zudio

ZUDIO PROJECT

Store Sales and Customer Analytics

zudio

Introduction

In Today's retail market, businesses need to use data to make smart decisions and grow successfully. This project is about analyzing Zudio's sales, customer habits, and store performance using SQL and Python. By using these tools together, we aim to find useful insights that can help improve store operations, keep customers happy, and increase profits.

MySQL Questions

Easy Level (Basic Queries)

1.Retrieve the total number of stores per state.
2. List the top 5 products by total sales revenue.
3. Get the count of orders placed in each month.
4.Find the total sales generated by each store type (Owned vs. Rented).
5.Identify the most 3 frequently sold clothing Type.

Moderate Level (Intermediate Queries)

1. Calculate the average profit margin for each clothing type.
2. List the top 3 customers who have generated the highest sales revenue.
3. Determine the total quantity of products sold in each city.
4. Identify the stores that opened in the last two years along with their sales.
5. Analyze the sales trends by month for the current year.

Advanced Level (Complex Queries)

1.Find the state with the highest sales per square foot of selling area.
2. Identify the top 5 most profitable stores and their key performance metrics (e.g., sales, profit, sales per employee).
3. Calculate the percentage contribution of each category to the total sales across all stores.
4 Create a cohort analysis of customers based on their first purchase month and track their monthly sales contribution.
5. Identify anomalies where the sales amount deviates significantly from the average store sales in the same city.

Python Questions

Easy Level (Basic Queries)

1.Analyze Store Distribution by State.
2. Top 5 Products by Total Sales Profit.
3. Monthly Orders Count.
4.Store Type vs Product Sales.
5.Most Sold Clothing Category.

Moderate Or Advance Level (Intermediate Queries)

1.Average Profit Margin by Clothing Type.
2. Top 3 Customers by Highest Profit.
3. Total Quantity Sold by City.
4. Percentage of Total Sales by Operating Hours .
5. State with Highest Sales Profit per Square Foot
6. Sales Quantity Anomalies Detection

MySQL Queries

Simple Questions:-

Q1. Retrieve the total number of stores per state.

```
SELECT
   State, COUNT(Store) AS Total_Store
FROM
   zudio_data
GROUP BY State
ORDER BY Total_Store DESC;
```

Q2). List the top 5 products by total sales revenue.

```
SELECT
   Product_ID, SUM(Price) AS Total_Sales_Revenue
FROM
   zudio_data
GROUP BY Product_ID
ORDER BY Total_Sales_Revenue
LIMIT 5;
```

Q3). Get the count of orders placed in each month.

```
SELECT
   Month, COUNT(*) AS Total_Orders
FROM
   zudio_data
GROUP BY Month
ORDER BY Total_Orders DESC;
```

MySQL Queries

Q4). Find the total sales generated by each store type (Owned vs. Rented).

```
SELECT
   Store_Type, COUNT(*) AS Total_Sales
FROM
   zudio_data
GROUP BY Store_Type;
```

Q5). Identify the most Top 3 frequently sold clothing Type.

```
SELECT
   Clothing_Type, COUNT(*) AS Total_Sales
FROM
   zudio_data
GROUP BY Clothing_Type
ORDER BY Total_Sales DESC
LIMIT 3;
```

Moderate Level (Intermediate Queries)

Q1). Calculate the average profit margin for each clothing type.

```
SELECT
   Clothing_Type,
   ROUND(AVG(Sales_Profit), 2) AS Avgerage_Profit
FROM
   zudio_data
GROUP BY Clothing_Type
ORDER BY Avgerage_Profit DESC;
```

MySQL Queries

Q2). List the top 3 customers who have generated the highest sales revenue.

```
SELECT
   Customer_Name, SUM(Price) AS Higest_Sales_Revenue
FROM
   zudio_data
GROUP BY Customer_Name
ORDER BY Higest_Sales_Revenue DESC
LIMIT 3;
```

Q3). Determine the total quantity of products sold in each city.

```
SELECT
   City, SUM(Quantity) AS Total_Products_Sold
FROM
   zudio_data
GROUP BY City
ORDER BY Total_Products_Sold DESC;
```

Q4). Identify the stores that opened in the last two years along with their sales.

```
SELECT
    Store_Open_Date, COUNT(*) AS Sales
FROM
   zudio_data
WHERE
   STR_TO_DATE(Store_Open_Date, '%d-%m-%Y') >= DATE_SUB(CURDATE(), INTERVAL 2 YEAR)
GROUP BY Store_Open_Date;
```

MySQL Queries

Q5). Analyze the sales trends by month for the current year.

```
SELECT
   Month, COUNT(*) AS Sales
FROM
   zudio_data
WHERE
   YEAR(STR_TO_DATE(Order_Date, '%Y-%m-%d')) = YEAR(CURDATE())
GROUP BY Month;
```

Advanced Level (Complex Queries)

Q1).Find the state with the highest sales quantity -per square foot of selling area.

```
SELECT
   State,
   sum(Quantity/ Selling_Area_Size) AS Highest_Sales_per_Square_Foot
FROM
   zudio_data
GROUP BY
   State
ORDER BY
   Highest_Sales_per_Square_Foot DESC
LIMIT 1;
```

MySQL Queries

Q2).Identify the top 5 most profitable stores(Postal_Code) and their key performance metrics (e.g., sales, profit, sales per employee).

```
SELECT
   Postal_Code,
   SUM(Quantity) AS Sales,
   SUM(Sales_Profit) AS Profit,
   SUM(Quantity) / COUNT(Customer_ID) AS Sales_Per_Employee
FROM
   zudio_data
GROUP BY Postal_Code
ORDER BY Profit DESC
LIMIT 5;
```

Q3).Calculate the percentage contribution of each category to the total sales across all store.

```
SELECT
   Category,
   Sum(Quantity) AS Total_Sales,
   (SUM(Quantity) / (SELECT SUM(Quantity) FROM zudio_data) * 100) AS Percentage_Contribution
FROM
   zudio_data
GROUP BY
   Category
ORDER BY
   Percentage_Contribution DESC;
```

MySQL Queries

Q4).Create a cohort analysis of customers based on their first purchase month and track their monthly sales contribution.

```
 WITH first_purchase AS (
  SELECT
    Customer_ID,
    DATE_FORMAT(MIN(STR_TO_DATE(Order_Date, '%Y-%m-%d')), '%Y-%m') AS first_purchase_month
  FROM
    zudio_data
  WHERE
    Order_Date IS NOT NULL
  GROUP BY
    Customer_ID
),
monthly_sales AS (
  SELECT
    Customer_ID,
    DATE_FORMAT(STR_TO_DATE(Order_Date, '%Y-%m-%d'), '%Y-%m') AS purchase_month,
    SUM(Quantity) AS monthly_sales
  FROM
    zudio_data
  WHERE
    Order_Date IS NOT NULL
  GROUP BY
    Customer_ID, DATE_FORMAT(STR_TO_DATE(Order_Date, '%Y-%m-%d'), '%Y-%m')
)
```

MySQL Queries

```sql
SELECT
 fp.first_purchase_month AS cohort_month,
 ms.purchase_month AS contribution_month,
 COUNT(DISTINCT ms.Customer_ID) AS customers_in_cohort,
 SUM(ms.monthly_sales) AS total_sales
FROM
 first_purchase fp
JOIN
 monthly_sales ms
ON
 fp.Customer_ID = ms.Customer_ID
GROUP BY
 fp.first_purchase_month, ms.purchase_month
ORDER BY
 fp.first_purchase_month, ms.purchase_month;
```

MySQL Queries

Q5).Identify anomalies where the sales amount deviates significantly from the average store sales in the same city.

```
    WITH city_sales AS (
    SELECT
        zd.Order_ID,
        zd.City,
        zd.Price,
        AVG(zd.Price) OVER (PARTITION BY zd.City) AS avg_city_sales,
        ABS(zd.Price - AVG(zd.Price) OVER (PARTITION BY zd.City)) AS deviation
    FROM
        zudio_data zd
)
SELECT
    Order_ID,
    City,
    Price,
    avg_city_sales,
    deviation
FROM
    city_sales
WHERE
    deviation > 0.3 * avg_city_sales
ORDER BY
    deviation DESC;
```

Python Queries

Easy Level (Basic Queries)

1.Analyze Store Distribution by State.

```
Store_Distribution = df.groupby('State')['Order_ID'].count().reset_index()
Store_Distribution.columns = ['State','Total_Orders']
Store_Distribution = Store_Distribution.sort_values(by='Total_Orders',ascending = False)
Store_Distribution

plt.figure(figsize = (10,6))
plt.bar(Store_Distribution['State'],Store_Distribution['Total_Orders'],color=('r','b','k','y','g','k','r','b'))
plt.title('State Vs Total_Orders',fontsize = 14)
plt.xlabel('State',fontsize = 12)
plt.ylabel('Total_Orders',fontsize = 12)
plt.xticks(rotation = 45 ,fontsize =10)
plt.tight_layout()
plt.show()
```

2. Top 5 Products by Total Sales Profit.

```
Products = df.groupby('Clothing_Type')['Sales_Profit'].sum().reset_index()
Products.columns = ['Products','Total_Profit']
Top_Products = Products.sort_values(by = 'Total_Profit',ascending = False).head(5)
Top_Products

plt.figure(figsize=(7,6))
plt.barh(Top_Products['Products'],Top_Products['Total_Profit'],color = ('r','b','k','y','g'))
plt.title('Products Vs Total_Profit')
plt.xlabel('Total_Profit')
plt.ylabel('Products')
plt.xticks(rotation = 30 ,fontsize = 10)
plt.show()
```

Python Queries

3. Monthly Orders Count.

```
Orders = df.groupby('Month')['Order_ID'].count().reset_index()
Orders.columns = ['Month', 'Total_Orders']
Orders = Orders.sort_values(by='Total_Orders',ascending = False)
Orders

plt.figure(figsize = (10,4))
sns.lineplot(x="Month",y="Total_Orders",data=Orders,palette='spring')
plt.xticks(rotation = 45)
plt.show()
```

4.Store Type vs Product Sales.

```
Store = df.groupby('Store_Type')['Product_ID'].count().reset_index()
Store.columns = ['Store_Type','Total_Product_Sell']
Store = Store.sort_values(by='Total_Product_Sell',ascending=False)
Store

labels = Store['Store_Type']
sizes = Store['Total_Product_Sell']
plt.pie(sizes,labels=labels,autopct='%1.1f%%', startangle=90)
plt.show()
```

5.Most Sold Clothing Category.

```
Clothing_Category = df.groupby('Category')['Quantity'].sum().reset_index()
Clothing_Category.columns = ['Clothing_Category','Most_Sold']
Clothing_Category = Clothing_Category.sort_values(by='Most_Sold',ascending = False)
Clothing_Category
```

Python Queries

Moderate Or Advance Level (Intermediate Queries)

1.Average Profit Margin by Clothing Type.

```
Avg_profit = df.groupby('Clothing_Type')['Sales_Profit'].mean().reset_index()
Avg_profit.columns = ['Clothing_Type','Avg_Profit']
Avg_profit = Avg_profit.sort_values(by='Avg_Profit',ascending = False)
Avg_profit

plt.figure(figsize=(8,4))
sns.barplot(x= 'Clothing_Type',y= 'Avg_Profit',data = Avg_profit , palette = 'spring')
plt.xlabel('Clothing Category', fontsize=12)
plt.ylabel('Average Profit ', fontsize=12)
plt.title('Average Profit by Clothing Type', fontsize=14)
plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```

2. Top 3 Customers by Highest Profit.

```
Customers = df.groupby('Customer_Name')['Sales_Profit'].sum().reset_index()
Customers.columns = ['Customer_Name','Total_Profit']
Customers = Customers.sort_values(by='Total_Profit',ascending = False).head(3)
Customers

plt.figure(figsize=(8, 5))
plt.bar(Customers['Customer_Name'], Customers['Total_Profit'], color='orange')
plt.xlabel('Customer Name', fontsize=12)
plt.ylabel('Total Profit', fontsize=12)
plt.title('Top 3 Customers by Profit', fontsize=14)
plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```

Python Queries

3. Total Quantity Sold by City.

```
City = df.groupby('City')['Quantity'].sum().reset_index()
City.columns = ['City','Total_Sales']
City = City.sort_values(by='Total_Sales',ascending = False)
City

plt.figure(figsize = (14,4))
sns.scatterplot(x='City',y='Total_Sales',data=City,marker = '*', alpha = 1)
plt.xticks(rotation = 75 , fontsize = 10)
plt.show()
```

4. Percentage of Total Sales by Operating Hours .

```
total_sales = df.groupby('Operating_Hours')['Quantity'].sum().reset_index()
total_sales.columns = ['Operating_Hours','Total_Sales']
total_sales = total_sales.sort_values(by='Total_Sales',ascending = False)
total_sales

sns.lineplot(x='Operating_Hours',y='Total_Sales',data = total_sales)
plt.xlabel('Operating_Hours', fontsize=12)
plt.ylabel('Total Sales', fontsize=12)
plt.title('Operating_Hours Vs Total_Sales', fontsize=14)
plt.xticks(rotation=15, fontsize=10)
plt.tight_layout()
plt.show()
```

5. State with Highest Sales Profit per Square Foot

```
df['Profit_per_SqFt'] = df['Sales_Profit'] / df['Selling_Area_Size(sq ft)']
state_sales = df.groupby('State')['Profit_per_SqFt'].mean().reset_index()
state_sales.columns = ['State','Profit_per_SqFt']
highest_state = state_sales.loc[state_sales['Profit_per_SqFt'].idxmax()]
highest_state

plt.figure(figsize=(8, 5))
plt.bar(state_sales['State'], state_sales['Profit_per_SqFt'], color='skyblue')
plt.xlabel('State', fontsize=12)
plt.ylabel('Profit per Square Foot', fontsize=12)
plt.title('Profit per Square Foot by State', fontsize=14)
plt.xticks(rotation=45, fontsize=10)
plt.tight_layout()
plt.show()
```

6. Sales Quantity Anomalies Detection

```
city_stats = df.groupby('City')['Quantity'].agg(['mean', 'std']).reset_index()
city_stats.rename(columns={'mean': 'City_Avg_Sales', 'std': 'City_Sales_Std'}, inplace=True)
df = pd.merge(df, city_stats, on='City')
df['Z_Score'] = (df['Quantity'] - df['City_Avg_Sales']) / df['City_Sales_Std']
threshold = 2
df['Anomaly'] = np.abs(df['Z_Score']) > threshold
df

normal_data = df[df['Anomaly'] == False]
anomaly_data = df[df['Anomaly'] == True]
plt.figure(figsize=(14, 5))
plt.scatter(normal_data['City'], normal_data['Quantity'], label='Normal', color='blue')
plt.scatter(anomaly_data['City'], anomaly_data['Quantity'], label='Anomaly', color='red', marker='x')
plt.xlabel('City', fontsize=12)
plt.ylabel('Quantity', fontsize=12)
plt.title('Sales Anomalies Detection by City', fontsize=14)
plt.legend()
plt.xticks(rotation=60)
plt.tight_layout()
plt.show()
```