

# SQL Target Business Case Study

Name: B.Rekha Raghu

## Answer Sheet

1.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.

Answer:

SELECT

column\_name,  
data\_type

FROM

my-firtst-project-444304. Target.INFORMATION\_SCHEMA.COLUMNS

WHERE

table\_name='customers';

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

=

## 2. Get the time RANGE BETWEEN which the orders were placed.

**Answer:**

SELECT

MIN(order\_purchase\_timestamp) AS min\_order\_purchase\_timestamp,

MAX(order\_purchase\_timestamp) AS max\_order\_purchase\_timestamp

FROM

Target.orders;

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

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

**Answer:**

SELECT

COUNT(DISTINCT customer\_city) AS city\_count,

COUNT(DISTINCT customer\_state) AS state\_count

FROM

`Target.customers`;

JOB INFORMATION		RESULTS	CHART
Row	city_count	state_count	
1	4119	27	

## 2. In-depth Exploration:

1. Is there a growing trend IN the no. OF orders placed OVER the past years?

Answer: Yes, we can see the growing trend over the past years.

```
WITH cte AS (  
  SELECT  
    DISTINCT EXTRACT(year FROM order_purchase_timestamp) AS purchase_year,  
    COUNT(order_id) AS no_of_orders  
  FROM `Target.orders`  
  GROUP BY 1  
  order by 1 asc),  
previous_data AS (  
  SELECT *,  
    LAG(no_of_orders,1) OVER(ORDER BY purchase_year) AS prev  
  FROM cte)  
SELECT purchase_year,  
  no_of_orders,  
  prev,  
  ((no_of_orders-prev)/prev) AS trend  
FROM previous_data  
order by 1;
```

Row	purchase_year	no_of_orders	prev	trend
1	2016	329	null	null
2	2017	45101	329	136.0851063829...
3	2018	54011	45101	0.197556595197...

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

### Answer:

with monthlyorders as

```
(select  extract(month from order_purchase_timestamp) as orders_month,
        count(order_id) as order_count
```

```
from `Target.orders`
```

```
group by 1
```

```
order by 1),
```

prevorders as

```
(select *,
        lag(order_count,1) over( order by orders_month) as prev_order_count
from monthlyorders)
```

```
select *,
```

```
((order_count - prev_order_count)/prev_order_count) as trend
```

```
from prevorders
```

```
order by orders_month
```

### Query results

[SAVE RESULTS](#) ▼

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	orders_month ▼	order_count ▼	prev_order_count ▼	trend ▼		
1	1	8069	null	null		
2	2	8508	8069	0.054405750402...		
3	3	9893	8508	0.162787964268...		
4	4	9343	9893	-0.05559486505...		
5	5	10573	9343	0.131649363159...		
6	6	9412	10573	-0.10980800151...		
7	7	10318	9412	0.096260093497...		
8	8	10843	10318	0.050881953867...		
9	9	4305	10843	-0.60296965784...		
10	10	4959	4305	0.151916376306...		

10 rows

**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

**Answer: The Brazilian customers mostly placed their orders during Afternoon time.**

```
select * 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 "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 orderstime,
            count(distinct order_id) as order_count
from `Target.orders`
group by 1
order by 2 desc
limit 1)
```

Query results

 SAVE RESULTS ▾

 OPEN IN ▾



JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	orderstime ▾	order_count ▾				
1	Afternoon	38135				

Results per page:

50 ▾

1 - 1 of 1

Activate Windows



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

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

**Answer:**

```
select c.customer_state,
       extract (month from o.order_purchase_timestamp) as order_months,
       count(distinct o.order_id) as order_count
from `Target.orders` o
join `Target.customers` c
on o.customer_id=c.customer_id
group by 1,2
order by 2,1
```

Query results SAVE RESULTS OPEN IN

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state	order_months	order_count			
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			

Results per page: 50 1 - 50 of 322

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

**Answer:**

```
select customer_state,count(distinct customer_id) customer_count from  
`Target.customers` group by 1 order by 1,2;
```

Row	customer_state	customer_count
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
13	MT	907
14	PA	975
15	PB	536
16	PE	1652

#### 4.Impact on Economy: Analyze 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).

#### Answer:

```
with cte as
(select extract(year from o.order_purchase_timestamp) as year1,
sum(p.payment_value) as cost
from `Target.orders` o
join `Target.payments` p
on o.order_id = p.order_id
WHERE EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 AND EXTRACT(YEAR
FROM o.order_purchase_timestamp) IN (2017,2018)
group by 1
order by 1),
payment as
(select *,lag(cost,1) over(order by year1) as prev_cost
from cte)
select *,
round(ifnull(((cost-prev_cost)/prev_cost)*100,0),2) as percentage_increase_in_cost
from payment order by year1
```

Query results					
JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS
Row	year1	cost	prev_cost	percentage_increase	
1	2017	3669022.119999...	null	0.0	
2	2018	8694733.839999...	3669022.119999...	136.98	



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

### Answer:

```
select c.customer_state,
       sum(i.price) as Total_price,
       (sum(i.price)/count(distinct i.order_id)) as Average_order_price
from `Target.order_items` i
join `Target.orders` o
on o.order_id=i.order_id
join Target.customers c
on c.customer_id=o.customer_id
group by 1
order by 1
```

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTI
Row	customer_state	Total_price	Average_order_price		
1	AC	15982.94999999...	197.3203703703...		
2	AL	80314.80999999...	195.4131630170...		
3	AM	22356.84000000...	152.0873469387...		
4	AP	13474.29999999...	198.1514705882...		
5	BA	511349.9900000...	152.2781387730...		
6	CE	227254.7099999...	171.2544913338...		
7	DF	302603.9399999...	142.4018541176...		
8	ES	275037.3099999...	135.8208938271...		
9	GO	294591.9499999...	146.7822371699...		
10	MA	119648.2199999...	161.6867837837...		
11	MG	1585308.029999...	137.3274454261...		
12	MS	116812.6399999...	164.7568970380...		
13	MT	156453.5299999...	173.2597231450...		
14	PA	178947.8099999...	184.4822783505...		
15	PB	115268.0799999...	216.6693233082...		
16	PE	262788.0299999...	159.4587560679...		
17	PI	86914.07999999...	176.2963083164...		
18	PR	683083.7600000...	136.6714205682...		
19	RJ	1824092.669999...	142.9315679360...		
20	RN	83034.97999999...	172.2717427385...		
21	RO	46140.64000000...	186.8042105263...		

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

Answer:

```
select c.customer_state,
       sum(i.freight_value) as Total_freight_price,
       (sum(i.freight_value)/count(distinct i.order_id)) as
Average_freight_order_price
from `Target.order_items` i
join `Target.orders` o
on o.order_id=i.order_id
join Target.customers c
on c.customer_id=o.customer_id
group by 1
order by 1;
```

Row	customer_state	Total_freight_price	Average_freight_order_price
1	AC	3686.750000000...	45.51543209876...
2	AL	15914.589999999...	38.72163017031...
3	AM	5478.890000000...	37.27136054421...
4	AP	2788.500000000...	41.00735294117...
5	BA	100156.67999999...	29.82628945801...
6	CE	48351.589999999...	36.43676714393...
7	DF	50625.499999999...	23.82376470588...
8	ES	49764.599999999...	24.57511111111...
9	GO	53114.979999999...	26.46486297957...
10	MA	31523.770000000...	42.59968918918...
11	MG	270853.4600000...	23.46270443520...
12	MS	19144.030000000...	27.00145275035...

## 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.

Answer:

```
select date_diff(order_delivered_customer_date,order_purchase_timestamp,day)
as Delivery_Time,
date_diff(order_delivered_customer_date,order_estimated_delivery_date,day) as
Estimated_Delivery_Time
from `Target.orders`;
```

Row	Delivery_Time	Estimated_Delivery_Time
1	30	12
2	30	-28
3	35	-16
4	30	-1
5	32	0
6	29	-1
7	43	4
8	40	4
9	37	1
10	33	5
11	38	6
12	36	2

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

**Answer:**

```
with cte as
(select c.customer_state,
       round((sum(i.freight_value)/count(distinct i.order_id)),2) as
Average_freight_order_price
  from `Target.order_items` i
 join `Target.orders` o
  on o.order_id=i.order_id
 join Target.customers c
  on c.customer_id=o.customer_id
 group by 1
 order by 1),
rank1 as
(select *,
       dense_rank() over( order by Average_freight_order_price desc) as
highest_drnk,
       dense_rank() over( order by Average_freight_order_price asc) as lowest_drnk
  from cte)
select customer_state, Average_freight_order_price
  from rank1 where lowest_drnk <=5
 union all
select customer_state, Average_freight_order_price
  from rank1 where highest_drnk <=5
 order by Average_freight_order_price desc;
```

Query results

SAVE RESULTS

OPEN IN

JOB INFORMATION

RESULTS

CHART

JSON

EXECUTION DETAILS

EXECUTION GRAPH

Row	customer_state	Average_freight_order
1	RR	48.59
2	PB	48.35
3	RO	46.22
4	AC	45.52
5	PI	43.04
6	RJ	23.95
7	DF	23.82
8	PR	23.58
9	MG	23.46
10	SP	17.37

Results per page: 50

1 - 10 of 10

Activate Windows

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

**Answer:**

```

with AverageDelivertime as
(SELECT
c.customer_state,ROUND(AVG(DATE_DIFF(DATE(o.order_delivered_customer_date),
DATE(o.order_purchase_timestamp),Day)),2) AS Average_delivery_time
from `Target.orders` o
join `Target.customers` c
on c.customer_id=o.customer_id
group by 1
order by 2),
rank1 as
(select customer_state,Average_delivery_time,
dense_rank() over(order by Average_delivery_time desc) as
Highest_Average_delivery_time,
dense_rank() over(order by Average_delivery_time asc) as
Lowest_Average_delivery_time
from AverageDelivertime)
select customer_state,Average_delivery_time
from rank1
where Highest_Average_delivery_time <=5 or
Lowest_Average_delivery_time <=5
order by Average_delivery_time desc;

```

## Query results

JOB INFORMATION		RESULTS	CHART	JSON
Row	customer_state	Average_delivery_time		
1	RR	29.34		
2	AP	27.18		
3	AM	26.36		
4	AL	24.5		
5	PA	23.73		
6	SC	14.91		
7	DF	12.9		
8	MG	11.95		
9	PR	11.94		
10	SP	8.7		

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

**Answer:**

```
SELECT c.customer_state,
ROUND(AVG(DATE_DIFF(DATE(o.order_estimated_delivery_date),
DATE(o.order_delivered_customer_date),Day)),2) AS avg_delivery_days
from `Target.orders` o
join `Target.customers` c
on c.customer_id=o.customer_id
group by 1
order by 2
LIMIT 5;
```

## Query results

JOB INFORMATION		RESULTS	CHART	JSON	EXECUTION DETAILS	EXECUTION GRAPH
Row	customer_state ▼	avg_delivery_days ▼				
1	AL	8.71				
2	MA	9.57				
3	SE	10.02				
4	ES	10.5				
5	BA	10.79				

## 6. Analysis based on the payments:

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

### Answer:

```
select p.payment_type,
       extract (month from o.order_purchase_timestamp) as monthlyorders,
       count(distinct o.order_id) as order_count
from `Target.orders` o
join `Target.payments` p
on o.order_id=p.order_id
group by 1,2
order by 2,1;
```

JOB INFORMATION		RESULTS	CHART	JSON	EXECUT
Row	payment_type ▼	monthlyorders ▼	order_count ▼		
1	UPI	1	1715		
2	credit_card	1	6093		
3	debit_card	1	118		
4	voucher	1	337		
5	UPI	2	1723		
6	credit_card	2	6582		
7	debit_card	2	82		
8	voucher	2	288		
9	UPI	3	1942		
10	credit_card	3	7682		
11	debit_card	3	109		
12	voucher	3	395		

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

**Answer:**

**SELECT**

```

    payment_installments AS installments,
COUNT(order_id) AS num_orders,
FROM `Target.payments`
WHERE payment_installments >= 1
GROUP BY payment_installments
ORDER BY num_orders DESC;
```



Row	installments	num_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133
12	15	74