TARGET STUDY

Q1.A. Data type of all columns in the “customers” table.

```
Select
  column_name,
  DATA_TYPE
from "southern-idea-387512.Target_SQL.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:

Data Type of all the columns except customer_zip_code_prefix is “STRING” and the Data type of customer_zip_code_prefix is “INTEGER”.

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

```
select
  min(order_purchase_timestamp) as Min_time,
  max(order_purchase_timestamp) as Max_time
from `Target_SQL.orders`
```

Output:

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

Insights:

The first order was placed on 4th sept,2016 at 9:15 pm.
The last order was placed on 17th oct,2018 at 5:30 pm

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

```
select
  count(distinct customer_city) as no_of_city,
  count(distinct customer_state) as no_of_state
from `Target_SQL.customers`
```

Output:

Row	no_of_city	no_of_state
1	4119	27

Insights:

Total number of unique city and states in the dataset are 4119 and 27 respectively which have been ordering between the years 2016 and 2018.

II. In-depth Exploration:

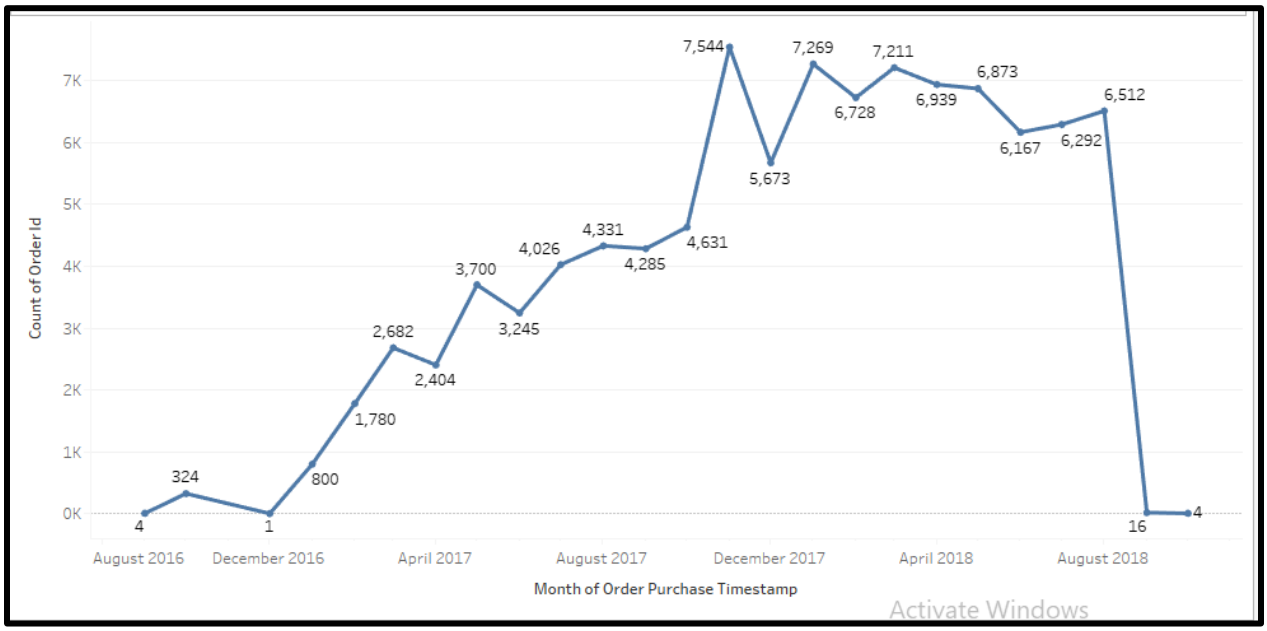
A. Is there a growing trend in the no. of orders placed over the past years?

```
SELECT
  format_timestamp("%Y-%B",order_purchase_timestamp) as Month,
  COUNT(order_id) as No_of_Orders,
FROM `Target_SQL.orders`
  GROUP BY Month
  ORDER BY Month
```

Output:

Row	Month	No_of_Orders
1	2016-December	1
2	2016-October	324
3	2016-September	4
4	2017-April	2404
5	2017-August	4331
6	2017-December	5673
7	2017-February	1780
8	2017-January	800
9	2017-July	4026
10	2017-June	3245
11	2017-March	2682

Insights:



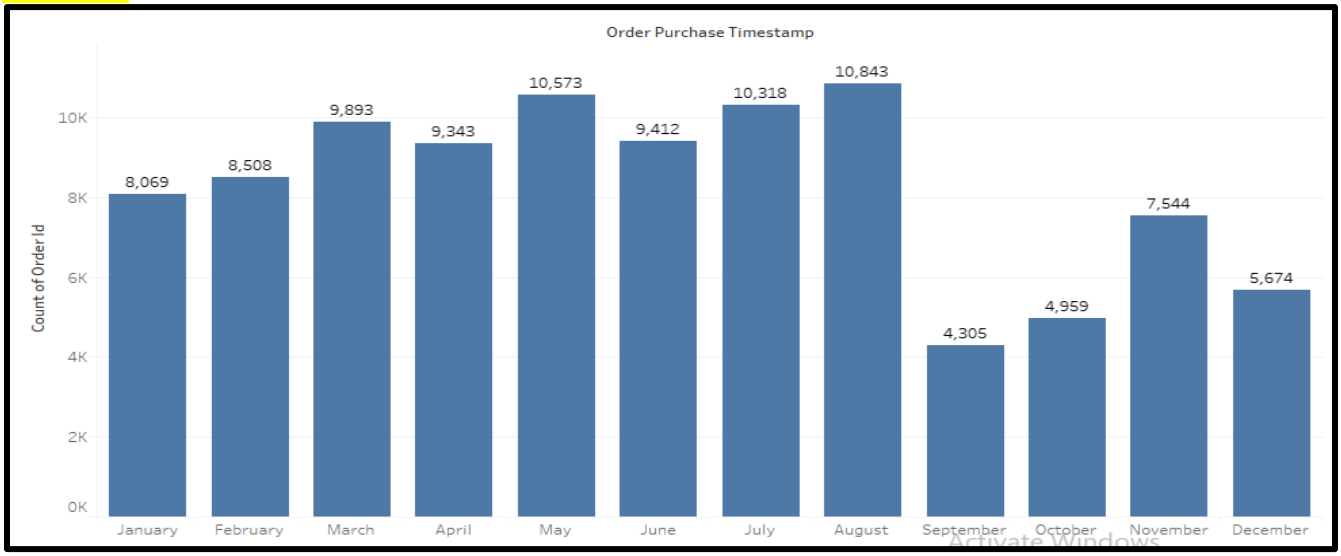
B. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

```
select
  extract(month from order_purchase_timestamp) as Month,
  format_date('%B',order_purchase_timestamp) as Month_Name,
  count(order_id) as no_of_orders
from `Target_SQL.orders`
  group by Month,Month_Name
  order by Month
```

Output:

Row	Month	Month_Name	no_of_orders
1	1	January	8069
2	2	February	8508
3	3	March	9893
4	4	April	9343
5	5	May	10573
6	6	June	9412
7	7	July	10318
8	8	August	10843
9	9	September	4305
10	10	October	4959
11	11	November	7544
12	12	December	5674

Insights:



Looking at the bar graph, we can determine the peaks on no. of orders for various months.

C. 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
  count(*) as no_of_orders,
  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 13 and 18 then
'Afternoon'
    else 'Night'
  end as time_of_the_day
from `Target_SQL.orders`
  group by time_of_the_day
  order by time_of_the_day
```

Output:

Row	no_of_orders	time_of_the_day
1	38135	Afternoon
2	5242	Dawn
3	27733	Mornings
4	28331	Night

Insights:

From the output we can infer that the most number of orders placed in the afternoon with total count of orders as 38135 and the least number of orders placed in the Dawn with 5242 orders.

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

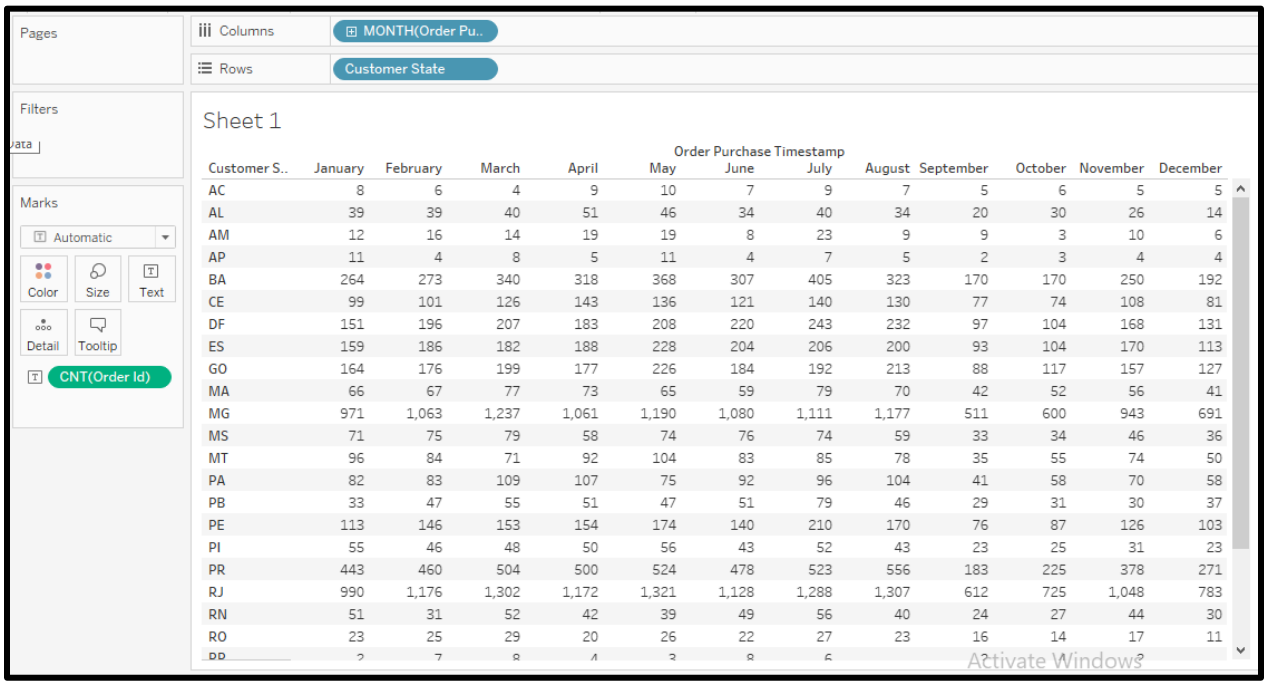
A. Get the month on month no. of orders placed in each state

```
select
  c.customer_state,
  extract(month from o.order_purchase_timestamp) as month,
  count(o.order_id) as no_of_orders
from `Target_SQL.orders` o
inner join
`Target_SQL.customers` c
on o.customer_id=c.customer_id
  group by c.customer_state,month
  order by c.customer_state,month desc;
```

Output:

Row	customer_state	month	no_of_orders
1	AC	12	5
2	AC	11	5
3	AC	10	6
4	AC	9	5
5	AC	8	7
6	AC	7	9
7	AC	6	7
8	AC	5	10
9	AC	4	9
10	AC	3	4
11	AC	2	6
12	AC	1	8

Insights:



Order Purchase Timestamp

Customer S..	January	February	March	April	May	June	July	August	September	October	November	December
AC	8	6	4	9	10	7	9	7	5	6	5	5
AL	39	39	40	51	46	34	40	34	20	30	26	14
AM	12	16	14	19	19	8	23	9	9	3	10	6
AP	11	4	8	5	11	4	7	5	2	3	4	4
BA	264	273	340	318	368	307	405	323	170	170	250	192
CE	99	101	126	143	136	121	140	130	77	74	108	81
DF	151	196	207	183	208	220	243	232	97	104	168	131
ES	159	186	182	188	228	204	206	200	93	104	170	113
GO	164	176	199	177	226	184	192	213	88	117	157	127
MA	66	67	77	73	65	59	79	70	42	52	56	41
MG	971	1,063	1,237	1,061	1,190	1,080	1,111	1,177	511	600	943	691
MS	71	75	79	58	74	76	74	59	33	34	46	36
MT	96	84	71	92	104	83	85	78	35	55	74	50
PA	82	83	109	107	75	92	96	104	41	58	70	58
PB	33	47	55	51	47	51	79	46	29	31	30	37
PE	113	146	153	154	174	140	210	170	76	87	126	103
PI	55	46	48	50	56	43	52	43	23	25	31	23
PR	443	460	504	500	524	478	523	556	183	225	378	271
RJ	990	1,176	1,302	1,172	1,321	1,128	1,288	1,307	612	725	1,048	783
RN	51	31	52	42	39	49	56	40	24	27	44	30
RO	23	25	29	20	26	22	27	23	16	14	17	11

We can see the no. of orders placed by each state in different-different months.

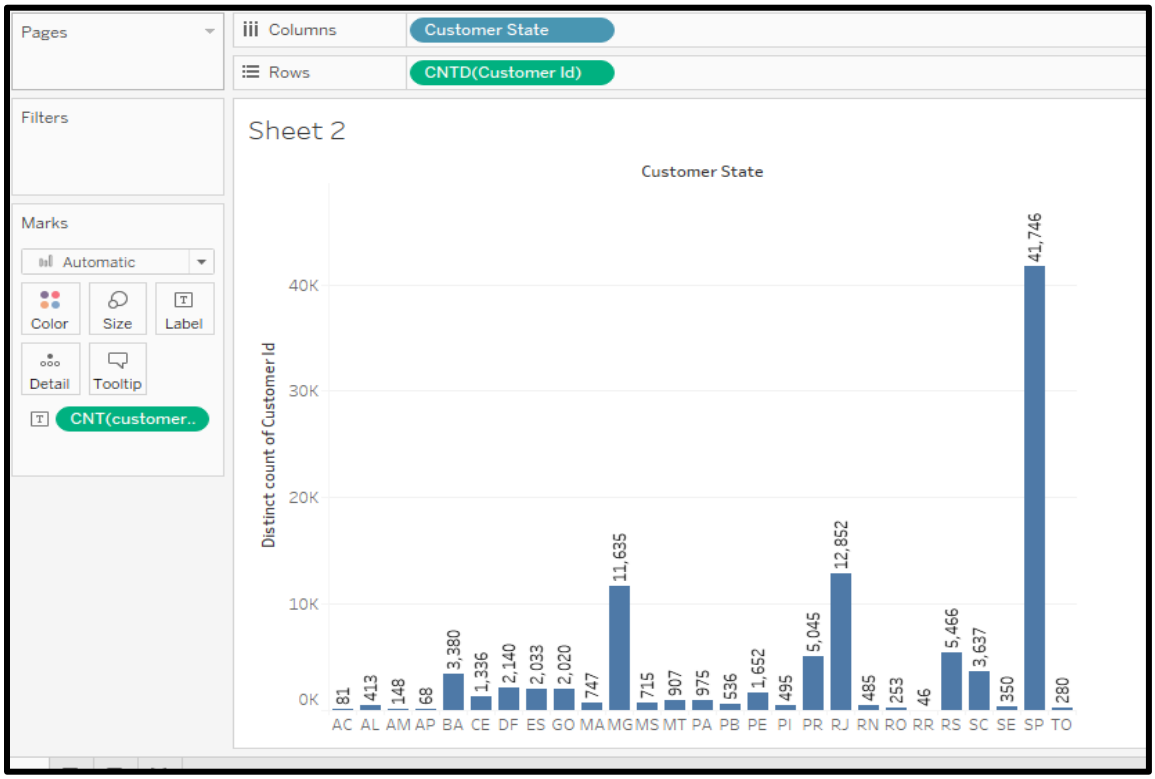
B. How are the customers distributed across all the states?

```
select
  customer_state,
  count(customer_unique_id) as no_of_unique_customers
from `Target_SQL.customers`
  group by customer_state
  order by customer_state
```

Output:

Row	customer_state	no_of_unique_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
11	MG	11635
12	MS	715

Insights:



We can see that the highest no. of customers is in SP with the count 41,746 and least is in RR with the count 46.

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

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

```
with cost_per_year as
(
select
  extract (year from o.order_purchase_timestamp) as Year,
  round(sum(payment_value)) Cost_of_orders
from `Target_SQL.payments` p join
`Target_SQL.orders` o
on p.order_id = o.order_id
  where extract (month from o.order_purchase_timestamp) between 1 and 8
  group by Year
  order by Year
)
select
  round((y2.Cost_of_orders/y1.Cost_of_orders-1)*100) as Percentage_Increase
from cost_per_year y1 join cost_per_year y2
on y2.year= y1.year+1;
```

Output:

Row	Percentage_Increase
1	137.0

Insights:

We get the percentage increase from year 2017 to 2018 is 137 %

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

```
select
  c.customer_state,
  round(sum(oi.price),2) as total_price,
  round(avg(oi.price),2) as average_price
from `Target_SQL.customers` c
inner join
`Target_SQL.orders` o
on c.customer_id = o.customer_id
inner join
`Target_SQL.order_items` oi
on o.order_id=oi.order_id
  group by c.customer_state
  order by c.customer_state
```

Output:

Row	customer_state	total_price	average_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63

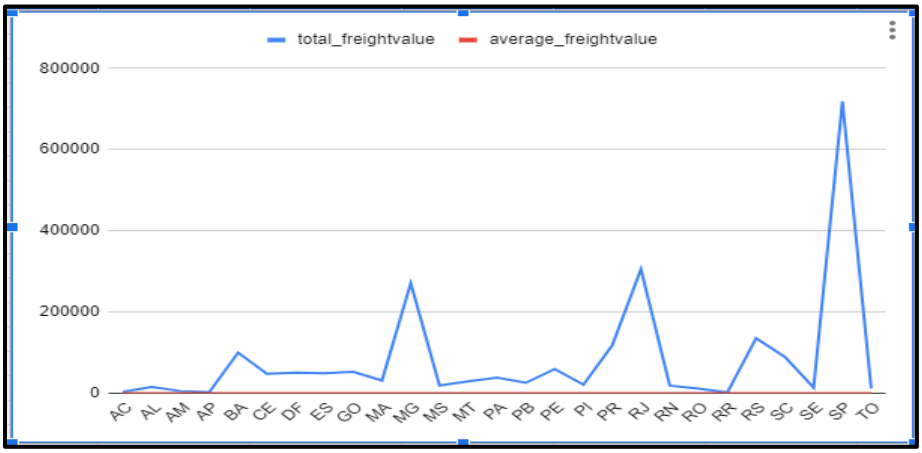
C. Calculate the Total & Average value of order freight for each state.

```
select
  c.customer_state,
  round(sum(oi.freight_value),2) as total_freightvalue,
  round(avg(oi.freight_value),2) as average_freightvalue
from `Target_SQL.customers` c
inner join
`Target_SQL.orders` o
on c.customer_id = o.customer_id
inner join
`Target_SQL.order_items` oi
on o.order_id=oi.order_id
  group by c.customer_state
  order by c.customer_state
```

Output:

Row	customer_state	total_freightvalue	average_freightvalue
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

Insights:



V. Analysis based on sales, freight and delivery time.

A. 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(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_deliver,
  date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery
from `Target_SQL.orders`
  where order_delivered_customer_date is not null and order_purchase_timestamp is not
null and order_estimated_delivery_date is not null
  order by order_id
```

Output:

Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18
11	00061f2a7bc09da83e415a52d...	4	10
12	00063b381e2406b52ad42947...	10	0

Row	order_id	time_to_deliver	diff_estimated_delivery
20	000e63d38ae8c00bbcb5a3057...	3	8
21	000e906b789b55f64edcb1f84...	17	-2
22	000f25f4d72195062c040b12d...	15	19
23	001021efaa8636c29475e7734...	9	14
24	0010b2e5201cc5f1ae7e9c6cc...	11	3
25	00119ff934e539cf26f92b9ef0...	10	14
26	0011d82c4b53e22e84023405f...	10	19
27	00125cb692d0488780980661...	15	12
28	00130c0eee84a3d909e75bc08...	1	6
29	0013503b13da1eac68621939...	14	7
30	00137e170939bba5a3134e23...	17	6
31	001427c0ec99cf8af737bd88e...	18	14

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

```
with base as

(SELECT
  DISTINCT c.customer_state as Top_5_States,
  round(AVG(oi.freight_value) OVER(PARTITION BY c.customer_state),2) AS avg_price
FROM `Target_SQL.customers` c
join
`Target_SQL.orders` o
on c.customer_id=o.customer_id
join
`Target_SQL.order_items` oi on
o.order_id = oi.order_id),

top_5 as
(select
  base.Top_5_States,
  base.avg_price as Highest_Avg_Freight_Value,
  row_number() over (order by avg_price desc) as rn
from base),

bottom_5 as
(select
  base.Top_5_States as Bottom_5_States,
  base.avg_price as Lowest_Avg_Freight_Value,
  row_number() over (order by avg_price) as rn
from base)

select * except (rn) from top_5 join bottom_5 on top_5.rn = bottom_5.rn
limit 5;
```

Output:

Row	Top_5_States	Highest_Avg_Freight	Bottom_5_States	Lowest_Avg_Freight
1	RR	42.98	SP	15.15
2	PB	42.72	PR	20.53
3	RO	41.07	MG	20.63
4	AC	40.07	RJ	20.96
5	PI	39.15	DF	21.04

Insights:

RR being the state with High average freight value and SP being the state with lowest average freight value

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

```
with base as
(SELECT DISTINCT c.customer_state,
ceil(AVG(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day))
OVER(PARTITION BY c.customer_state)) AS avg_delivery_time
FROM `Target_SQL.customers` c
join `Target_SQL.orders` o
on c.customer_id=o.customer_id),

top_5 as
(select dense_rank() over (order by avg_delivery_time desc) as
Top_5_Rank,customer_state as Top_5_State,
base.avg_delivery_time as Top_avg,
row_number() over (order by avg_delivery_time desc) as rn
from base ),

bottom_5 as
(select dense_rank() over (order by avg_delivery_time) as Bottom_5_Rank,
customer_state as Bottom_5_State,
base.avg_delivery_time as bottom_avg ,
row_number() over (order by avg_delivery_time) as rn
from base)

select * except(rn) from top_5 join bottom_5 on top_5.rn = bottom_5.rn
where top_5.Top_5_Rank <=5 or bottom_5.Bottom_5_Rank <=5
order by top_5.Top_5_Rank, bottom_5.Bottom_5_Rank ;
```

Output:

Row	Top_5_Rank	Top_5_State	Top_avg	Bottom_5_Rank	Bottom_5_State	bottom_avg
1	1	RR	29.0	1	SP	9.0
2	2	AP	27.0	2	MG	12.0
3	3	AM	26.0	2	PR	12.0
4	4	AL	25.0	3	DF	13.0
5	5	PA	24.0	4	RS	15.0
6	6	SE	22.0	4	RJ	15.0
7	6	MA	22.0	4	SC	15.0
8	7	CE	21.0	5	GO	16.0
9	7	AC	21.0	5	ES	16.0
10	8	PB	20.0	5	MS	16.0

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

```
with Avg_Del as

(SELECT
  distinct dense_rank() over (order by Avg_Del_Est_diff desc) Rank,
  customer_state,
  Table1.Avg_Est_Del_time,
  Table1.Avg_Act_DeliveryTime ,

  CASE
  WHEN Table1.Avg_Del_Est_diff = 0 THEN 'On-Time'
  WHEN Table1.Avg_Del_Est_diff > 0  THEN concat( Table1.Avg_Del_Est_diff,' Days
Fast')
  ELSE  concat(abs(Table1.Avg_Del_Est_diff),' Days Late')
END as Delivery_Status
FROM
(
  SELECT
    c.customer_state,
    ceil(avg(timestamp_diff(o.order_estimated_delivery_date,o.order_purchase_timestam
p,day)) over(partition by c.customer_state)) as Avg_Est_Del_time ,
    ceil(avg(timestamp_diff(o.order_delivered_customer_date,o.order_purchase_timestam
p, day)) over(partition by c.customer_state)) as Avg_Act_DeliveryTime,
    ceil(avg(timestamp_diff(o.order_estimated_delivery_date,order_delivered_customer_
date,day)) over(partition by c.customer_state)) as Avg_Del_Est_diff
  FROM `Target_SQL.orders` o join `Target_SQL.customers` c on o.customer_id =
c.customer_id
  WHERE o.order_delivered_customer_date is not null AND o.order_purchase_timestamp is
not null AND o.order_estimated_delivery_date is not null
) as Table1)
select *
from Avg_Del
where Rank <=5
order by Rank;
```

Output:

Row	Rank	customer_state	Avg_Est_Del_time	Avg_Act_DeliveryTim	Delivery_Status
1	1	AC	41.0	21.0	20 Days Fast
2	1	RO	39.0	19.0	20 Days Fast
3	2	AM	45.0	26.0	19 Days Fast
4	2	AP	46.0	27.0	19 Days Fast
5	3	RR	46.0	29.0	17 Days Fast
6	4	MT	32.0	18.0	14 Days Fast
7	4	PA	37.0	24.0	14 Days Fast
8	5	RS	29.0	15.0	13 Days Fast

VI. Analysis based on the payments:

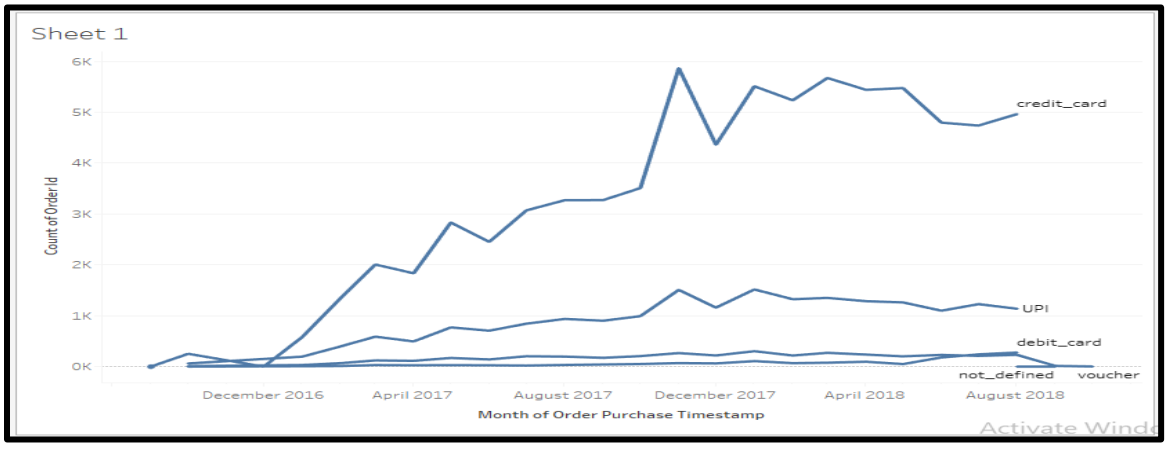
A. Find the month on month no. of orders placed using different payment types.

```
select
  count(o.order_id) as no_of_orders,
  p.payment_type,
  format_date ('%Y-%B', o.order_purchase_timestamp) as month
from `Target_SQL.orders` o
join
`Target_SQL.payments` p
on o.order_id=p.order_id
group by p.payment_type,month
order by month desc;
```

Output:

Row	no_of_orders	payment_type	month
1	15	voucher	2018-September
2	1	not_defined	2018-September
3	4	voucher	2018-October
4	5497	credit_card	2018-May
5	1263	UPI	2018-May
6	51	debit_card	2018-May
7	324	voucher	2018-May
8	5691	credit_card	2018-March
9	1352	UPI	2018-March
10	78	debit_card	2018-March
11	391	voucher	2018-March
12	4813	credit_card	2018-June

Insights:



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

```
SELECT
COUNT(DISTINCT order_id) AS no_of_order
FROM
`Target_SQL.payments`
WHERE
payment_installments >= 1
```

Output:

Row	no_of_order
1	99438

Insights:

Total no. of Order Ids whose payment installments have been paid is 99438.