

SQL BUSSINESS CASE STUDY

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

1.Q.Data type of all columns in the "customers" table.

SELECT

column_name,

data_type

FROM

`poetic-standard-396616.SQL_BUSSINESS_CASE.INFORMATION_SCHEMA.COLUMNS`

where table_name = 'customers';

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAI
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: We observe that ,customer table consists of String and int data types.

Recommendation: customer table consists of String and int data types.

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

```
select min(order_purchase_timestamp) from_date ,max(order_purchase_timestamp)to_date
from `SQL_BUSSINESS_CASE.orders`;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAIL
Row	from_date	to_date		
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC		

Insights: The time range between which the orders were placed between 2016-09-04 21:15:19 UTC and 2018-10-17 17:30:18 UTC

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

```
select customer_city,customer_state,count(*) total_count from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o
on c.customer_id=o.customer_id
group by customer_city,customer_state
order by customer_city,customer_state;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	CHART	PREVIEW
Row	customer_city	customer_state	total_count			
1	abadia dos dourados	MG	3			
2	abadiania	GO	1			
3	abaete	MG	12			
4	abaetetuba	PA	11			
5	abaiara	CE	2			
6	abaira	BA	2			
7	abare	BA	2			
8	abatia	PR	3			
9	abdon batista	SC	1			
10	abelardo luz	SC	6			
11	abrantos	BA	2			

Insights: Observed that, the count of orders are not consistent, they were fluctuating over the months and years.

Assumptions: Thinking that, there may be customers, who haven't placed any orders, so, I filter them by using 2 tables (customers and orders)

2. In-depth Exploration:

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

```
select extract(year from order_purchase_timestamp) Year ,count(*)No_Of_Orders
from `SQL_BUSSINESS_CASE.orders`
group by Year
order by Year;
```

Query results

JOB INFORMATION		RESULTS		JSON
Row	Year		No_Of_Orders	
1	2016	329		
2	2017	45101		
3	2018	54011		

Insights: Observed that, no. of orders are increasing year by year. So, we can say that the no of orders are growing trend over years.

The increasing percentage in no. of orders over years are also not constant, we can see there is a drastic change in them

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

```
select extract(year from order_purchase_timestamp) Year,extract(Month from
order_purchase_timestamp) Month,count(*)
from `SQL_BUSSINESS_CASE.orders`
group by Year,Month
order by Year,Month;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION
Row	Year	Month	No_Of_Orders	
1	2016	9	4	
2	2016	10	324	
3	2016	12	1	
4	2017	1	800	
5	2017	2	1780	
6	2017	3	2682	
7	2017	4	2404	
8	2017	5	3700	
9	2017	6	3245	
10	2017	7	4026	
11	2017	8	4001	

Insights: The seasonality of the months is gradually maintained the consistency in the year 2017 and in 2018 but there is a huge drop on orders at the end of 2018.

3.Q 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

with cte as(

select *,case

when extract(hour from order_purchase_timestamp) >= 0 and extract(hour from order_purchase_timestamp)<7 then "Dawn"

when extract(hour from order_purchase_timestamp)>=7 and extract(hour from order_purchase_timestamp)<13 then "Mornings"

when extract(hour from order_purchase_timestamp)>= 13 and extract(hour from order_purchase_timestamp)<19 then "Afternoon"

when extract(hour from order_purchase_timestamp)>=19 and extract(hour from

```
order_purchase_timestamp)<=23 then "Night"
```

```
end as temp
```

```
from `SQL_BUSSINESS_CASE.orders`)
```

```
select temp time_of_the_day,count(*) no_of_orders from cte
```

```
group by temp
```

Query results

JOB INFORMATION		RESULTS	JSON	EXEC
Row	time_of_the_day	no_of_orders		
1	Mornings	27733		
2	Dawn	5242		
3	Afternoon	38135		
4	Night	28331		

Insights: The count of orders is increased at Afternoon time of the day. The smaller number of orders are placed at Dawn.

Recommendation: To introduce new order items, I think it is better to introduce the products at Afternoon time of the day

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

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

with cte as

```
(select c.customer_state,extract(year from order_purchase_timestamp) order_year,  
extract(month from order_purchase_timestamp) order_month, count(*) no_of_orders
```

```
from `SQL_BUSSINESS_CASE.orders` o join `SQL_BUSSINESS_CASE.customers` c on  
o.customer_id=c.customer_id
```

```
group by order_year,order_month,customer_state
```

order by order_year,order_month)

```
select customer_state,order_year,  
order_month, no_of_orders-(lag(no_of_orders) over(order by order_year,order_month))  
from cte  
order by order_year,order_month,customer_state
```

Query results					
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	CHART PREVIEW
Row	customer_state	order_year	order_month	f0_	
1	RR	2016	9	null	
2	RS	2016	9	0	
3	SP	2016	9	1	
4	AL	2016	10	-9	
5	BA	2016	10	2	
6	CE	2016	10	-32	
7	DF	2016	10	2	
8	ES	2016	10	-3	
9	GO	2016	10	6	
10	MA	2016	10	0	

Insights: State to State the count orders placed for a month are completely varying. There is no consistency in the rise or fall of orders that are placed by different state people

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

```
select customer_state,count(customer_id)No_of_customers_per_state from  
`SQL_BUSSINESS_CASE.customers`
```

group by customer_state

order by No_of_customers_per_state

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	customer_state	No_of_customers_per_state		
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		
11	PE	1652		

desc,customer_state

Insights: we observed that highest no. of customers are from State (SP)

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

1.Q 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 cte as(

select extract(year from order_purchase_timestamp) Year,sum(p.payment_value) total

from `SQL_BUSSINESS_CASE.payments` p join `SQL_BUSSINESS_CASE.orders` o on p.order_id = o.order_id

where extract(year from order_purchase_timestamp) in (2017,2018) and

extract(month from order_purchase_timestamp)>0 and extract(month from order_purchase_timestamp)<9

group by Year)

select distinct ((select total from cte where year=2018)-(select total from cte where

```
year=2017))*100/(select total from cte where year=2018)
percent_increase_from_2017_to_2018 from cte
```

Query results

JOB INFORMATION		RESULTS	JSON
Row		percent_increase_from_2017_to_2018	
1		57.80178913446683	

Insights: Observed that,there is about 57% increase in the cost of orders from year 2017 to 2018 between jan to Aug only

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

```
select c.customer_state,round(sum(oi.price),2) total,round(avg(oi.price),2) average from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id
```

```
join `SQL_BUSSINESS_CASE.order_items` oi on o.order_id=oi.order_id
```

```
group by c.customer_state
```

```
order by c.customer_state
```


Query results				
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	customer_state ▼	total ▼	average ▼	
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: the highest total price of orders is from the state with code SP

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

```
select c.customer_state,round(sum(oi.freight_value),2) total,round(avg(oi.freight_value),2)
average from `SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id
```

```
join `SQL_BUSSINESS_CASE.order_items` oi on o.order_id=oi.order_id
```

```
group by c.customer_state
```

```
order by c.customer_state
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	customer_state	total	average	
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	
11	MC	270252.46	20.62	

Insights: the highest freight value is from the state with code SP

5. Analysis based on sales, freight and delivery time

1 Q 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:

$\text{time_to_deliver} = \text{order_delivered_customer_date} - \text{order_purchase_timestamp}$

$\text{diff_estimated_delivery} = \text{order_estimated_delivery_date} - \text{order_delivered_customer_date}$

```
select order_id, date_diff(order_delivered_customer_date, order_purchase_timestamp, day)
time_to_deliver, date_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) diff_estimated_delivery
from `SQL_BUSSINESS_CASE.orders`;
```

</

Insights: some of the orders are delivering on the expected date, some of the orders are taking more time than expected and some taking less time than expected. The lag between the delivery is in a small number of days only.

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

with cte1 as(

```
select row_number() over(order by avg(oi.freight_value) desc)
high,c.customer_state,round(avg(oi.freight_value),2) average from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id join `SQL_BUSSINESS_CASE.order_items` oi on
o.order_id=oi.order_id
```

group by c.customer_state

order by high)

,cte2 as(

```
select row_number() over(order by avg(oi.freight_value) asc)
low,c.customer_state,round(avg(oi.freight_value),2) average from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id
```

```
join `SQL_BUSSINESS_CASE.order_items` oi on o.order_id=oi.order_id
```

group by c.customer_state

order by low)

select cte1.customer_state highest,cte2.customer_state lowest

from cte1 join cte2

on cte1.high = cte2.low

where cte1.high <=5 and

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAIL
Row	highest	lowest		
1	RR	SP		
2	PB	PR		
3	RO	MG		
4	AC	RJ		
5	PI	DF		

cte2.low<=5

Insights: The first 5 states are highest avg freight value and last 5 states are lowest freight value.

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

WITH cte AS (

with cte1 as(

select row_number() over(order by
avg(o.order_delivered_customer_date-o.order_purchase_timestamp) desc)
high,c.customer_state,round(avg(oi.freight_value),2) average from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id

join `SQL_BUSSINESS_CASE.order_items` oi on o.order_id=oi.order_id

group by c.customer_state

order by high)

```
,cte2 as(
select row_number() over(order by
avg(o.order_delivered_customer_date-o.order_purchase_timestamp) asc)
low,c.customer_state,round(avg(oi.freight_value),2) average from
`SQL_BUSSINESS_CASE.customers` c join `SQL_BUSSINESS_CASE.orders` o on
c.customer_id=o.customer_id

join `SQL_BUSSINESS_CASE.order_items` oi on o.order_id=oi.order_id

group by c.customer_state
order by low)
```

```
select cte1.customer_state highest,cte2.customer_state lowest
from cte1 join cte2
on cte1.high = cte2.low
where cte1.high <=5 and
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DET
Row	highest		lowest	
1	RR		SP	
2	AP		PR	
3	AM		MG	
4	AL		DF	
5	PA		SC	

```
cte2.low<=5
```

Insights: The first 5 states are highest avg delivery time and last 5 states are lowest freight value.

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.

with cte as

```
(select c.customer_state, (avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp,day))-avg(date_diff(order_estimated_delivery_date
,order_delivered_customer_date,day))) average from `SQL_BUSSINESS_CASE.orders` o join
`SQL_BUSSINESS_CASE.customers` c on o.customer_id=c.customer_id

group by c.customer_state)
```

```
select customer_state from
```

```
(select customer_state,dense_rank() over(order by cte.average desc) highest from cte t where
highest <= 5
```

```
order by highest
```

Query results

JOB INFORMATION		RESULTS
Row	customer_state	
1	AL	
2	RR	
3	MA	
4	SE	
5	CE	

Insights: Observed that this top 5 states above are the states where the order delivery is really fast as compared to the estimated date of delivery.

6..Analysis based on the payments:

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

with cte as

```
(select p.payment_type,extract(year from order_purchase_timestamp)
order_year,extract(month from order_purchase_timestamp) order_month, count(*)
no_of_orders
```

```
from `SQL_BUSSINESS_CASE.orders` o join `SQL_BUSSINESS_CASE.payments` p on o.order_id =
p.order_id
```

```
group by order_year,order_month,payment_type
```

order by order_year,order_month)

select payment_type,order_year,order_month, no_of_orders-(lead(no_of_orders) over(order by order_year,order_month)) month_on_month from cte

order by order_year,order_month

Query results					
JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS	CHART PREVIEW
Row	payment_type	order_year	order_month	month_on_month	
1	credit_card	2016	9	-251	
2	credit_card	2016	10	191	
3	voucher	2016	10	21	
4	debit_card	2016	10	1	
5	UPI	2016	10	40	
6	credit_card	2016	12	-582	
7	credit_card	2017	1	386	
8	UPI	2017	1	136	
9	debit_card	2017	1	-1347	
10	voucher	2017	1	52	

Insights: payment_type to payment_type the count orders placed for a month are completely varying. There is no consistency in the rise or fall of orders that are placed by different payment type

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

select payment_installments,count(distinct order_id)no_of_orders from
`SQL_BUSSINESS_CASE.payments` where payment_installments > 0

group by payment_installments;

Query results			
JOB INFORMATION		RESULTS	JSON
Row	payment_installment	no_of_orders	
1	1	49060	
2	2	12389	
3	3	10443	
4	4	7088	
5	5	5234	
6	6	3916	
7	7	1623	
8	8	4253	
9	9	644	
10	10	5315	
11	11	22	

Insights: These are the total no. of orders placed on the basis of the payment installments that have been paid.(i.e.>0)

