Target Business Case Study Using SQL

25 February 2025 19:03

Basic data exploration

1) Data type of all Columns in Order Item Table

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

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

2) The Time range between the orders are placed

```
SELECT
MIN(order_purchase_timestamp) AS Order_plcaed_from,
MAX(order_purchase_timestamp) AS Order_plcaed_to
FROM target.orders
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

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

3) Total Number of Unique customers

```sql
SELECT COUNT(DISTINCT customer_unique_id) AS Total_unique_customer
FROM `target.customer`
LIMIT 10
```

Query results

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

Row	Total_unique_custor
1	96096

4) Total number of products and different product category the inventory holds

```sql
SELECT
COUNT(product_id) AS Total_no_of_products,
COUNT(DISTINCT product_category) AS Total_Product_category
FROM target.products
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Total_no_of_product	Total_Product_categ
1	32951	73

5) Finding out total number of cities and states in our dataset

```sql
SELECT
COUNT(DISTINCT geolocation_city) AS No_of_city,
COUNT(DISTINCT geolocation_state) AS No_of_states
FROM target.geoloc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	No_of_city	No_of_states
1	8011	27

Exploratory Data Analysis

1) Finding growth trend in Number of orders placed over the years

```sql
SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
COUNT(order_id) AS No_of_orders
FROM `target.orders`
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)
ORDER BY Year
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

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

Insight: From the above Trend we can see that there is Significant boost in Orders Placed between 2016 and 2017

2) Is there a monthly Seasonality on number of Orders being place

```sql
SELECT *,
DENSE_RANK() OVER(ORDER BY No_of_orders DESC) AS Month_Ranking
FROM(
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS Mon,
COUNT(order_id) AS No_of_orders
FROM `target.orders`
GROUP BY EXTRACT(MONTH FROM order_purchase_timestamp)
)AS t1
ORDER BY Month_Ranking
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Mon	No_of_orders	Month_Ranking
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

Insight: From the above Output Maximum Average orders are placed Between May through August

3) During what time of the day customer place most of their Order. Considering below time frames

0-6 hrs : Dawn
7-12 hrs : Mornings
13-18 hrs : Afternoon
19-23 hrs : Night

```sql
SELECT
Timing,
COUNT(order_id) AS Total_Order
FROM
(
```

```sql
SELECT
Timing,
COUNT(order_id) AS Total_Order
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 Timing
FROM `target.orders`) AS t1
GROUP BY timing
ORDER BY Total_Order DESC
```

Query results

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

Row	Timing	Total_Order
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insight: The output give clear picture that most of the Orders are Placed during Mid of the day

4) Month on Month number of orders Placed over year

```sql
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
COUNT(order_id) AS total_orders,
FROM
`target.orders` AS ord JOIN
`target.customer` AS cus ON ord.customer_id = cus.customer_id
GROUP BY EXTRACT(MONTH FROM order_purchase_timestamp) ,EXTRACT(YEAR FROM order_purchase_timestamp)
ORDER BY order_month, order_year
```

Query results

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

Row	order_month	order_year	total_orders
1	1	2017	800
2	1	2018	7269
3	2	2017	1780
4	2	2018	6728
5	3	2017	2682
6	3	2018	7211
7	4	2017	2404
8	4	2018	6939
9	5	2017	3700
10	5	2018	6873

Insight: Comparing to previous Year Sales we can see there is Definitely a growth trend in Sales on Each month

5) How are the customer distributed across the state

```sql
SELECT
customer_state,
COUNT(order_id) AS Max_Order
FROM `target.orders` AS ord
JOIN `target.customer` AS cus
ON ord.customer_id = cus.customer_id
GROUP BY customer_state
ORDER BY Max_Order DESC
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	Max_Order
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

6) Analysing % of increase in cost of order for 2018 comparing to 2017

```sql
SELECT
CONCAT(ROUND(((Y_2018 - Y_2017)/Y_2017)*100,2) ,"% Increase Over 2017 & 2018") AS Increase_Percent
FROM(
SELECT
  MAX(CASE WHEN ord_year = 2017 THEN ROUND(Total_pay_value,0) END) AS Y_2017,
  MAX(CASE WHEN ord_year = 2018 THEN ROUND(Total_pay_value,0) END) AS Y_2018
FROM
(
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) AS ord_year,
  SUM(payment_value) AS Total_pay_value
FROM `target.orders` AS ord
JOIN `target.payments` AS pay
ON ord.order_id = pay.order_id
GROUP BY EXTRACT(YEAR FROM order_purchase_timestamp)) AS t1)AS t2
```

Query results

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

Row	Increase_Percent
1	20% Increase Over 2017 & 2018

7) Calculating total & average value of order cost amount of each state

```sql
SELECT
cus.customer_state,
ROUND(SUM(payment_value),2) AS Total_order_price,
ROUND(AVG(payment_value),0) AS Average_order_price

FROM `target.orders` AS ord
JOIN `target.customer`AS cus
ON ord.customer_id = cus.customer_id
JOIN `target.payments` AS pay
ON pay.order_id = ord.order_id
GROUP BY cus.customer_state
ORDER BY Total_order_price DESC
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	Total_order_price	Average_order_price
1	SP	5998226.96	138.0
2	RJ	2144379.69	159.0
3	MG	1872257.26	155.0
4	RS	890898.54	157.0
5	PR	811156.38	154.0
6	SC	623086.43	166.0
7	BA	616645.82	171.0
8	DF	355141.08	161.0
9	GO	350092.31	166.0

8) Calculating total & average value of order price of each state

```sql
SELECT
cus.customer_state,
ROUND(SUM(price),2) AS total_price,
ROUND(AVG(PRICE),2) AS avg_price
FROM `target.order_item` AS ordit
JOIN `target.orders` AS ord
ON ordit.order_id = ord.order_id
JOIN `target.customer` AS cus
ON ord.customer_id = cus.customer_id
GROUP BY cus.customer_state
ORDER BY total_price DESC
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	total_price	avg_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27

9) Calculating total & average value of freight amount of each state

```sql
SELECT
cus.customer_state,
ROUND(SUM(freight_value),2) AS total_freight,
ROUND(AVG(freight_value),2) AS avg_freight
FROM `target.order_item` AS ordit
JOIN `target.orders` AS ord
ON ordit.order_id = ord.order_id
JOIN `target.customer` AS cus
ON ord.customer_id = cus.customer_id
GROUP BY cus.customer_state
ORDER BY total_freight DESC
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	total_freight	avg_freight
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) Analysing number of days taken to deliver an Order And finding the Difference in days between Actual and Estimated delivery days

```sql
SELECT
  *,
  expected_days - days_to_deliver AS diff_in_days
FROM
(
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,day) AS days_to_deliver,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,day) AS expected_days
FROM
(
SELECT *
```

Query results

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

Row	order_id	days_to_deliver	expected_days	diff_in_days
1	1b3190b2dfa9d789e1f14c05b...	208	19	-189
2	ca07593549f1816d26a572e06...	209	28	-181
3	47b40429ed8cce3aee9199792...	191	15	-176
4	2fe324febf907e3ea3f2aa9650...	189	22	-167
5	285ab9426d6982034523a855f...	194	28	-166
6	440d0d17af552815d15a9e41a...	195	30	-165
7	c27815f7e3dd0b926b5855262...	187	25	-162
8	0f4519c5f1c541ddec9f21b3bd...	194	32	-162
9	d24e8541128cea179a11a6517...	175	13	-162

11) Finding TOP 10 states where order delivery is faster compared to estimated delivery date

```sql
SELECT
customer_state,
estimated_days - delivered_in_days AS Avg_diff
FROM (
SELECT
customer_state,
ROUND(AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)),0) AS delivered_in_days,
ROUND(AVG(DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY)),0) AS estimated_days
FROM `target.orders` AS ord
JOIN `target.customer` AS cus
ON ord.customer_id = cus.customer_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY cus.customer_state) AS t1
ORDER BY Avg_diff DESC
LIMIT 10
```

Press Alt+F1 for access

Query results

SAVE RESULTS | OPEN IN

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

Row	customer_state	Avg_diff
1	AC	20.0
2	AP	19.0
3	AM	19.0
4	RO	19.0
5	RR	17.0
6	PA	14.0
7	PB	13.0

12) Finding top 5 Countries with highest average freight value

```sql
WITH main AS (
SELECT *
FROM `target.order_item` AS ordit
JOIN `target.orders` AS ord
ON ordit.order_id = ord.order_id
JOIN `target.customer` AS cus
ON cus.customer_id = ord.customer_id)

SELECT
  customer_state,
  round(AVG(freight_value),2) AS average_freight
FROM main
GROUP BY customer_state
ORDER BY average_freight DESC
LIMIT 5
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

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

13) Finding Number of Order placed Month on month Using different payment types

```sql
SELECT
EXTRACT (MONTH FROM order_purchase_timestamp) AS Month,
payment_type,
COUNT(DISTINCT ord.order_id) AS total_orders
FROM `target.orders` AS ord
JOIN `target.payments` AS pay
ON ord.order_id = pay.order_id
GROUP BY EXTRACT (MONTH FROM order_purchase_timestamp), pay.payment_type
ORDER BY Month
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Month	payment_type	total_orders
1	1	credit_card	6093
2	1	UPI	1715
3	1	voucher	337
4	1	debit_card	118
5	2	credit_card	6582
6	2	UPI	1723
7	2	voucher	288
8	2	debit_card	82
9	3	credit_card	7682

Insights: From the above output we can See that Consumers prefer Credit Card payment method over the others

14) Finding Total Number of order places on Basis of Payment Installation

```sql
SELECT
pay.payment_installments,
COUNT(DISTINCT ord.order_id) AS total_orders
FROM `target.orders` AS ord
JOIN `target.payments` AS pay
ON ord.order_id = pay.order_id
GROUP BY pay.payment_installments
ORDER BY pay.payment_installments
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	payment_installment	total_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644