1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:**
    1. **Data type of all columns in the "customers" table.**

Query:

```
select column_name, data_type
from `scalar-dsml-sql-
438715.Target_SQL_Business_Case.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```

Output:

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

Insights:

a. Though the column name customer_unique_id sounds unique However, it has duplicate records as below:

Query:

```
select customer_unique_id, count(customer_unique_id) as
customer_unique_id_rep_count
from `Target_SQL_Business_Case.customers`
group by 1
having count(customer_unique_id) > 1
order by 2 desc
```

Output:

Row	customer_unique_id	customer_unique_id
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff346268	7
4	6469f99c1f9dfae7733b25662e7f1782	7
5	1b6c7548a2a1f9037c1fd3ddfed95f33	7
6	47c1a3033b8b77b3ab6e109eb4d5fdf3	6
7	f0e310a6839dce9de1638e0fe5ab282a	6
8	12f5d6e1cbf93dafd9dcc19095df0b3d	6
9	dc813062e0fc23409cd255f7f53c7074	6
10	de34b16117594161a6a89c50b289d3…	6

Recommendation:

Since customer_id is already a primary key, and a single table can have only one primary key therefore the nature of customer_unique_id seems redundant.

2. **Get the time range between which the orders were placed**

Query:

```
select min(order_purchase_timestamp) as first_order_timestamp,
       max(order_purchase_timestamp) as last_order_timestamp
from `scalar-dsml-sql-438715.Target_SQL_Business_Case.orders`
```

Output:

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

Insights:

a. The table contains 99441 unique customer records within this time range

b. Maximum no. of customers are from 'sao paulo' city

Query:

```
select customer_city, count(customer_id) as max_customer_count
from `scalar-dsml-sql-438715.Target_SQL_Business_Case.customers`
group by 1
order by 2 desc
limit 1
```

Output:

Row	customer_city	max_customer_count
1	sao paulo	15540

c. Minimum no. of customers are from multiple cities.

Query:

```
select customer_city, count(customer_id) as min_customer_count
from `scalar-dsml-sql-438715.Target_SQL_Business_Case.customers`
group by 1
order by 2, 1
```

Output:

Row	customer_city	min_customer_count
1	abadiania	1
2	abdon batista	1
3	acajutiba	1
4	acari	1
5	acucena	1
6	adhemar de barros	1
7	adrianopolis	1
8	adustina	1
9	agisse	1
10	agrestina	1

Recommendation:

Offers should be provided on the first order at locations with the lowest customer count. To get benefited from offers more people will try to register and it will eventually increase the customer count.

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 `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
```

Output:

Row	state_count	city_count
1	27	4119

Insight:

a. The Target is operating on an average in 152 cities per state
b. The is a good amount of job opportunities in these states

Recommendation:

Target should hire more employees in the given state/cities

2. In-depth Exploration:

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

Query:

```
with get_month_year_from_timestamp as
(select order_id, order_purchase_timestamp, extract(year from
order_purchase_timestamp) as year,
        extract(month from order_purchase_timestamp) as month
from `Target_SQL_Business_Case.orders`),

order_count_per_year as
(select year, count(order_id) as order_count
from get_month_year_from_timestamp
group by year
order by 1),

order_comparision_year_wise as
(select year, order_count, lag(order_count, 1) over(order by year) as
prev_yr_order_count
from order_count_per_year
order by year)

select year, order_count, ifnull(concat(round(((order_count-
prev_yr_order_count)/prev_yr_order_count)*100,2),"%"),'NA') as
percentage_increase
from order_comparision_year_wise
```

Output:

Row	year	order_count	percentage_increase
1	2016	329	NA
2	2017	45101	13608.51%
3	2018	54011	19.76%

Insights:

a. Yes, there has been a massive growth in the year 2017 followed by consistent growth in 2018

Recommendation:

Target should focus on customer retention

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

Answer:

a. **Lowest order in 2016 December month**

Query:

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

```
select year, month, order_count
from(select year, month, order_count, dense_rank() over (order by
order_count) as order_rank
     from get_order_count_year_month_wise)
where order_rank = 1
```

Output:

year	month	order_count
2016	12	1

b. Highest order in 2017 November month

Query:

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

select year, month, order_count
from(select year, month, order_count, dense_rank() over (order by
order_count desc) as order_rank
     from get_order_count_year_month_wise)
where order_rank = 1
```

Output:

year	month	order_count
2017	11	7544

c. **Rising trend in 2017 where it started from 800**

Query:

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

Output:

Row	year	month	order_count
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

d. **Steady growth in 2018**

Query:

```
select year, month, order_count
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 `Target_SQL_Business_Case.orders`
     group by 1, 2)
where year = 2018
order by 1, 2
```

Output:

Row	year	month	order_count
1	2018	1	7269
2	2018	2	6728
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	6	6167
7	2018	7	6292
8	2018	8	6512
9	2018	9	16
10	2018	10	4

e. **Compared to December 2016 there has been significantly increase in the order count in December 2017**

Query:

```
select year, month, order_count
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 `Target_SQL_Business_Case.orders`
     group by 1, 2)
where month = 12
order by 1, 2
```

Output:

Row	year	month	order_count
1	2016	12	1
2	2017	12	5673

Insights:

a. Lowest order in 2016 December month
b. Highest order in 2017 November month
c. Rising trend in 2017 where it started from 800
d. Steady growth in 2018
e. Compared to December 2016 there has been significantly increase in the order count in December 2017

Recommendation:

Target should focus on customer retention and acquire new customers

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
Answer: 13-18 hrs : Afternoon

Query: 0-6 hrs : Dawn

```
with get_order_count_by_hours as
(select extract(hour FROM order_purchase_timestamp) as hours,
count(order_id) as order_count
 from `Target_SQL_Business_Case.orders`
 group by 1
)

select sum(order_count) as total_orders_between_0hr_to_6hr
from get_order_count_by_hours
where hours between 0 and 6
```

Output:

Row	total_orders_between_0hr_to_6hr
1	5242

7-12 hrs : Mornings
Query:

```
with get_order_count_by_hours as
```

```
(select extract(hour FROM order_purchase_timestamp) as hours,
count(order_id) as order_count
 from `Target_SQL_Business_Case.orders`
 group by 1
)

select sum(order_count) as total_orders_between_7hr_to_12hr
from get_order_count_by_hours
where hours between 7 and 12
```

Output:

Row	total_orders_between_7hr_to_12hr
1	27733

13-18 hrs : Afternoon

Query:

```
with get_order_count_by_hours as
(select extract(hour FROM order_purchase_timestamp) as hours,
count(order_id) as order_count
 from `Target_SQL_Business_Case.orders`
 group by 1
)

select sum(order_count) as total_orders_between_13hr_to_18hr
from get_order_count_by_hours
where hours between 13 and 18
```

Output:

Row	total_orders_between_13hr_to_18hr
1	38135

19-23 hrs : Night

Query:

```
with get_order_count_by_hours as
(select extract(hour FROM order_purchase_timestamp) as hours,
count(order_id) as order_count
 from `Target_SQL_Business_Case.orders`
 group by 1
)
select sum(order_count) as total_orders_between_19hr_to_23hr
from get_order_count_by_hours
where hours between 19 and 23
```

Output:

Row	total_orders_between_19hr_to_23hr
1	28331

Insights:

a. Highest orders were placed during afternoon time
b. Lowest orders were placed during dawn time

Recommendation: Happy hours should be introduce to attract more customers during the dawn time

3. **Evolution of E-commerce orders in the Brazil region:**
   1. **Get the month on month no. of orders placed in each state**

Query:

```
select c.customer_state, extract(year from o.order_purchase_timestamp) as
year,
        extract(month from o.order_purchase_timestamp) as month,
count(o.order_id) as order_count
from `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
group by 1, 2, 3
order by 1, 2, 3
```

Output:

Row	customer_state	year	month	order_count
1	AC	2017	1	2
2	AC	2017	2	3
3	AC	2017	3	2
4	AC	2017	4	5
5	AC	2017	5	8
6	AC	2017	6	4
7	AC	2017	7	5
8	AC	2017	8	4
9	AC	2017	9	5
10	AC	2017	10	6

Insights:

a. It seems like order count gets increased as the year end approaches due to holiday seasons

Recommendation:

Hire more staffs on contract during the holiday season

   2. **How are the customers distributed across all the states?**

Query:

```
select c.customer_state, count(c.customer_id) as customer_count
from `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
group by 1
order by 2 desc
```

Output:

Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

Insights:

a. Most customers are from SP state
b. Least customer are from RR state

Recommendation:

SP state customers should be given some reward points whereas, RR state customers should be given cash back so that they are retained.

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

```
with get_cost_of_order as
(select extract(year from o.order_purchase_timestamp) as year,
       sum(p.payment_value) as cost_of_orders,
from `Target_SQL_Business_Case.orders` as o
inner join `Target_SQL_Business_Case.payments` as p
on o.order_id = p.order_id
where extract(year from o.order_purchase_timestamp) between 2017 and 2018
      and extract(month from order_purchase_timestamp) between 1 and 8
group by 1
order by 1)

select year, cost_of_orders,
       round((cost_of_orders-lag(cost_of_orders, 1) over(order by
year))/lag(cost_of_orders, 1) over(order by year)*100,2) as per_increase
from get_cost_of_order
order by 1
```

    Output:

Row	year	cost_of_orders	per_increase
1	2017	3669022.119999…	*null*
2	2018	8694733.839999…	136.98

    Insights:
        a. There has been significant growth in the order percentage

    Recommendation:
        Strengthen the customer support to handle inquires/issues

    2. **Calculate the Total & Average value of order price for each state.**
    Query:

```
select c.customer_state, round(sum(oi.price),2) as total_order_price,
       round(sum(oi.price)/count(distinct(oi.order_id)),2) as
average_order_price
from `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
inner join `Target_SQL_Business_Case.order_items` as oi
on o.order_id = oi.order_id
group by 1
order by 2 desc, 3 desc
```

Output:

Row	customer_state	total_order_price	average_order_price
1	SP	5202955.05	125.75
2	RJ	1824092.67	142.93
3	MG	1585308.03	137.33
4	RS	750304.02	138.13
5	PR	683083.76	136.67
6	SC	520553.34	144.12
7	BA	511349.99	152.28
8	DF	302603.94	142.4
9	GO	294591.95	146.78
10	ES	275037.31	135.82

Insights:

SP state customers have contributed a good amount in increasing the company revenue

Recommendation:

Invest in that state for market expansion

3. 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(sum(oi.freight_value)/count(distinct(o.order_id)),2) as
average_freight_value
from `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
inner join `Target_SQL_Business_Case.order_items` as oi
on o.order_id = oi.order_id
group by 1
order by 2 desc, 3 desc
```

Output:

Row	customer_state	total_freight_value	average_freight_value
1	SP	718723.07	17.37
2	RJ	305589.31	23.95
3	MG	270853.46	23.46
4	RS	135522.74	24.95
5	PR	117851.68	23.58
6	BA	100156.68	29.83
7	SC	89660.26	24.82
8	PE	59449.66	36.07
9	GO	53114.98	26.46
10	DF	50625.5	23.82

Insights:

SP state shows the shipping and logistics costs is very high

Recommendation:

a. Regional inventories should be close to customer market
b. Outsource to 3rd party logistics
c. Re-evaluate shipping strategies

5. **Analysis based on sales, freight and delivery time.**
   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.
      Query:

```
with timediff_timestamp_format as
(select customer_id, order_id,
       (order_delivered_customer_date-order_purchase_timestamp) as
time_to_deliver_timestamp_format,
       (order_delivered_customer_date-order_estimated_delivery_date) as
diff_estimated_delivery_timestamp_format
from `Target_SQL_Business_Case.orders`
order by 3 desc)

select customer_id, order_id,
       floor((del_hr+del_min+del_sec)/86400) as time_to_deliver_in_days,
       floor((est_hr+est_min+est_sec)/86400) as diff_in_delivery_in_days
from(select customer_id, order_id,
     extract(hour from time_to_deliver_timestamp_format)*3600 as del_hr,
     extract(minute from time_to_deliver_timestamp_format)*60 as del_min,
     extract(second from time_to_deliver_timestamp_format) as del_sec,

     extract(hour from diff_estimated_delivery_timestamp_format)*3600 as
est_hr,
     extract(minute from diff_estimated_delivery_timestamp_format)*60 as
est_min,
     extract(second from diff_estimated_delivery_timestamp_format) as
est_sec
     from timediff_timestamp_format)
order by 3 desc, 4 desc
```

Output:

Row	customer_id	order_id	time_to_deliver_in_days	diff_in_delivery_in_days
1	75683a92331068e2d281b11a...	ca07593549f1816d26a572e06...	209.0	181.0
2	d306426abe5fca15e54b645e4...	1b3190b2dfa9d789e1f14c05b...	208.0	188.0
3	7815125148cfa1e8c7fee1ff79...	440d0d17af552815d15a9e41a...	195.0	165.0
4	9cf2c3fa2632cee748e1a59ca9...	285ab9426d6982034523a855f...	194.0	166.0
5	1a8a4a30dc296976717f44e78...	0f4519c5f1c541ddec9f21b3bd...	194.0	161.0
6	217906bc11a32c1e470eb7e08...	2fb597c2f772eca01b1f5c561b...	194.0	155.0
7	cb2caaaead400c97350c37a3f...	47b40429ed8cce3aee9199792...	191.0	175.0
8	65b14237885b3972ebec28c0f...	2fe324febf907e3ea3f2aa9650...	189.0	167.0
9	8199345f57c6d1cbe9701f924...	2d7561026d542c8dbd8f0daea...	188.0	159.0
10	f85e9ec0719b16dc4dd0edd43...	c27815f7e3dd0b926b5855262...	187.0	162.0

Insights:

Deliveries are consistently delayed by a significant number of days

Recommendation:

a. Analyse the root cause for the delivery delay

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

Query:

```
with get_avg_freight_per_state as
(select distinct(c.customer_state), avg(oi.freight_value) over (partition
by c.customer_state) as avg_freight_per_state
from `Target_SQL_Business_Case.customers` as c
inner join `Target_SQL_Business_Case.orders` as o
on c.customer_id = o.customer_id
inner join `Target_SQL_Business_Case.order_items` as oi
on o.order_id = oi.order_id),

get_highest_avg_freight as
(select 'highest Top 5 states avg freight value' as value_type,
        rank, round(avg_freight_per_state, 2) as avg_freight_per_state,
customer_state
from(select customer_state, avg_freight_per_state, dense_rank() over
(order by avg_freight_per_state desc) as rank
     from get_avg_freight_per_state
    )
where rank <= 5
),

get_lowest_avg_freight as
(select 'lowest Top 5 states avg freight value' as value_type,
        rank, round(avg_freight_per_state, 2) as avg_freight_per_state,
customer_state
from(select customer_state, avg_freight_per_state, dense_rank() over
(order by avg_freight_per_state) as rank
     from get_avg_freight_per_state
    )
where rank <= 5
),

combined_high_low_freight as
(select value_type, rank, customer_state, avg_freight_per_state
from get_highest_avg_freight
union distinct
select value_type, rank, customer_state, avg_freight_per_state
from get_lowest_avg_freight
order by 1, 2
)

select case when rank = 1 then value_type else '' end as value_type,
rank, customer_state, avg_freight_per_state
from combined_high_low_freight
```

Output:

Row	value_type	rank	customer_state	avg_freight_per_state
1	highest Top 5 states avg freight value	1	RR	42.98
2		2	PB	42.72
3		3	RO	41.07
4		4	AC	40.07
5		5	PI	39.15
6	lowest Top 5 states avg freight value	1	SP	15.15
7		2	PR	20.53
8		3	MG	20.63
9		4	RJ	20.96
10		5	DF	21.04

Insights:

a. Many state has highest average freight value. This could be due to geographical challenges which could be preventing the smooth delivery process.

b. Whereas, on the other some states has the lowest freight value. This could be due to geographical factors which could be facilitating the smooth delivery process.

Recommendation:

Use low cost region for the inventory.

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

Query:

```
with timediff_timestamp_format as
(select customer_id,
       (order_delivered_customer_date-order_purchase_timestamp) as
time_to_deliver_timestamp_format
from `Target_SQL_Business_Case.orders`),

delivery_time_days as
(select customer_id,
       floor((del_hr+del_min+del_sec)/86400) as time_to_deliver_in_days,
from(select customer_id,
     extract(hour from time_to_deliver_timestamp_format)*3600 as del_hr,
     extract(minute from time_to_deliver_timestamp_format)*60 as del_min,
     extract(second from time_to_deliver_timestamp_format) as del_sec
     from timediff_timestamp_format)),

get_avg_delivery_time_per_state as
(select distinct(c.customer_state), avg(d.time_to_deliver_in_days) over
(partition by c.customer_state) as avg_delivery_time_per_state
from `Target_SQL_Business_Case.customers` as c
inner join delivery_time_days as d
on c.customer_id = d.customer_id),

get_highest_avg_delivery_time as
(select 'highest Top 5 states avg delivery time' as value_type,
```

```
        rank, floor(avg_delivery_time_per_state) as
avg_delivery_time_per_state, customer_state
from(select customer_state, avg_delivery_time_per_state, dense_rank()
over (order by avg_delivery_time_per_state desc) as rank
     from get_avg_delivery_time_per_state
    )
where rank <= 5
),

get_lowest_avg_delivery_time as
(select 'lowest Top 5 states avg delivery time' as value_type,
        rank, floor(avg_delivery_time_per_state) as
avg_delivery_time_per_state, customer_state
from(select customer_state, avg_delivery_time_per_state, dense_rank()
over (order by avg_delivery_time_per_state) as rank
     from get_avg_delivery_time_per_state
    )
where rank <= 5
),

combined_highest_lowest_delivery_time as
(select value_type, rank, customer_state, avg_delivery_time_per_state
from get_highest_avg_delivery_time
union distinct
select value_type, rank, customer_state, avg_delivery_time_per_state
from get_lowest_avg_delivery_time
order by 1, 2
)

select case when rank = 1 then value_type else '' end as value_type,
       rank, customer_state, avg_delivery_time_per_state as
avg_delivery_time_in_days_per_state
from combined_highest_lowest_delivery_time
```

Output:

Row	value_type	rank	customer_state	avg_delivery_time_in_days_per_state
1	highest Top 5 states avg delivery time	1	RR	28.0
2		2	AP	26.0
3		3	AM	25.0
4		4	AL	24.0
5		5	PA	23.0
6	lowest Top 5 states avg delivery time	1	SP	8.0
7		2	PR	11.0
8		3	MG	11.0
9		4	DF	12.0
10		5	SC	14.0

Insights:

a. Many states are showing high delivery time. This could be due to geographical challenges which could be preventing the smooth delivery process.

b. Whereas, on the ones has the lowest delivery time. This could be due to geographical factors which could be facilitating the smooth delivery process.

Recommendation:

Use low cost region for the inventory.

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

Query:

```
with timediff_timestamp_format as
(select customer_id,
       (order_delivered_customer_date-order_purchase_timestamp) as
actual_delivery_timestamp_format,
       (order_estimated_delivery_date-order_purchase_timestamp) as
estimated_delivery_timestamp_format
from `Target_SQL_Business_Case.orders`),

join_cust_order as
(select c.customer_state,
td.actual_delivery_timestamp_format,td.estimated_delivery_timestamp_forma
t
from `Target_SQL_Business_Case.customers` as c
inner join timediff_timestamp_format as td
on c.customer_id = td.customer_id),

act_and_est_del_days as
(select customer_state,
       floor((act_hr+act_min+act_sec)/86400) as actual_delivery_in_days,
       floor((est_hr+est_min+est_sec)/86400) as
estimated_delivery_in_days
from(select customer_state,
     extract(hour from actual_delivery_timestamp_format)*3600 as act_hr,
     extract(minute from actual_delivery_timestamp_format)*60 as act_min,
     extract(second from actual_delivery_timestamp_format) as act_sec,

     extract(hour from estimated_delivery_timestamp_format)*3600 as
est_hr,
     extract(minute from estimated_delivery_timestamp_format)*60 as
est_min,
     extract(second from estimated_delivery_timestamp_format) as est_sec
     from join_cust_order)),

get_avg_of_act_and_est_delivery as
(select customer_state,
        floor(avg(actual_delivery_in_days) over (partition by
customer_state)) as avg_actual_delivery_in_days,
        floor(avg(estimated_delivery_in_days) over (partition by
customer_state)) as avg_estimated_delivery_in_days
from act_and_est_del_days),

get_diff_in_delivery_days as
(select customer_state, (avg_actual_delivery_in_days-
avg_estimated_delivery_in_days) as calc_diff_in_delivery_days
from get_avg_of_act_and_est_delivery),
```

```
get_min_delivery_days as
(select distinct(customer_state), min(calc_diff_in_delivery_days) over
(partition by customer_state) as min_delivery_days
from get_diff_in_delivery_days)

select customer_state, min_delivery_days, rank
from(select customer_state, min_delivery_days,
            dense_rank() over (order by min_delivery_days) as rank
     from get_min_delivery_days)
where rank <=5
order by 3
```

Output:

Here (-) minus indicates delivery has been done before the estimated date

Row	customer_state	min_delivery_days	rank
1	AC	-20.0	1
2	RO	-20.0	1
3	AP	-19.0	2
4	AM	-19.0	2
5	RR	-18.0	3
6	RS	-14.0	4
7	MT	-14.0	4
8	PE	-13.0	5
9	PA	-13.0	5
10	PR	-13.0	5

Insights:

a. AC, RO has the fastest delivery which is 20 days before the estimated date

Recommendation:

Though AC has showed a good delivery rate it is lacking access to the ports which can be enhanced as follows:

a. Enhance road, river and air cargo
b. Establish local and cross border ware house

6. **Analysis based on the payments:**
   1. **Find the month on month no. of orders placed using different payment types.**

Query:

```
with get_year_month_payment_type as
(select p.payment_type,
       extract(year from o.order_purchase_timestamp) as year,
       extract(month from o.order_purchase_timestamp) as month,
       o.order_id
from `Target_SQL_Business_Case.orders` as o
inner join `Target_SQL_Business_Case.payments` as p
on o.order_id = p.order_id)

select payment_type, year, month, count(order_id) as order_count
from get_year_month_payment_type
group by 1, 2, 3
order by 4 desc
```

Output:

Row	payment_type	year	month	order_count
1	credit_card	2017	11	5897
2	credit_card	2018	3	5691
3	credit_card	2018	1	5520
4	credit_card	2018	5	5497
5	credit_card	2018	4	5455
6	credit_card	2018	2	5253
7	credit_card	2018	8	4985
8	credit_card	2018	6	4813
9	credit_card	2018	7	4755
10	credit_card	2017	12	4377

Insights:

a. Most people have chosen credit_card for paying their orders

Recommendation:

Credit card companies can collaborate with retailers for cashbacks, discount etc

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

Query:

```
select payment_installments, count(order_id) as order_count
from `Target_SQL_Business_Case.payments`
where payment_installments >=1
group by 1
order by 1
```

Output:

Row	payment_installments	order_count
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

Insights:

a. Out of 99441 total orders in the dataset, 52546 orders i.e. approx. 52% of orders are purchased over EMI

Recommendations:

b. Promote EMI options
c. Offer transparent EMI plans
d. Reward loyalty who regular uses EMI