Target Business Analysis

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

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

Query:

```
SELECT column_name, data_type FROM `scaler-dsml-rc.Scaler_SQL_Project`.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

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

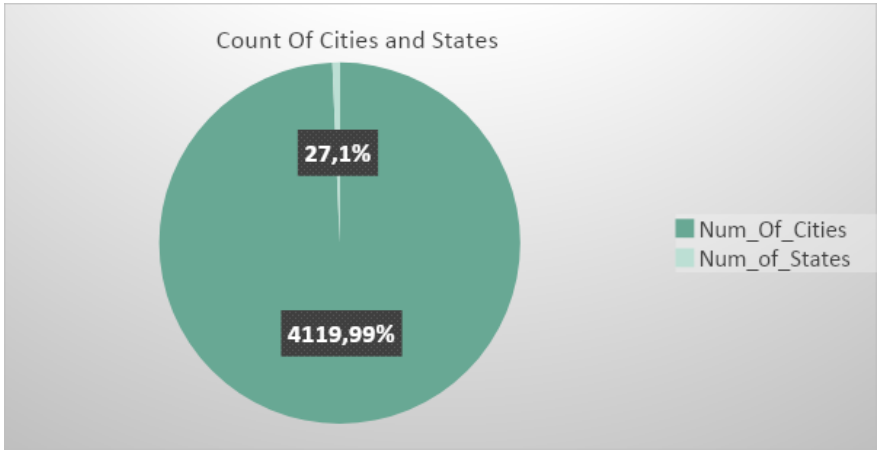
Query:

```
select min(order_purchase_timestamp) as First_Order,max(order_purchase_timestamp) as Last_order
from ` scaler-dsml-rc.Scaler_SQL_Project.orders`
```

Output:

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

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



Query:

```
select Count(distinct customer_city) as Num_Of_Cities,Count(distinct customer_state) as Num_of_States
from `scaler-dsml-rc.Scaler_SQL_Project.customers`
```

Output:

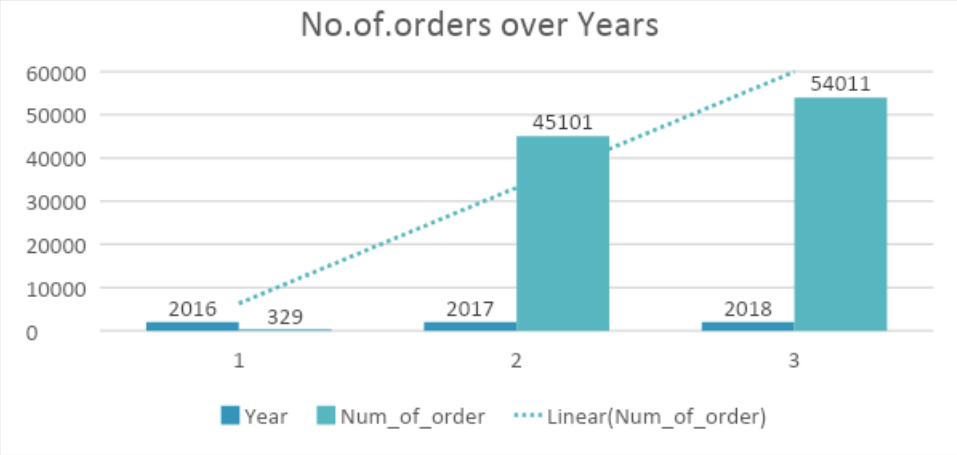
Row	Num_Of_Cities	Num_of_States
1	4119	27

From the Part 1 analysis,

1. **We can see that Customers table has 5 columns in which 4 columns are string type and 1 int.**
2. **Orders from this dataset were placed between Sep 2016 to Oct 2018.**
3. **Total count of City is 4119 and no of states are 27.**

II. In-depth Exploration:

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



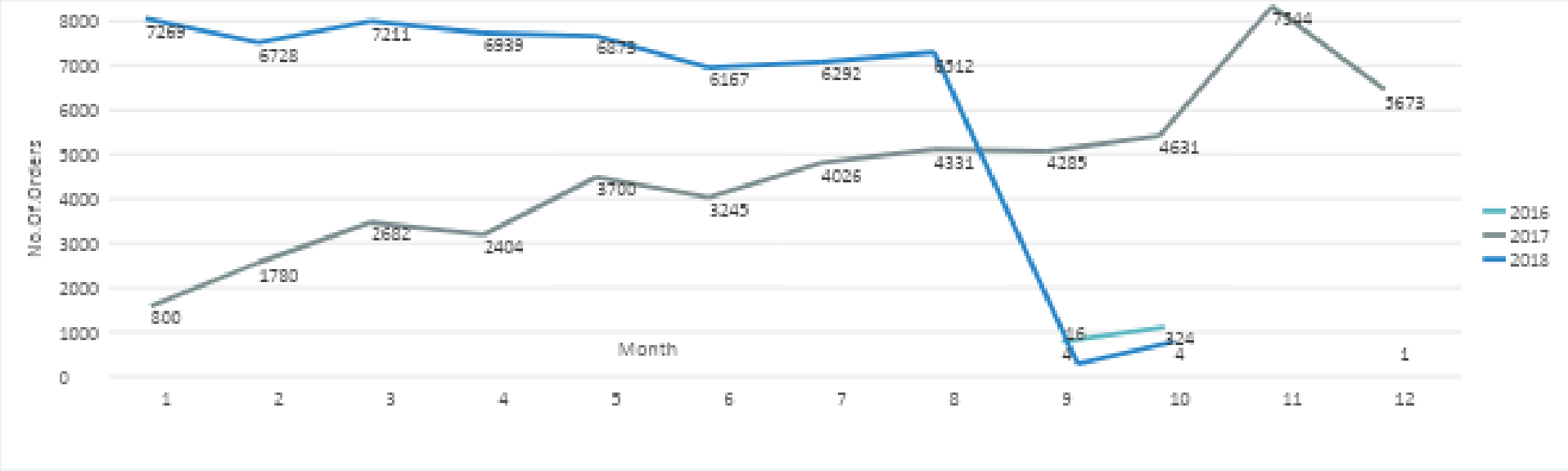
Query:

```
select  Extract(year from order_purchase_timestamp) as Year,count(order_id) as Num_of_order
from `scaler-dsml-rc.Scaler_SQL_Project.orders`
GROUP BY Year
order by Year
```

Output:

Row	Year	Num_of_order
1	2016	329
2	2017	45101
3	2018	54011

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



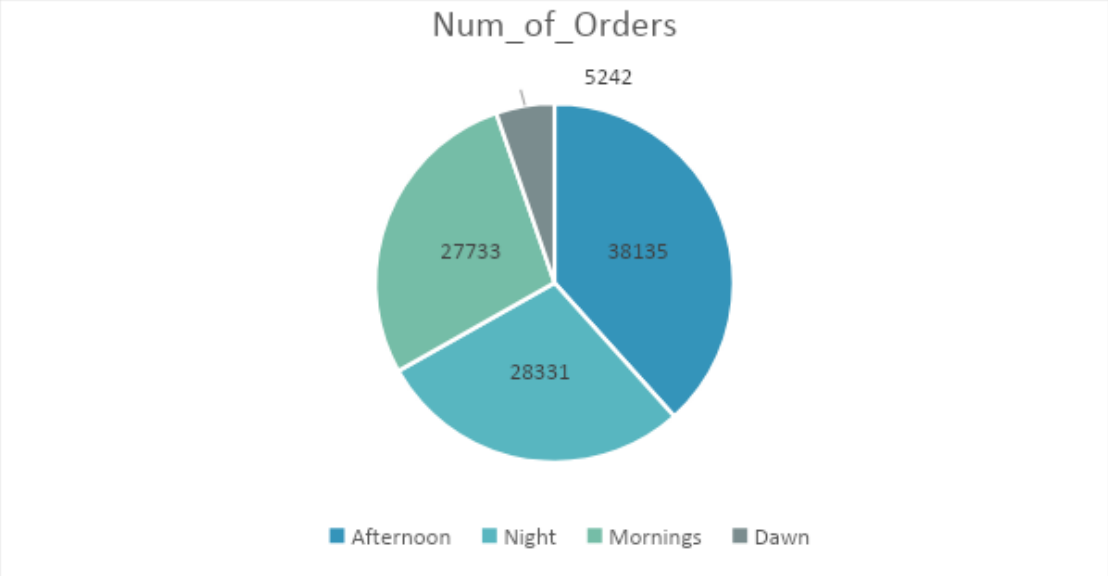
Query:

```
select extract(year from order_purchase_timestamp) as Year,extract(Month from order_purchase_timestamp) as Month,count(order_id) as num_of_orders
from `scaler-dsml-rc.Scaler_SQL_Project.orders`
group by Year,Month
order by num_of_orders desc
```

Output:

Row	Year	Month	num_of_orders
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873
6	2018	2	6728
7	2018	8	6512
8	2018	7	6292
9	2018	6	6167
10	2017	12	5673

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



Query:

```
select 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"
when Extract(hour from order_purchase_timestamp) between 19 and 23 then "Night"
End as Time,Count(order_id) as Num_of_Orders
FROM `scaler-dsml-rc.Scaler_SQL_Project.orders`
group by Time
```

```
order by Num_of_Orders desc
```

Output:

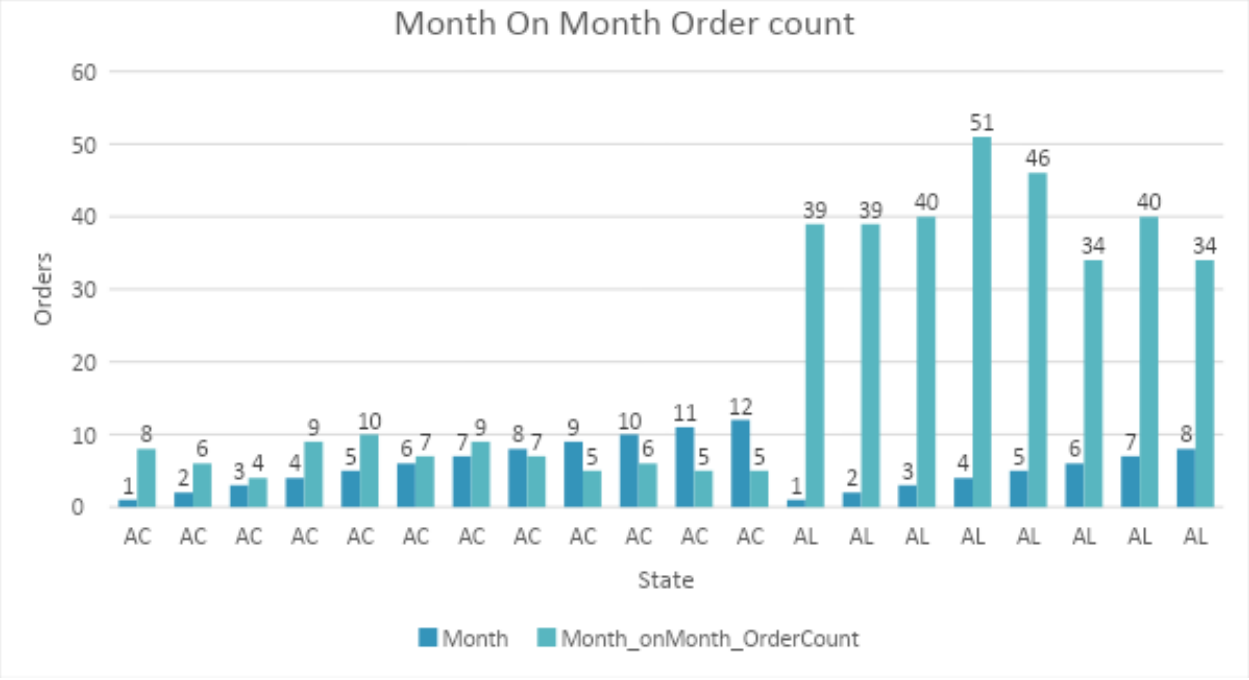
Row	Time	Num_of_Orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Part II Analysis,

1. **From the above analysis, we can see the growing trend in the number of orders over the given years.**
2. **We can see that no of orders have increased from Nov 2017 to Mid of 2018.**
3. **From the above analysis, we can see that highest orders were placed during Afternoon time.**

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

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



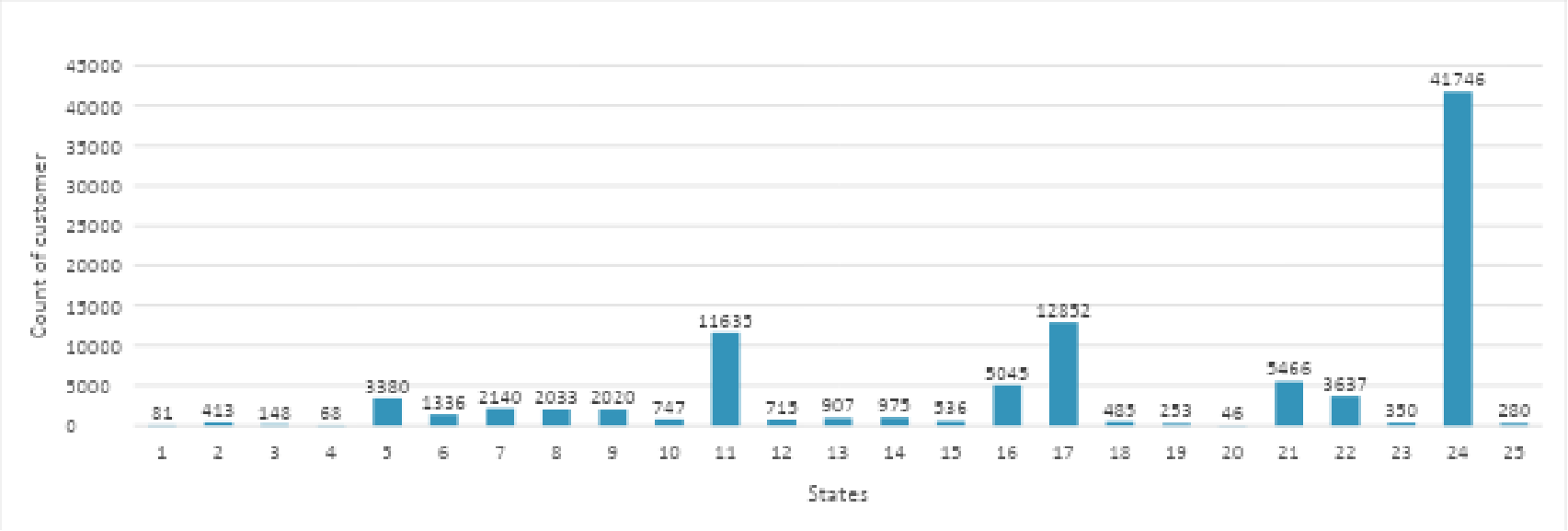
Query:

```
select c.customer_state as State,Extract(month from o.order_purchase_timestamp) as Month, count(o.order_id) as Month_onMonth_OrderCount
from `scaler-dsml-rc.Scaler_SQL_Project.orders` o
left join `scaler-dsml-rc.Scaler_SQL_Project.customers` c
on o.customer_id = c.customer_id
group by State,Month
order by State,Month
```

Output:

Row	State	Month	Month_onMonth_OrderCount
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6

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



Query:

```
select customer_state,count(distinct customer_id) as Count_of_customer
FROM `scaler-dsml-rc.Scaler_SQL_Project.customers`
group by customer_state
order by customer_state
```

Output:

Row	customer_state	Count_of_customer
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

Part III Analysis,

1. **Around 10 states have less than or equal to 500 orders cumulatively on monthly basis and around 17 states have more than 500 orders cumulatively on monthly basis.**
2. **Customer are distributed across 27 states in which 3 state MG, RJ and SP have more than 10000 count and PR and RS has count more than 5000.**

Totally 7 states have less than 5000 customers distributed in each of them but greater than 1000 and 15 States that have count less than 1000.

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).

Query:

```
select *,round(((Total_sales-previous)/previous)*100,2) as Percent_increase from
(select extract(year from o.order_purchase_timestamp) as Year,
round(sum(p.payment_value),2)  as Total_sales,round(lag(sum(p.payment_value)) over(order by sum(p.payment_value)),2) as previous
from `scaler-dsml-rc.Scaler_SQL_Project.orders` o
join `scaler-dsml-rc.Scaler_SQL_Project.payments` p on o.order_id=p.order_id
where (extract(year from o.order_purchase_timestamp) = 2017 or extract(year from o.order_purchase_timestamp) = 2018) and extract(month
from o.order_purchase_timestamp) between 1 and 8
group by Year) t1
```

Output:

Row	Year	Total_sales	previous	Percent_increase
1	2017	3669022.12	*null*	*null*
2	2018	8694733.84	3669022.12	136.98

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

Query:

```
select C.customer_state,round(sum(OI.price),2) as TotalPrice,round(Avg(OI.price),2) as AvgPrice
FROM `scaler-dsml-rc.Scaler_SQL_Project.customers` C
join  `scaler-dsml-rc.Scaler_SQL_Project.orders` O on C.customer_id = O.customer_id
join `scaler-dsml-rc.Scaler_SQL_Project.order_items` OI on O.order_id=OI.order_id
group by C.customer_state
order by C.customer_state
```

Output:

Row	customer_state	TotalPrice	AvgPrice
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

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

Query:

```
select C.customer_state,round(sum(OI.freight_value),2) as TotalFreightVal,round(Avg(OI.freight_value),2) as AvgPrice
FROM `scaler-dsml-rc.Scaler_SQL_Project.customers` C
join  `scaler-dsml-rc.Scaler_SQL_Project.orders` O on C.customer_id = O.customer_id
join `scaler-dsml-rc.Scaler_SQL_Project.order_items` OI on O.order_id=OI.order_id
group by C.customer_state
order by TotalFreightVal,AvgPrice
```

Output:

Row	customer_state	TotalPrice	AvgPrice
1	RR	2235.19	42.98
2	AP	2788.5	34.01
3	AC	3686.75	40.07
4	AM	5478.89	33.21
5	RO	11417.38	41.07
6	TO	11732.68	37.25
7	SE	14111.47	36.65
8	AL	15914.59	35.84
9	RN	18860.1	35.65
10	MS	19144.03	23.37

Part IV Analysis,

1. **In the year 2018 increase % of cost is 136.98 whereas 2017 don't have any data since the previous year data unavailable.**
2. **In Total, 17 states have Total price greater than 100000.**
3. **Similarly, 6 states have freight value greater than 100000.**

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.

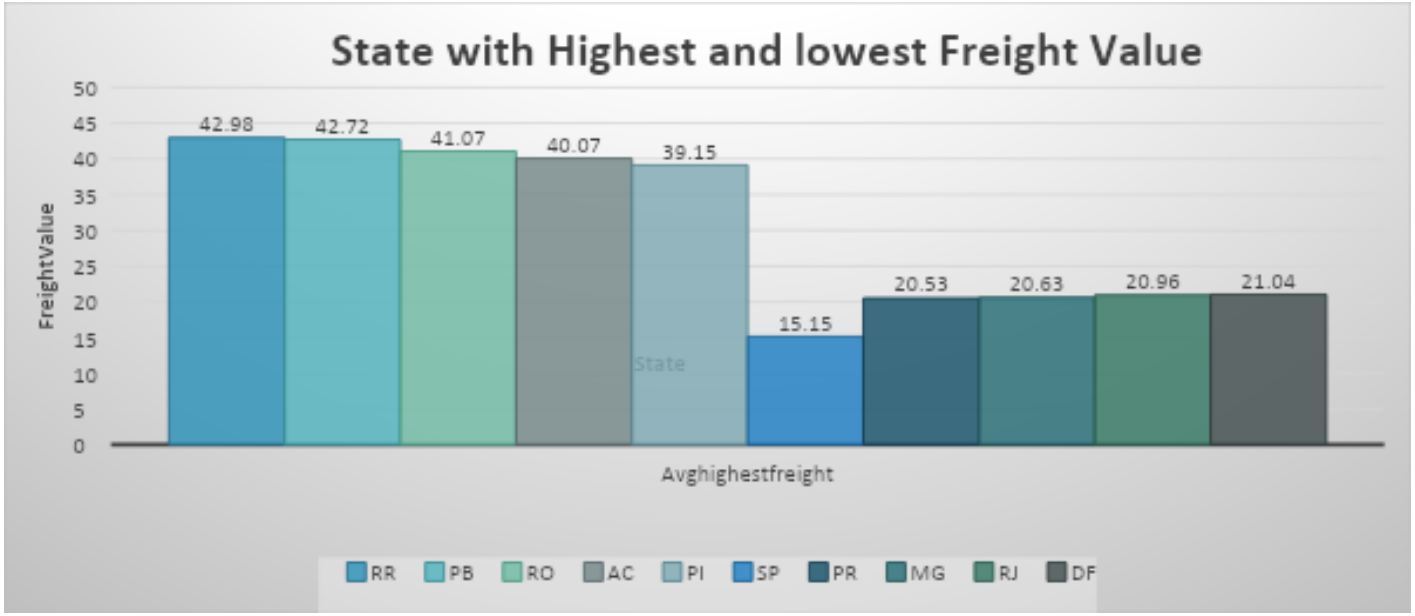
Query:

```
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 `scaler-dsml-rc.Scaler_SQL_Project.orders`
```

Output:

Row	order_id	Time_To_Deliver	diff_estimated_delive
1	1950d777989f6a877539f5379...	30	-12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	28
3	65d1e226dfaeb8cdc42f66542...	35	16
4	635c894d068ac37e6e03dc54e...	30	1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	1
7	276e9ec344d3bf029ff83a161c...	43	-4
8	54e1a3c2b97fb0809da548a59...	40	-4
9	fd04fa4105ee8045f6a0139ca5...	37	-1
10	302bb8109d097a9fc6e9cefc5...	33	-5

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



Query:

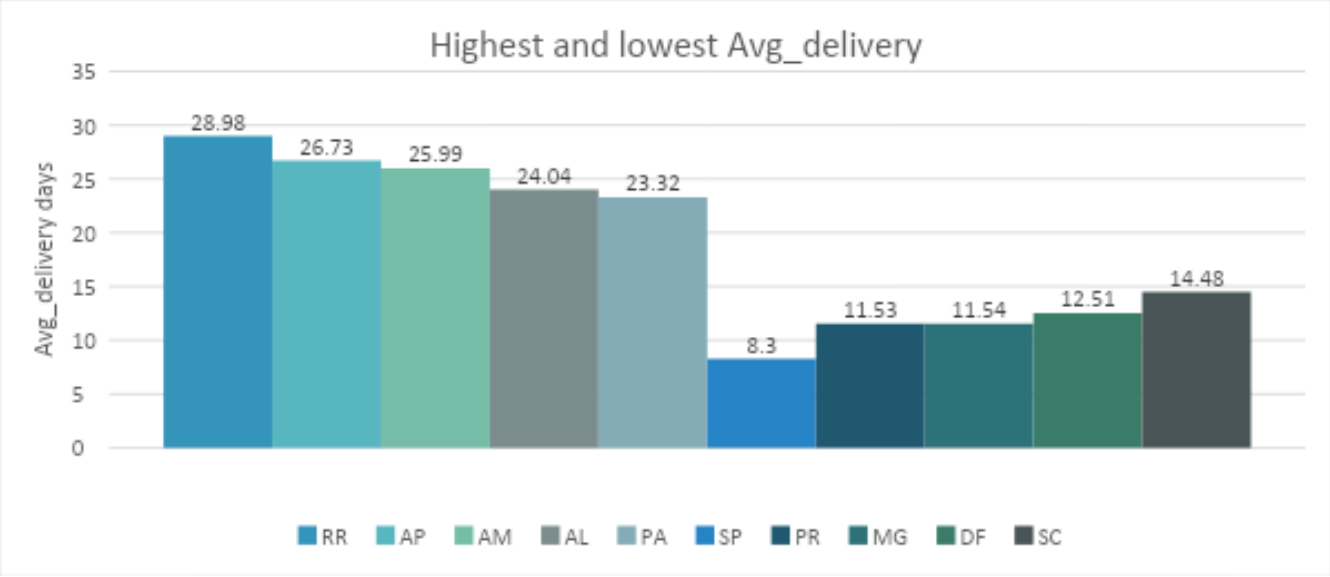
```
with avgfreight as (select c.customer_state,round(avg(oi.freight_value),2) as Highest_freightvalue, Row_number() over(order by
round(avg(oi.freight_value),2)) as lowest,Row_number() over(order by round(avg(oi.freight_value),2) desc) as highest
FROM `scaler-dsml-rc.Scaler_SQL_Project.customers` c
left join `scaler-dsml-rc.Scaler_SQL_Project.orders` o on c.customer_id = o.customer_id
join `scaler-dsml-rc.Scaler_SQL_Project.order_items` oi on o.order_id= oi.order_id
group by c.customer_state
```

```
) select a.customer_state as StateWithHighestFreightVal,a.Highest_freightvalue as Avghighestfreight,b.customer_state as StateWithLowestFreightVal,b.Highest_freightvalue as Avglowestfreight from avgfreight a inner join avgfreight b on a.highest =b.lowest limit 5
```

Output:

Row	StateWithHighestFreightVal	Avghighestfreight	StateWithLowestFreightVal	Avglowestfreight
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

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



Query:

```
select * from
(select c.customer_state,round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2) as avg_delivery_highest,
row_number() over(order by avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day))desc) rownum_highest
from  `scaler-dsml-rc.Scaler_SQL_Project.customers` c
```

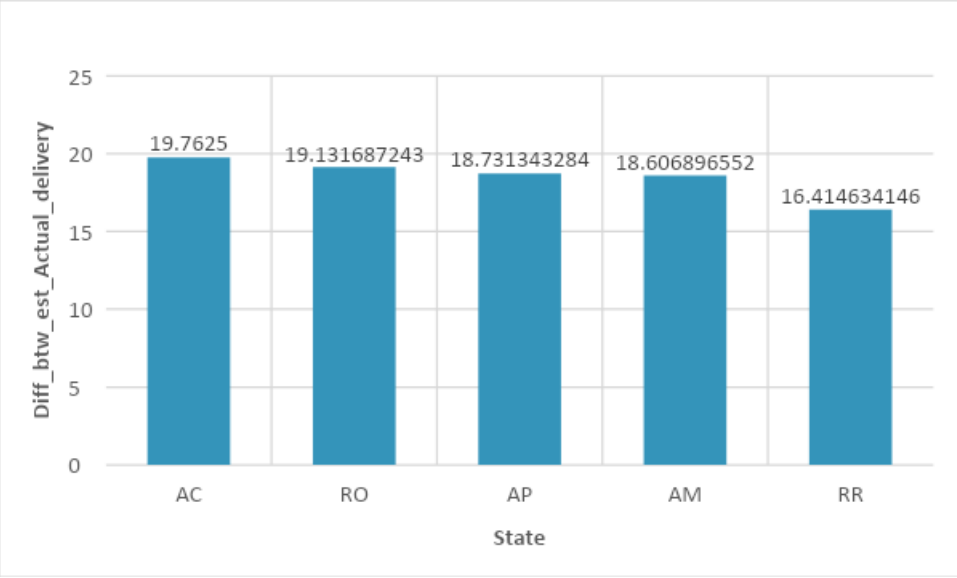
```
join `scaler-dsml-rc.Scaler_SQL_Project.orders` o on c.customer_id=o.customer_id
group by c.customer_state
order by avg_delivery_highest desc ) t1
join(select c.customer_state,round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2) as
avg_delivery_lowest,row_number() over(order by avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day))asc)
rownum_lowest
from  `scaler-dsml-rc.Scaler_SQL_Project.customers` c
join `scaler-dsml-rc.Scaler_SQL_Project.orders` o on c.customer_id=o.customer_id
group by c.customer_state
order by avg_delivery_lowest desc) t2 on t1.rownum_highest=t2.rownum_lowest
limit 5
```

Output:

Row	customer_state	avg_delivery_highest	rownum_highest	customer_state_1	avg_delivery_lowest	rownum_lowest
1	RR	28.98	1	SP	8.3	1
2	AP	26.73	2	PR	11.53	2
3	AM	25.99	3	MG	11.54	3
4	AL	24.04	4	DF	12.51	4
5	PA	23.32	5	SC	14.48	5

D. 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.



Query:

```
select c.customer_state,round(avg(Date_Diff(order_estimated_delivery_date,order_delivered_customer_date,day)),2) as
Diff_order_estimated_act_delivery
```

```
FROM `scaler-dsml-rc.Scaler_SQL_Project.customers` c
left join `scaler-dsml-rc.Scaler_SQL_Project.orders` o on c.customer_id=o.customer_id
where o.order_status='delivered'
group by c.customer_state
order by Diff_order_estimated_act_delivery desc
```

Output:

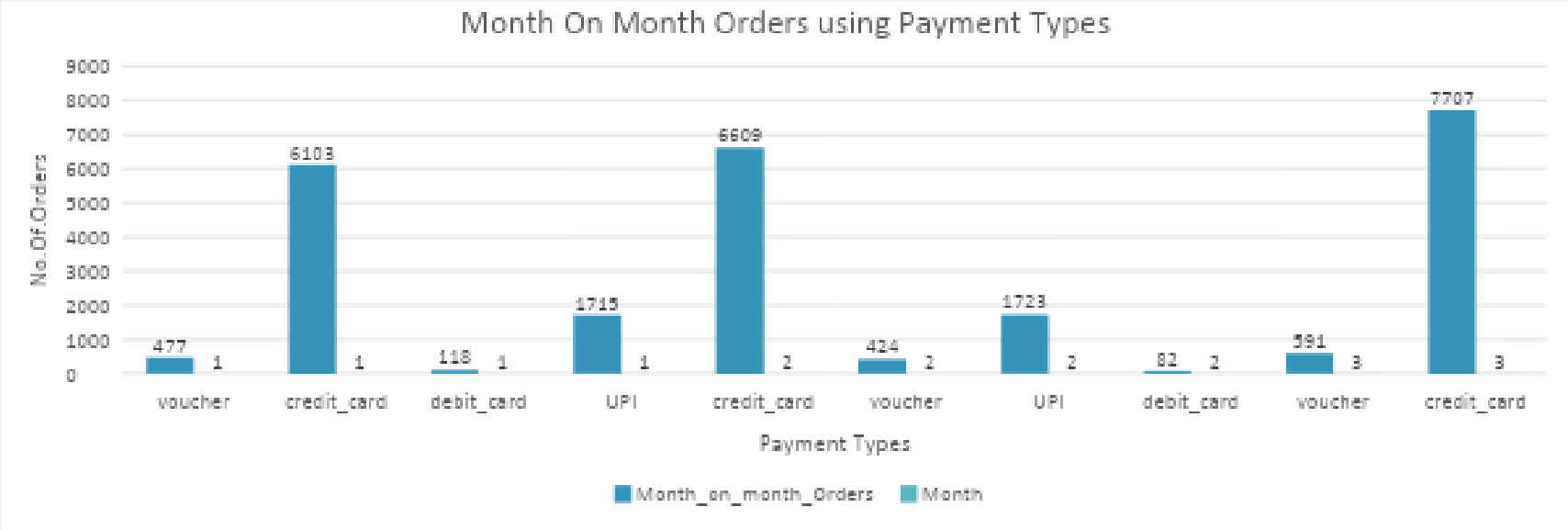
Row	customer_state	Diff_order_estimated
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Part V Analysis,

1. **From the above analysis, we can understand that some of the deliveries were delayed.**
2. **Top 5 states with highest Avg freight values are RR, PB, RO, AC and PI and the states with the lowest freight values are SP, PR, MG, RJ and DF.**
3. **States with highest delivery time are RR, AP, AM, AL and PA and the states with lowest delivery time are SP, PR, MG, DF and SC.**
4. **5 states that have fastest delivery compared to estimated delivery are AC, RO, AP, AM and RR.**

VI. Analysis based on the payments:

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



Query:

```
select P.payment_type,Extract(month from o.order_purchase_timestamp) as Month,count(o.order_id) as Month_on_month_Orders
FROM `scaler-dsml-rc.Scaler_SQL_Project.payments` P
join `scaler-dsml-rc.Scaler_SQL_Project.orders` o on P.order_id=o.order_id
group by P.payment_type,Month
order by Month
```

Output:

Row	payment_type	Month	Month_on_month_Or
1	voucher	1	477
2	credit_card	1	6103
3	debit_card	1	118
4	UPI	1	1715
5	credit_card	2	6609
6	voucher	2	424
7	UPI	2	1723
8	debit_card	2	82
9	voucher	3	591
10	credit_card	3	7707

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

Query:

```
select payment_installments,count(order_id) as Num_of_orders FROM `scaler-dsml-rc.Scaler_SQL_Project.payments`
group by payment_installments
order by payment_installments desc
```

Output:

Row	payment_installment	Num_of_orders
1	24	18
2	23	1
3	22	1
4	21	3
5	20	17
6	18	27
7	17	8
8	16	5
9	15	74
10	14	15
11	13	16
12	12	133
13	11	23
14	10	5328
15	9	644

Part VI Analysis,

1. **Payments done during March, May, July and August have more than 10000 orders and rest of the months have less than 10000.**
2. **Orders have been received for all the payment instalments except for 19.**

Submitted On: 23-06-2023

RAMA CHITRA M