrderDelivery` where (EstVsActual=0)) as
onTimeDeliveries
from `SQL_Target.OrderDelivery`
```

Output:

Row	AvgNoOfDaysToDeliver	LateDeliveries	EarlyDeliveries	onTimeDeliveries
1	12.094085575687346	6535	87187	2754

Inference:

On average, orders were delivered in approximately 12 days. Less than 6% of deliveries are late deliveries, while nearly 87% of deliveries are early deliveries

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

Query:

```
create view `SQL_Target.FreightPerState` as (select c.customer_state,
avg(freight_val_per_order) as avg_freight_val
from (select order_id, sum(freight_value) as freight_val_per_order
    from `SQL_Target.order_items`
    group by order_id) oi
join `SQL_Target.orders` o
on oi.order_id = o.order_id
join `SQL_Target.customers` c
on o.customer_id = c.customer_id
group by c.customer_state);

select customer_state, avg_freight_val
from (select *, dense_rank()over(order by avg_freight_val desc) as rnk
from `SQL_Target.FreightPerState`) l
where l.rnk <=5
order by l.rnk
```

Output:

Row	customer_state	avg_freight_val
1	RR	48.59108695652...
2	PB	48.34535714285...
3	RO	46.22421052631...
4	AC	45.51543209876...
5	PI	43.03894523326...

Inference:
The top 5 states with the highest freight values, in decreasing order, are RR, PB, RO, AC and PI

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

Query:

```
select customer_state, avg_freight_val
from (select *, dense_rank()over(order by avg_freight_val asc) as rnk
from `SQL_Target.FreightPerState`) l
where l.rnk <=5
order by l.rnk
```

Output:

Row	customer_state	avg_freight_val
1	SP	17.37095033232...
2	MG	23.46270443520...
3	PR	23.57976790716...
4	DF	23.82376470588...
5	RJ	23.94525231154...

Inference:
The top 5 states with the lowest freight values, in increasing order, are SP, MG, PR, DF and RJ

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

Query:

```
with cte2 as
(select c.customer_state, avg(NoOfDaysToDeliver) as AvgDeliveryTime
from `SQL_Target.OrderDelivery` od
join `SQL_Target.customers` c
on od.customer_id = c.customer_id
group by c.customer_state
),
cte3 as (
(select customer_state, AvgDeliveryTime
```

```
from (select *, dense_rank() over(order by AvgDeliveryTime desc) as rnk
from cte2) l
where l.rnk<=5
order by l.rnk)
union all
(select customer_state, AvgDeliveryTime
from (select *, dense_rank() over(order by AvgDeliveryTime) as rnk
from cte2) l
where l.rnk<=5
order by l.rnk))

select *
from cte3
order by AvgDeliveryTime desc
```

Output:

Row	customer_state	AvgDeliveryTime
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...
6	SC	14.47956019171...
7	DF	12.50913461538...
8	MG	11.54381329810...
9	PR	11.52671135486...
10	SP	8.298061489072...

Inference:

The top 5 rows in the result indicate the states with the highest delivery time, in decreasing order: RR, AP, AM, AL and PA

Conversely, the bottom 5 rows in the result indicate the states with the highest delivery time, in decreasing order: SC, DF, MG, PR and SP

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

Query:

```
select c.customer_state, avg(EstVsActual) as AvgDeliveryTime
from `SQL_Target.OrderDelivery` od
join `SQL_Target.customers` c
on od.customer_id = c.customer_id
group by c.customer_state
order by AvgDeliveryTime desc
limit 5
```

Output:

Row	customer_state	AvgDeliveryTime
1	AC	19.7625
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

Inference:

The top 5 states with the fastest delivery times are AC, RO, AP, AM, RR. On average, in the state of AC, deliveries are occurring 19 days in advance of the estimated delivery date.

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

Query:

```
select l.year, l.month, payment_type, count(*) as monthly_orders
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 `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id) l
group by l.year, l.month, payment_type
order by l.year, l.month, payment_type
```

Output:

Row	year	month	payment_type	monthly_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61
11	2017	2	UPI	398

Inference:

Credit card is the preferred payment method for the majority of transactions across most months

#6.1. Detailed Analysis

Query:

```
select count(o.order_id) as totalorders, p.payment_type,
sum(p.payment_value) as totalvalue
from `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id
group by p.payment_type
order by totalorders desc
```

Output:

Row	totalorders	payment_type	totalvalue
1	76795	credit_card	12542084.18999...
2	19784	UPI	2869361.269999...
3	5775	voucher	379436.8700000...
4	1529	debit_card	217989.7900000...
5	3	not_defined	0.0

Inference:

The preferred payment method for most transactions is credit card, accounting for approximately 75% of payments, while the least preferred method is debit card

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

Query:

```
select p.payment_installments, count(o.order_id) as totalorders,100*
count(o.order_id) / (select count(order_id) from `SQL_Target.payments`) as
paymentinstallment_percent ,
sum(100* count(o.order_id) / (select count(order_id) from
`SQL_Target.payments`)) over(order by p.payment_installments) as
cum_installment
from `SQL_Target.orders` o
join `SQL_Target.payments` p
on o.order_id = p.order_id
group by p.payment_installments
order by totalorders desc
```

Output:

Row	payment_installment	totalorders	paymentinstallment_	cum_installment
1	1	52546	50.58044394817...	50.58236913539...
2	2	12413	11.94867450859...	62.53104364399...
3	3	10461	10.06969177752...	72.60073542151...
4	4	7098	6.832489459599...	79.43322488111...
5	10	5328	5.128698765954...	99.67175557822...

Inference:
The majority of transactions, approximately 51%, are paid in a single installment. The maximum installment period observed is 24 months. Remarkably, 99.67% of transactions are completed in less than 10 installments.

#7. Recommendations

#7.1 Based on Freight Value

The freight value for SP, RJ and MG are the highest, indicating that many products ordered from these states may not have nearby sellers. Consequently, identifying frequent product sellers from these states or nearby could help reduce freight costs.

#7.2 Based on customer strength in states

Customers are not evenly distributed across states; approximately 70% of customers are from SP, RJ, and MG. Moreover, it's likely that a significant portion of these customers resides in tier I cities. Focusing promotional efforts on tier II cities could potentially boost revenue. Additionally, implementing targeted advertising and offering discounts based on geographical location may further enhance sales.

#7.3 Based on payment

Given the predominant use of credit cards in purchases, implementing discounts or cashback offers tied to credit card could significantly enhance sales.

----------XXXX----------