SQL BUSINESS CASE:

Submitted by:

Shivani Prajapati

Problem Statement:

Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

Question 1. What does 'good' look like?

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.

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
customer_id	STRING	NULLABLE	-	-	-	-	-
customer_unique_id	STRING	NULLABLE	-	-	-	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
customer_city	STRING	NULLABLE	-	-	-	-	-
customer_state	STRING	NULLABLE	-	-	-	-	-

Query: Select * from ‘sqlproject.customers’ limit 10

Output:

Query results

SAVE RESULTS

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

Row	customer_id	customer_unique_id	customer_zip_code_i	customer_city	customer_state
1	0735e7e4298a2ebbb46649346...	fcb003b1bdc0df64b4d065d9b...	59650	acu	RN
2	903b3d86e3990db01619a4ebe...	46824822b15da44e983b021d...	59650	acu	RN
3	38c97666e962d4fea7fd6a83e...	b6108acc674ae5c99e29adc10...	59650	acu	RN
4	77c2f46cf580f4874c9a5751c2...	402cce5c0509000eed9e77fec...	63430	ico	CE
5	4d3ef4cfffb8ad4767c199c36a...	6ba00666ab7eada5ceec279b2...	63430	ico	CE
6	3000841b86e1fbe9493b52324...	796a0b1a21f597704057184a1...	63430	ico	CE
7	3c325415ccc7e622c66dec4bc...	05d1d2d9f0161c5f397ce7fc77...	63430	ico	CE
8	04f3a7b250e3be964f01bf22bc...	c34585a0276ecc5e4fb03de75...	63430	ico	CE
9	894202b8ef01f4719a4691e79...	01a4fe5fc00bbdb0b0a4af5a53...	63430	ico	CE
10	9d715b9fb75a9d081c14126c0...	8f399f3b7ace8e6245422c9e1f...	63430	ico	CE

Insights: Datatype of column 'customer_zip_code_prefix' is INTEGER, and Datatype of columns 'customer_id', 'customer_unique_id', 'customer_city', 'customer_state' is STRING.

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

```
Select extract (year from order_purchase_timestamp) as Year,
Min(order_purchase_timestamp) as Min_order_purchased,
Max(order_purchase_timestamp) as Max_order_purchased,
timestamp_diff(max(order_purchase_timestamp), min(order_purchase_timestamp), day) as days_diff
from `sqlproject.orders`
group by Year
order by Year
```

Output:

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

Row	Year	Min_order_purchased	Max_order_purchased	days_diff
1	2016	2016-09-04 21:15:19 UTC	2016-12-23 23:16:47 UTC	110
2	2017	2017-01-05 11:56:06 UTC	2017-12-31 23:29:31 UTC	360
3	2018	2018-01-01 02:48:41 UTC	2018-10-17 17:30:18 UTC	289

Insights:

- **In 2016, the orders were placed on the difference of 110 days, with first order placed on 4th September and last order placed on 23rd December.**
- **In 2017, the orders were placed on the difference of 360 days, with first order placed on 5th January and last order placed on 31st December.**
- **In 2018, the orders were placed on the difference of 289 days, with first order placed on 1st January and last order placed on 17 October.**

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

```
select count(distinct x.customer_city) as cities, count(distinct x.customer_state) as states from
```

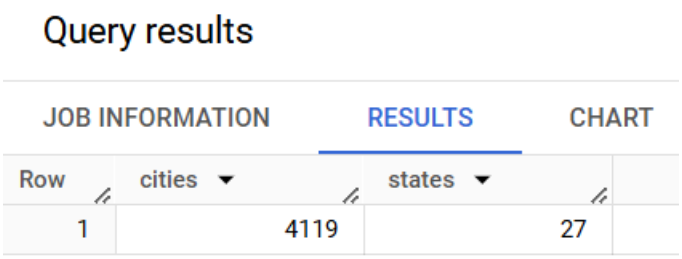
(select *

from `sqlproject.customers`

inner join `sqlproject.orders`

using(customer_id)) x

Output:



Insights: **Customers from 4119 cities and 27 states have placed the order in between 2016 and 2018.**

Question 2. In-depth Exploration:

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

Answer:

select extract(year from order_purchase_timestamp) as Year, count(*) as no_of_orders

from `sqlproject.orders`

group by extract(year from order_purchase_timestamp)

order by Year

output:

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

Insights: As clearly mentioned in the output, 329 orders were placed in 2016, 45101 were placed in 2017 and 54011 were placed in 2018. So, yes there is a growing trend of orders placed.

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

Answer:

```
select extract(month from order_purchase_timestamp) as Months,

format_datetime("%B", order_purchase_timestamp) as month_name,

count(*) as no_of_orders

from `sqlproject.orders`

group by extract(month from order_purchase_timestamp), format_datetime("%B", order_purchase_timestamp)

order by no_of_orders desc
```

Output:

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

Insights: Maximum number of orders were placed in the month of August, followed by May and then July.

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**

Solution:

```
select y.time_, count(*) as no_of_order_placed from
 (select hours_,
 case
    when hours_ <= 6 then "Dawn"
    when hours_ between 7 and 12 then "Mornings"
    when hours_ between 13 and 18 then "Afternoon"
    else "Night"
 end as time_ from
  (
    select extract(hour from order_purchase_timestamp) as hours_
    from `sqlproject.orders`
    order by hours_
  )x
 ) y
group by y.time_
order by no_of_order_placed desc
```

Output:

Row	time_	no_of_order_placed
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights: Maximum orders were placed during Afternoon time.

Question 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(month from o.order_purchase_timestamp) as Month,
format_datetime("%B", o.order_purchase_timestamp) as month_name,
count(o.order_id) as no_of_orders
```

```
from `sqlproject.orders` o
inner join `sqlproject.customers` c
using(customer_id)
group by c.customer_state, Month, month_name
order by c.customer_state, Month
```

Row	customer_state	Month	month_name	no_of_orders
1	AC	1	January	8
2	AC	2	February	6
3	AC	3	March	4
4	AC	4	April	9
5	AC	5	May	10
6	AC	6	June	7
7	AC	7	July	9
8	AC	8	August	7
9	AC	9	September	5
10	AC	10	October	6
11	AC	11	November	5

Insights: Maximum orders were placed in May.

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

```
select customer_state, count(customer_id) as no_of_customers from
`sqlproject.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

Insights: Maximum customers are situated in SP state, followed by RJ state, and then MG state.

Question 4. Impact on Economy: Analyse 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.

```
WITH order_costs AS (
  SELECT
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
    SUM(p.payment_value) AS total_payment_value
  FROM `sqlproject.orders` o
  JOIN `sqlproject.payments` p ON o.order_id = p.order_id
  WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1
AND 8
  GROUP BY order_year, order_month
)

SELECT
  (SUM(CASE WHEN order_year = 2018 THEN total_payment_value ELSE 0 END)
-
   SUM(CASE WHEN order_year = 2017 THEN total_payment_value ELSE 0 END))
/
```

```
    SUM(CASE WHEN order_year = 2017 THEN total_payment_value ELSE 0 END)
* 100 AS percentage_increase

FROM order_costs;
```

Output:

Row	percentage_increase
1	136.9768716466...

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

```
select c.customer_state, round(sum(p.payment_value), 2) as total_price,
round(avg(p.payment_value), 2) as average_value

from `sqlproject.customers` c

inner join `sqlproject.orders` o

using(customer_id)


inner join `sqlproject.payments` p

using(order_id)

group by c.customer_state

order by total_price desc
```

output:

Row	customer_state	total_price	average_value
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71

Insights: Total & Average value of order price for each state is given in the output.

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

```
select c.customer_state, round(sum(ot.freight_value), 2) as total_freight_value,
round(avg(ot.freight_value), 2) as average_freight_value

from `sqlproject.customers` c

inner join `sqlproject.orders` o

using(customer_id)


inner join `sqlproject.order_items` ot

using(order_id)

group by c.customer_state

order by total_freight_value desc
```

Output:

Row	customer_state	total_freight_value	average_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

Insights: The Total & Average value of order freight for each state is given in the table above.

Question 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.

   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
   a. **time_to_deliver = order_delivered_customer_date - order_purchase_timestamp**
   b. **diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date**

```
select  order_id,

date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_deliver,

date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as diff_estimated_delivery

from `sqlproject.orders`
```

Output:

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

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

```
(SELECT c.customer_state, round(AVG(oi.freight_value), 2) AS avg_freight
FROM `sqlproject.customers` c
JOIN
  `sqlproject.orders` o
```

```
 using(customer_id)

JOIN
`sqlproject.order_items` oi
using(order_id)
   group by c.customer_state
   order by avg_freight desc limit 5)

union all
(SELECT c.customer_state, round(AVG(oi.freight_value), 3) AS lowest_avg_freight
FROM `sqlproject.customers` c
JOIN
  `sqlproject.orders` o
 using(customer_id)

JOIN
`sqlproject.order_items` oi
using(order_id)
   group by c.customer_state
   order by lowest_avg_freight asc limit 5)
```

Output:

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
6	SP	15.147
7	PR	20.532
8	MG	20.63
9	RJ	20.961
10	DF	21.041

Insights: States with highest freight value are given in the first 5 rows of the output: Those are- RR > PB > RO > AC > PI.

States with the lowest freight value are given in the last 5 rows of the output. Those are- PI < SP < PR < MG < RJ < DF.

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

Highest delivery_time (in Days):

```
select c.customer_state,
round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, day)), 2) as highest_delivery_time
from `sqlproject.customers` c
join `sqlproject.orders` o
using(customer_id)
group by c.customer_state
order by highest_delivery_time desc
limit 5
```

Output:

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

Lowest delivery time (in Days):

```
select c.customer_state,
round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, day)), 2) as lowest_delivery_time
from `sqlproject.customers` c
join `sqlproject.orders` o
using(customer_id)
group by c.customer_state
order by lowest_delivery_time asc
limit 5
```

Output:

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

Insights: Highest delivery time is taken in the state RR, and lowest is taken in the state SP.

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.

```
select customer_state, round(estimated_days-delivered_days, 2) as
fastest_days_taken from
(select c.customer_state,
round(avg(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, day)), 2) as delivered_days,
round(avg(date_diff(o.order_estimated_delivery_date,
o.order_purchase_timestamp, day)), 2) as estimated_days
from `sqlproject.customers` c
inner join `sqlproject.orders` o
using(customer_id)
where o.order_purchase_timestamp is Not NULL
and o.order_delivered_customer_date is Not NULL
and o.order_estimated_delivery_date is Not NULL
group by c.customer_state ) x
order by estimated_days-delivered_days desc limit 5
```

Output:

Row	customer_state	fastest_days_taken
1	AC	20.08
2	RO	19.48
3	AP	19.14
4	AM	18.93
5	RR	16.65

Insights: Order delivery is really fast in the state AC. After that, state RO> AP> AM> RR.

Question 6: Analysis based on the payments:

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

```
select extract(month from o.order_purchase_timestamp) as month,
format_datetime("%B", o.order_purchase_timestamp) as month_name,
p.payment_type, count(o.order_id) as orders_placed

from `sqlproject.orders` o

inner join `sqlproject.payments` p

using(order_id)

group by month, month_name, p.payment_type

order by month
```

Output:

Row	month	month_name	payment_type	orders_placed
1	1	January	credit_card	6103
2	1	January	UPI	1715
3	1	January	voucher	477
4	1	January	debit_card	118
5	2	February	UPI	1723
6	2	February	credit_card	6609
7	2	February	voucher	424
8	2	February	debit_card	82
9	3	March	credit_card	7707
10	3	March	UPI	1942

Insights: month on month no. of orders placed using different payment types are given in detail in the output above. For eg., credit card is most used in the month of January, February and march etc. and debit card is the least used payment method in every month.

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

```
select payment_installments, count(order_id) as no_of_orders
from `sqlproject.payments`
group by payment_installments
```

Row	payment_installment	no_of_orders
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