

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

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

Ans:

```
SELECT *, data_type
FROM `shubham-scaler.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name= 'Customers'
```

Output:

column_name	ordinal_position	is_nullable	data_type
customer_id	1	YES	STRING
customer_unique_id	2	YES	STRING
customer_zip_code_prefix	3	YES	INT64
customer_city	4	YES	STRING
customer_state	5	YES	STRING

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

Ans:

```
SELECT MIN(order_purchase_timestamp) Start_time, MAX(order_purchase_timestamp) End_time
FROM `Target.Orders`
```

Output:

Start_time	End_time
2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

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

Ans:

```
SELECT Count(distinct customer_city) Cities, Count(distinct customer_state) States
FROM `Target.Customers` C JOIN `Target.Orders` O ON C.customer_id=O.customer_id
```

Output:

Row	Cities	States
1	4119	27

## 2.In-depth Exploration:

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

Ans:

```
WITH Yearly_orders AS
(SELECT EXTRACT(YEAR from order_purchase_timestamp) Order_year, Count(distinct order_id)
Total_orders
FROM `Target.Orders`
Group BY Order_year)
SELECT Order_year, Total_orders, LAG(Total_orders)OVER(ORDER BY Order_year)
Previous_year_orders,
ROUND((Total_orders/LAG(Total_orders)OVER(ORDER BY Order_year)*100),2) Yearly_growth
FROM Yearly_orders
ORDER BY Order_year
```

Output:

Row	Order_year	Total_orders	Previous_year_orders	Yearly_growth
1	2016	329	null	null
2	2017	45101	329	13708.51
3	2018	54011	45101	119.76

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

Ans:

```
SELECT EXTRACT(MONTH FROM order_purchase_timestamp) Monthly_orders, COUNT(order_id)
Total_orders
FROM `Target.Orders`
GROUP BY Monthly_orders
ORDER BY Monthly_orders
```

Output:

Row	Monthly_orders	Total_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959

2.3 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

Ans:

```
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 'Morning'
WHEN EXTRACT(Hour FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
ELSE 'Night' END Time_of_order,
COUNT(order_id) Total_orders
FROM `Target.Orders`
GROUP BY Time_of_order
ORDER BY Total_orders DESC
LIMIT 1
```

Output:

Row	Time_of_order	Total_orders
1	Afternoon	38135

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

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

Ans:

```
SELECT C.customer_state, EXTRACT(Month FROM O.order_purchase_timestamp) Month,
COUNT(O.order_id) Total_orders
FROM `Target.Customers` C JOIN `Target.Orders` O ON C.customer_id=O.customer_id
GROUP BY C.customer_state, Month
ORDER BY Month
```

Output:

Row	customer_state	Month	Total_orders
1	RN	1	51
2	SP	1	3351
3	MG	1	971
4	BA	1	264
5	RJ	1	990
6	RS	1	427
7	MA	1	66
8	CE	1	99
9	PA	1	82
10	PB	1	33

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

Ans:

```
SELECT customer_state, COUNT(customer_id) No_of_customers
FROM `Target.Customers`
GROUP BY customer_state
ORDER BY No_of_customers DESC
```

Output:

Row	customer_state	No_of_customers
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

Results per page: 50 1 - 27 of 27

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

4.1 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.

Ans:

```
WITH Yearly_order as
(SELECT EXTRACT(Year FROM O.order_purchase_timestamp) Order_year, SUM(P.payment_value)
Total_costs
FROM `Target.Payments` P JOIN `Target.Orders` O ON P.order_id=O.order_id
WHERE EXTRACT(Month FROM O.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY Order_year)
SELECT Round(C1.Total_costs,2) Total_cost_2017, Round(C2.Total_costs,2) Total_cost_2018,
ROUND((((C2.Total_costs-C1.Total_costs)/C1.Total_costs)*100,2) Percentage_increase
FROM (SELECT Total_costs FROM Yearly_order WHERE Order_year=2017)C1,(SELECT Total_costs FROM
Yearly_order WHERE Order_year=2018)C2
```

Output:

JOB INFORMATION		RESULTS	CHART	JSON	EX
Row	Total_cost_2017	Total_cost_2018	Percentage_increase		
1	3669022.12	8694733.84	136.98		

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

Ans:

```
SELECT C.customer_state, ROUND(SUM(P.payment_value),2) Total_value,  
ROUND(AVG(P.payment_value),2) Average_value  
FROM `Target.Orders` O JOIN `Target.Payments` P ON O.order_id=P.order_id JOIN `Target.Customers` C  
ON O.customer_id=C.customer_id  
GROUP BY C.customer_state  
ORDER BY Total_value
```

Output:

Row	customer_state	Total_value	Average_value
1	RR	10064.62	218.8
2	AP	16262.8	232.33
3	AC	19680.62	234.29
4	AM	27966.93	181.6
5	RO	60866.2	233.2
6	TO	61485.33	204.27
7	SE	75246.25	208.44
8	AL	96962.06	227.08
9	RN	102718.13	196.78
10	PI	108523.97	207.11

Results per page: 50 1 – 27 of 27

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

Ans:

```
SELECT C.customer_state, ROUND(SUM(OI.freight_value),2) Total_value, ROUND(AVG(OI.freight_value),2)  
Average_value  
FROM `Target.Orders` O JOIN `Target.Order_items` OI ON O.order_id=OI.order_id JOIN `Target.Customers`  
C ON O.customer_id=C.customer_id  
GROUP BY C.customer_state  
ORDER BY Total_value
```

Output:

Row	customer_state	Total_value	Average_value
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

Results per page: 50 1 – 27 of 27

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

5.1 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.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time\_to\_deliver** = order\_delivered\_customer\_date - order\_purchase\_timestamp
- **diff\_estimated\_delivery** = order\_delivered\_customer\_date - order\_estimated\_delivery\_date

Ans:

```
SELECT order_purchase_timestamp, order_delivered_customer_date, order_estimated_delivery_date,
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, Day) No_of_days,
DATE_DIFF(order_delivered_customer_date, order_estimated_delivery_date, Day) Deliverydate_difference
FROM `Target.Orders`
```

Output:

Row	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date	No_of_days	Deliverydate_difference
1	118-02-19 19:40:52 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	12
2	116-10-09 15:39:56 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	-28
3	116-10-03 21:01:41 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	-16
4	117-04-15 15:37:38 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	-1
5	117-04-14 22:21:54 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	117-04-16 14:56:13 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	-1
7	117-04-08 21:20:24 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	4
8	117-04-11 19:49:45 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	4
9	117-04-12 12:17:08 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	1
10	117-04-19 22:52:59 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	5

Results per page: 50 1 - 50 of 99441

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

Ans:

WITH Freight AS

```
(SELECT C.customer_state, DENSE_RANK()OVER(ORDER BY AVG(OI.freight_value)DESC) Desc_rank,
DENSE_RANK()OVER(ORDER BY AVG(OI.freight_value)) Asc_rank
FROM `Target.Customers` C JOIN `Target.Orders` O ON C.customer_id=O.customer_id
JOIN `Target.Order_items` OI ON O.order_id=OI.order_id
GROUP BY C.customer_state)
SELECT FI.customer_state Highest_freightvalue, FO.customer_state Lowest_freightvalue
FROM Freight FI JOIN Freight FO ON FI.Desc_rank=FO.Asc_rank
WHERE FI.Desc_rank<=5
ORDER BY FI.Desc_rank
```

Output:

Row	Highest_freightvalue	Lowest_freightvalue
1	RR	SP
2	PB	PR
3	RO	MG
4	AC	RJ
5	PI	DF

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

Ans:

WITH DEL\_Time AS

(SELECT

C.customer\_state,AVG(DATE\_DIFF(O.order\_delivered\_customer\_date,O.order\_purchase\_timestamp,Day))

Avg\_Del\_Time

FROM `Target.Customers` C JOIN `Target.Orders` O ON C.customer\_id=O.customer\_id

GROUP BY C.customer\_state),

Ste\_Rank AS

(SELECT customer\_state, DEL\_Time.Avg\_Del\_Time,

DENSE\_RANK()OVER(ORDER BY DEL\_Time.Avg\_Del\_Time Desc) Desc\_rank,

DENSE\_RANK()OVER(ORDER BY DEL\_Time.Avg\_Del\_Time) Asc\_rank

FROM DEL\_Time)

SELECT SR.customer\_state Highest\_delivery\_time, SL.customer\_state Lowest\_delivery\_time

FROM Ste\_Rank SR JOIN Ste\_Rank SL ON SR.Desc\_rank=SL.Asc\_rank

WHERE SR.Desc\_rank<=5

ORDER BY SR.Desc\_rank

Output:

Row	Highest_delivery_time	Lowest_delivery_time
1	RR	SP
2	AP	PR
3	AM	MG
4	AL	DF
5	PA	SC

5.4 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.

Ans:

```
WITH Fast_Delv AS
(SELECT C.customer__state,
AVG(DATE_DIFF(O.order__estimated__delivery__date,O.order__delivered__customer__date,Day)) Fast_Del
FROM `Target.Customers` C JOIN `Target.Orders` O ON C.customer__id=O.customer__id
WHERE O.order__delivered__customer__date IS NOT NULL
GROUP BY 1),
Fast_Del_Rank AS
(SELECT customer__state, DENSE_RANK()OVER(ORDER BY Fast_Del Desc) H_Rank
FROM Fast_Delv)
SELECT FDR.customer__state State
FROM Fast_Del_Rank FDR
WHERE H_Rank<=5
ORDER BY H_Rank
```

Output:

Row	State
1	AC
2	RO
3	AP
4	AM
5	RR

## 6. Analysis based on the payments:

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

Ans:

```
WITH CTE AS
(SELECT DISTINCT O.order__id, FORMAT_TIMESTAMP('%Y-%m',order__purchase__timestamp) Order__purchase,
P.payment__type
FROM `Target.Orders` O JOIN `Target.Payments` P ON O.order__id=P.order__id
WHERE P.payment__value>0),
Payment__type__count AS
```



```

(SELECT Order__purchase, payment__type, COUNT(order_id) No_of__orders
FROM CTE
GROUP BY Order__purchase,payment__type)
SELECT payment__type, Order__purchase, No_of__orders,
SUM(No_of__orders)OVER(Partition BY payment__type ORDER BY Order__purchase) Month_on__Month
FROM Payment__type__count
ORDER BY 1,2

```

Output:

Row	payment__type	Order__purchase	No_of__orders	Month_on__Month
1	UPI	2016-10	63	63
2	UPI	2017-01	197	260
3	UPI	2017-02	398	658
4	UPI	2017-03	590	1248
5	UPI	2017-04	496	1744
6	UPI	2017-05	772	2516
7	UPI	2017-06	707	3223
8	UPI	2017-07	845	4068
9	UPI	2017-08	938	5006
10	UPI	2017-09	903	5909

Results per page: 50 1 - 50 of 88

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

Ans:

```

SELECT payment__installments, COUNT(Distinct order_id) No_of__orders
FROM `Target.Payments`
WHERE payment__type= 'credit_card' AND payment__installments>1 AND payment__installments>0
GROUP BY 1
HAVING COUNT(order_id)>=1

```

Output:

Row	payment__installment	No_of__orders
1	2	12389
2	3	10443
3	4	7088
4	5	5234
5	6	3916
6	7	1623
7	8	4253
8	9	644

Load more

