Target Company

TARGET COMPANY PROJECT

USING SQL + PYTHON

www.linkedin.com/in/md-aftab-uddin

Date: Jan 04, 2025

Presented by: Md Aftab Uddin

Introduction

In this project, I analyzed a fictional e-commerce dataset to derive meaningful insights into customer behavior, sales trends, and business performance. Using Python with MySQL queries in Jupyter Notebook, I explored structured data stored in a relational database and focused on the following aspects:

- Extracting and summarizing key metrics, such as unique customer locations, total orders, and sales per category, to gain an overview of business operations.
- Exploring customer preferences and order behavior by analyzing installment payment trends and the average number of products ordered by city.
- Identifying sales patterns over time, such as monthly order distribution and cumulative sales trends, to understand seasonal demand and growth rates.
- Utilizing advanced SQL techniques, including joins, window functions, and aggregations, to calculate revenue contributions, customer retention rates, and high-value customer rankings.

Project Objectives

Data-Driven Insights

Use SQL in Python to analyze key metrics like customer demographics, sales, and order trends

Trend Analysis and Decision Support

Identify patterns in sales and customer behavior to support strategic decisions.

Advanced Analytical Techniques

Leverage SQL for metrics like retention rates, growth analysis, customer retention rates, moving averages, and year-over-year growth.

1. Calculate the percentage of orders that were paid in installments.

```
q4 = """  select (sum(case when payment_installments >=1 then 1
    else 0 end))/ count(*) *  100 from payments"""
cur.execute(q4)
data = cur.fetchall()
df = pd.DataFrame(data)
print('Total paid installmenys is:' ,data[0][0])
```

```
Total paid installmenys is: 99.9981
```

2. Count the number of orders placed in 2017.

```
que2 = """ select count(order_id) from orders
    where year(order_purchase_timestamp) = 2017"""
cur.execute(que2)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['Count'])
print("Orders placed in 2017:",data[0][0])
```

```
Orders placed in 2017: 45101
```

3. Find the total sales per category.

```
que3 = """ select products.product_category,
round(sum(payments.payment_value),2)
from products
join order_items
on products.product_id = order_items.product_id
join payments
on payments.order_id = order_items.order_id
group by products.product_category"""
cur.execute(que3)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['Categorty','Sales'])
df.head(10)
```

	Categorty	Sales
0	perfumery	506738.66
1	Furniture Decoration	1430176.39
2	telephony	486882.05
3	bed table bath	1712553.67
4	automotive	852294.33
5	computer accessories	1585330.45
6	housewares	1094758.13
7	babies	539845.66
8	toys	619037.69
9	Furniture office	646826.49

4. List all unique cities where customers are located.

```
query = """ select distinct customer_city from customers"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data)
df.columns = ['Customers_City']
df.head(10)
```

	Customers_City
0	franca
1	sao bernardo do campo
2	sao paulo
3	mogi das cruzes
4	campinas
5	jaragua do sul
6	timoteo
7	curitiba
8	belo horizonte
9	montes claros

5. Count the number of customers from each state.

```
q5 = """select customer_state,
count(customer_id) from customers
group by customer_state"""
cur.execute(q5)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['State','Numbers'])
df.head(10)
```

	State	Numbers
0	SP	41746
1	SC	3637
2	MG	11635
3	PR	5045
4	RJ	12852
5	RS	5466
6	PA	975
7	GO	2020
8	ES	2033
9	BA	3380

6. Calculate the number of orders per month in 2018.

```
import calendar
q6 = """ select monthname(order_purchase_timestamp),
count(order_id) from ecommerce.orders
where year(order_purchase_timestamp) = 2018
group by monthname(order_purchase_timestamp)"""
cur.execute(q6)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['Month', 'Order_Num'])
df
```

	Month	Order_Num
0	July	6292
1	August	6512
2	February	6728
3	June	6167
4	March	7211
5	January	7269
6	May	6873
7	April	6939
8	September	16
9	October	4

7. Find the average number of products per order, grouped by customer city.

```
query = """ with count_pr_ord as
(select orders.order_id, orders.customer_id ,
count(order_items.order_item_id) as oc
from orders
join order_items
on orders.order_id = order_items.order_id
group by orders.order_id, orders.customer_id)
select customers.customer_city,
round(avg(count_pr_ord.oc),1)
from customers join count_pr_ord
on customers.customer_id = count_pr_ord.customer_id
group by customers.customer_city """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['customer_city', 'avg_order'])
df.head(10)
```

	customer_city	avg_order
0	sao paulo	1.2
1	sao jose dos campos	1.1
2	porto alegre	1.2
3	indaial	1.1
4	treze tilias	1.3
5	rio de janeiro	1.1
6	mario campos	1.3
7	guariba	1.0
8	cuiaba	1.2
9	franca	1.3

8. Calculate the percentage of total revenue contributed by each product category.

```
query = """select products.product_category,
round(((round(sum(payments.payment_value),2))/
(select sum(payments.payment_value) from payments) * 100),2)
from products
join order_items
on products.product_id= order_items.product_id
join payments
on order_items.order_id = payments.order_id
group by products.product_category """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['prod_ctegory', 'percentage'])
df.head(10)
```

prod_ctegory	percentage
perfumery	3.17
Furniture Decoration	8.93
telephony	3.04
bed table bath	10.70
automotive	5.32
computer accessories	9.90
housewares	6.84
babies	3.37
toys	3.87
Furniture office	4.04

9. Identify the correlation between product price and the number of times a product has been purchased.

```
query = """select products.product_category,
count(order_items.product_id),
round(avg(order_items.price))
from products
join order_items
on products.product_id = order_items.product_id
group by products.product_category"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['Products_cat', 'total_cnt', 'avg_price'])
print(np.corrcoef(df['total_cnt'], df['avg_price']))
```

```
[[ 1.         -0.1064395]
 [-0.1064395  1.       ]]
```

10. Calculate the total revenue generated by each seller, and rank them by revenue.

```
query = """ select  * ,
dense_rank() over(order by revenue desc) as rn from
(select order_items.seller_id,
round(sum( payments.payment_value),2) revenue
from order_items
join payments
on order_items.order_id = payments.order_id
group by order_items.seller_id) as a """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['seller_id', 'revenue', 'rank'])
df.head(10)
```

seller_id	revenue	rank
7c67e1448b00f6e969d365cea6b010ab	507166.91	1
1025f0e2d44d7041d6cf58b6550e0bfa	308222.04	2
4a3ca9315b744ce9f8e9374361493884	301245.27	3
1f50f920176fa81dab994f9023523100	290253.42	4
53243585a1d6dc2643021fd1853d8905	284903.08	5
da8622b14eb17ae2831f4ac5b9dab84a	272219.32	6
4869f7a5dfa277a7dca6462dcf3b52b2	264166.12	7
955fee9216a65b617aa5c0531780ce60	236322.30	8
fa1c13f2614d7b5c4749cbc52fecda94	206513.23	9
7e93a43ef30c4f03f38b393420bc753a	185134.21	10

11. Calculate the moving average of order values for each customer over their order history.

```
query = """select customer_id,
order_purchase_timestamp, avg(pay)
over (partition by customer_id order
by order_purchase_timestamp rows
between 2 preceding and current row) as moving_avg
from
(select orders.customer_id,
orders.order_purchase_timestamp,
payments.payment_value as pay
from orders
join payments
on orders.order_id = payments.order_id) as a"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns=['Customer_id','Time','Avg pay'])
df.head(10)
```

Customer_id	Time	Avg pay
00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	114.739998
000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	67.410004
0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	195.419998
0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	179.350006
000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	107.010002
0004164d20a9e969af783496f3408652	2017-04-12 08:35:12	71.800003
000419c5494106c306a97b5635748086	2018-03-02 17:47:40	49.400002
00046a560d407e99b969756e0b10f282	2017-12-18 11:08:30	166.589996
00050bf6e01e69d5c0fd612f1bcfb69c	2017-09-17 16:04:44	85.230003
000598caf2ef4117407665ac33275130	2018-08-11 12:14:35	1255.709961

12. Calculate the cumulative sales per month for each year.

```
query = """select years, months,
sum(total_payment)
over(order by years, months) pay from
(select year(orders.order_purchase_timestamp)
as years,
monthname(orders.order_purchase_timestamp)
as months,
round(sum(payments.payment_value),2)
total_payment
from orders
join payments
on orders.order_id = payments.order_id
group by years, months) as a
order by years asc """
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['years','month', 'cum_sales'])
df
```

	years	month	cum_sales
0	2016	December	19.62
1	2016	October	59110.10
2	2016	September	59362.34
3	2017	April	477150.37
4	2017	August	1151546.69
5	2017	December	2029948.17
6	2017	February	2321856.18
7	2017	January	2460344.22
8	2017	July	3052727.14
9	2017	June	3564003.52
10	2017	March	4013867.12
11	2017	May	4606785.94
12	2017	November	5801668.74
13	2017	October	6581346.62
14	2017	September	7309109.07
15	2018	April	8469894.55
16	2018	August	9492319.87
17	2018	February	10484783.21
18	2018	January	11599787.39
19	2018	July	12666328.14
20	2018	June	13690208.64
21	2018	March	14849860.76
22	2018	May	16003842.91
23	2018	October	16004432.58
24	2018	September	16008872.12

13. Calculate the year-over-year growth rate of total sales.

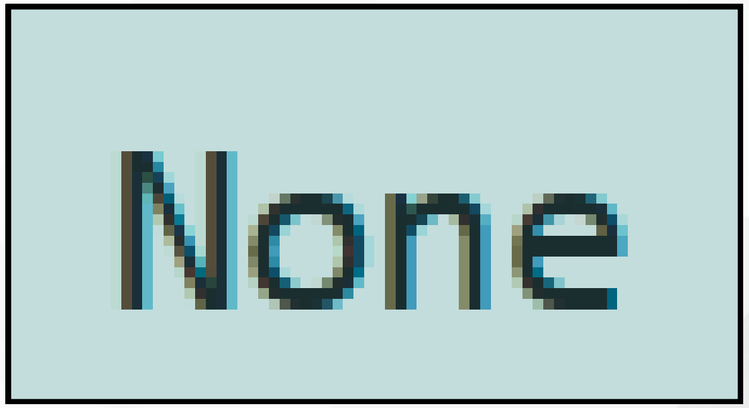
```
query = """ with a as
(select year(orders.order_purchase_timestamp) as years,
round(sum(payments.payment_value),2) as pay
from orders
join payments
on orders.order_id = payments.order_id
group by years)
select years, pay,
round(((pay - lag(pay, 1) over(order by years)) /
lag(pay, 1) over(order by years)) * 100,2) from a"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ['years', 'total_payment', 'growth'])
df['growth'] = df['growth'].fillna(0)
df
```

years	total_payment	growth
2016	59362.34	0.0
2017	7249746.73	12112.7
2018	8699763.05	20.0

14. Calculate the retention rate of customers, defined as the percentage of customers who make another purchase within 6 months of their first purchase.

```
query = """with a as (select customers.customer_id,
min(orders.order_purchase_timestamp) first_order
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.customer_id),
b as (select a.customer_id, count(distinct orders.order_purchase_timestamp) next_order
from a join orders
on orders.customer_id = a.customer_id
and orders.order_purchase_timestamp > first_order
and orders.order_purchase_timestamp <
date_add(first_order, interval 6 month)
group by a.customer_id)
select 100 * (count( distinct a.customer_id)/ count(distinct b.customer_id))
from a left join b
on a.customer_id = b.customer_id ;"""

cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data)
data[0][0]
```



15. Identify the top 3 customers who spent the most money in each year.

```
query = """select years,
customer_id, round(payment,2),
d_rank from
(select year(orders.order_purchase_timestamp) years,
orders.customer_id,
sum(payments.payment_value) payment,
dense_rank() over(partition by
year(orders.order_purchase_timestamp)
order by sum(payments.payment_value) desc) d_rank
from orders
join payments
on payments.order_id = orders.order_id
group by year(orders.order_purchase_timestamp),
orders.customer_id) as a
where d_rank <= 3 ;"""
cur.execute(query)
data = cur.fetchall()
df = pd.DataFrame(data, columns = ["years","customer_id","payment","rank"])
print(df)
```

years	customer_id	payment	rank
2016	a9dc96b027d1252bbac0a9b72d837fc6	1423.55	1
2016	1d34ed25963d5aae4cf3d7f3a4cda173	1400.74	2
2016	4a06381959b6670756de02e07b83815f	1227.78	3
2017	1617b1357756262bfa56ab541c47bc16	13664.08	1
2017	c6e2731c5b391845f6800c97401a43a9	6929.31	2
2017	3fd6777bbce08a352fddd04e4a7cc8f6	6726.66	3
2018	ec5b2ba62e574342386871631fafd3fc	7274.88	1
2018	f48d464a0baaea338cb25f816991ab1f	6922.21	2
2018	e0a2412720e9ea4f26c1ac985f6a7358	4809.44	3

Conclusion and Q&A

- Key Insights Uncovered: Extracted valuable information on customer behavior, sales trends, and operational performance using SQL queries in Python.
- Data-Driven Decision Support: Provided actionable insights for optimizing business strategies, improving customer retention, and enhancing revenue growth.
- Advanced Techniques Utilized: Successfully applied advanced SQL methods, including window functions and aggregations, to perform complex analysis and derive meaningful business metrics.

Target Company

BIG THANKS

Explore the Code on My GitHub Repository

www.AftabQuant.com