TARGET PROJECT

1.1) Data type of all Columns

SQL Query:

```
SELECT
column_name,
data_type
FROM target.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = "customer"
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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: From the output we can see most of the DATA_TYPE are in STRING .

1.2) The Time Range when the orders were placed

SQL Query:

```
SELECT
MIN(order_purchase_timestamp) AS From_date,
MAX(order_purchase_timestamp) AS To_date
FROM `target.orders`
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

Insights:

- From the output we can conclude that the order were placed between September-2016 and october 2018

1.3). Count the number of Cities and States in our dataset.

SQL Query:

```
SELECT
COUNT(DISTINCT geolocation_city) AS no_of_city,
COUNT(DISTINCT geolocation_state) AS no_of_states
FROM `target.locatIon`
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON

Row	no_of_city	no_of_states
1	8011	27

Insights: Number of state is 27

2) In-depth Exploration

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

SQL Query:

```
SELECT
Sales_year,
tb1.No_of_orders
FROM (
  SELECT
COUNT(*) AS No_of_orders,
EXTRACT(YEAR FROM order_purchase_timestamp)AS Sales_year
FROM `target.orders`
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)) AS tb1
ORDER BY Sales_year
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON

Row	Sales_year	No_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Insights: There is a growing trend from 2016 to 2018

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

SQL Query:

```
SELECT tb2.Sales_month,
tb2.No_of_orders,
RANK() OVER(ORDER BY tb2.No_of_orders DESC) AS Ranked_on_monthly_seasonality

FROM (
SELECT
Sales_month,
tb1.No_of_orders
FROM (
  SELECT
COUNT(*) AS No_of_orders,
EXTRACT(month FROM order_purchase_timestamp)AS Sales_month
FROM `target.orders`
GROUP BY EXTRACT(month FROM order_purchase_timestamp)) AS tb1
ORDER BY tb1.No_of_orders DESC
) AS tb2
ORDER BY Ranked_on_monthly_seasonality
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Sales_month	No_of_orders	Ranked_on_monthly_seasonality
1	8	10843	1
2	5	10573	2
3	7	10318	3
4	3	9893	4
5	6	9412	5
6	4	9343	6
7	2	8508	7
8	1	8069	8
9	11	7544	9
10	12	5674	10
11	10	4959	11
12	9	4305	12

Insights: Highest number of orders placed in AUGUST month.

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

SQL Query:

```
SELECT
Session,
COUNT(*) AS no_of_orders_InEachSession
FROM
(
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 AS Session
FROM `target.orders`) AS Sess_tb
```

```
GROUP BY Session
ORDER BY no_of_orders_InEachSession DESC
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXEC

Row	Session	no_of_orders_InEach
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insights: From the output we can see that during afternoon session the no of orders count is high

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

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

SQL Query:

```
SELECT
sd.customer_state,
sd.sale_year,
sd.sale_month,
TotalSales
FROM
(
SELECT
 DISTINCT customer_state,
 EXTRACT(month from order_purchase_timestamp) AS sale_month,
 EXTRACT(year from order_purchase_timestamp) AS sale_year,
 COUNT(*) AS TotalSales
FROM `target.orders` AS o
INNER JOIN `target.customer` AS c
ON o.customer_id = c.customer_id
GROUP BY customer_state, EXTRACT(month from order_purchase_timestamp), EXTRACT(year
from order_purchase_timestamp)) AS sd
ORDER BY customer_state, sale_year, sale_month
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	sale_year	sale_month	TotalSales
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: Month on month sales data taken

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

SQL Query:

```
SELECT
customer_state,
COUNT (Distinct customer_id) AS No_of_cust
FROM `target.customer`
GROUP BY customer_state
ORDER BY customer_state
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	No_of_cust
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

Insights: BA state is having more number of customers.

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)

SQL Query:

```
SELECT
Sim.YEAR,
ROUND(SUM(payment_value),2) As Total_sales
FROM
(SELECT
EXTRACT (YEAR FROM order_purchase_timestamp) AS YEAR,
payment_value

 FROM `target.payment` AS pay INNER JOIN
   `target.orders` AS ord
   ON pay.order_id=ord.order_id
   WHERE EXTRACT (MONTH FROM order_purchase_timestamp) BETWEEN 01 AND 08) AS Sim

GROUP BY Sim.YEAR
ORDER BY Sim.YEAR
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON

Row	YEAR	Total_sales
1	2017	3669022.12
2	2018	8694733.84

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

SQL Query:

```
SELECT
customer_state,
ROUND(SUM(price),2) AS total_price,
ROUND(AVG(price),2) AS AVG_price
FROM `target.order_item` AS toi
JOIN `target.orders` AS tao
ON toi.order_id = tao.order_id
JOIN `target.customer` AS tc
ON tc.customer_id = tao.customer_id
GROUP BY customer_state
ORDER BY customer_state
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_price	AVG_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

Insights: Total and average data is available from the output

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

```
SELECT
customer_state,
ROUND(SUM(freight_value),2) AS total_freight,
ROUND(AVG(freight_value),2) AS AVG_freight
FROM `target.order_item` AS toi
JOIN `target.orders` AS tao
ON toi.order_id = tao.order_id
JOIN `target.customer` AS tc
ON tc.customer_id = tao.customer_id
GROUP BY customer_state
ORDER BY customer_state
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight	AVG_freight
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

Insights: Total and average data is available from the output

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_estimated_delivery_date - order_delivered_customer_date

SQL 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.orders`
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTIC

Row	order_id	time_to_deliver	diff_estimated_delivery
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

Insight: Actual vs estimated time of deliver is found using above output

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

SQL Query for Highest Average Freight:

```
SELECT
customer_state,
AVG_fr.AVG_freight,
FROM
(SELECT
customer_state,
ROUND(AVG(freight_value),2) AS AVG_freight
FROM `target.order_item` AS toi
JOIN `target.orders` AS tao
ON toi.order_id = tao.order_id
JOIN `target.customer` AS tc
ON tc.customer_id = tao.customer_id
GROUP BY customer_state
ORDER BY customer_state) AS AVG_fr
ORDER BY AVG_fr.AVG_freight DESC
LIMIT 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	customer_state	AVG_freight
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

SQL Query For Lowest Average Freight

```
SELECT
customer_state,
AVG_fr.AVG_freight,
FROM
(SELECT
customer_state,
ROUND(AVG(freight_value),2) AS AVG_freight
FROM `target.order_item` AS toi
JOIN `target.orders` AS tao
ON toi.order_id = tao.order_id
JOIN `target.customer` AS tc
ON tc.customer_id = tao.customer_id
GROUP BY customer_state
ORDER BY customer_state) AS AVG_fr
ORDER BY AVG_fr.AVG_freight ASC
LIMIT 5
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUT

Row	customer_state	AVG_freight
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

Insights: RR state is having highest freight value and SP state is having highest freight value.

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

SQL Query:

```
SELECT
customer_state,
ROUND(avg(Date_diff(order_delivered_customer_date,order_purchase_timestamp,
day)),2)as avg_delivery_time
From `Target.orders` as TOD Join `Target.customers`as TCUST
on TOD.customer_id = TCUST.customer_id
group by customer_state
order by avg_delivery_time desc
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXI

Row	customer_state	avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

SQL Query:

```
SELECT
customer_state,
ROUND(avg(Date_diff(order_delivered_customer_date,order_purchase_timestamp,
day)),2)as avg_delivery_time
From `Target.orders` as TOD Join `Target.customers`as TCUST
on TOD.customer_id = TCUST.customer_id
group by customer_state
order by avg_delivery_time asc
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EX

Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Insights: RR state is having highest delivery time and SP state is having highest delivery time.

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.

SQL Query:

```
SELECT
customer_state,
ROUND(avg(Date_diff(order_estimated_delivery_date,order_delivered_customer_date,
day)),2)as Estimated_actual_delivery_interval
From `Target.orders` as TOD Join `Target.customers`as TCUST
on TOD.customer_id = TCUST.customer_id
group by customer_state
order by Estimated_actual_delivery_interval desc
Limit 5
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECU

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

Insight: AC customer state is the fastest state to deliver products.

6).Analysis based on the payments:

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

SQL Query:

```
SELECT
pay.payment_type,
pay.sale_year,
pay.sale_month,
TotalSales
FROM
(
SELECT
 DISTINCT payment_type,
 EXTRACT(month from order_purchase_timestamp) AS sale_month,
 EXTRACT(year from order_purchase_timestamp) AS sale_year,
 COUNT(*) AS TotalSales
FROM `target.orders` AS o
INNER JOIN `target.payment` AS p
ON o.order_id = p.order_id
GROUP BY payment_type, EXTRACT(month from order_purchase_timestamp), EXTRACT(year
from order_purchase_timestamp)) AS pay
ORDER BY payment_type, sale_year, sale_month
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	payment_type	sale_year	sale_month	TotalSales
1	voucher	2016	10	23
2	voucher	2017	1	61
3	voucher	2017	2	119
4	voucher	2017	3	200
5	voucher	2017	4	202
6	voucher	2017	5	289
7	voucher	2017	6	239
8	voucher	2017	7	364
9	voucher	2017	8	294
10	voucher	2017	9	287

Insights: Month on month order placed data taken

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

SQL Query:

```
SELECT
pay.payment_type,
pay.sale_year,
pay.sale_month,
TotalSales
FROM
(
SELECT
 DISTINCT payment_type,
 EXTRACT(month from order_purchase_timestamp) AS sale_month,
 EXTRACT(year from order_purchase_timestamp) AS sale_year,
 COUNT(*) AS TotalSales
FROM `target.orders` AS o
INNER JOIN `target.payment` AS p
ON o.order_id = p.order_id
GROUP BY payment_type, EXTRACT(month from order_purchase_timestamp), EXTRACT(year
from order_purchase_timestamp)) AS pay
ORDER BY payment_type, sale_year, sale_month
```

O/P:

Query results

JOB INFORMATION | RESULTS | JSON

Row	payment_installment	TotalSales
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328

Insights: Number of orders placed on the basis of the payment installments has been taken.