Name- Anushka Sawant.
Roll No- CS6-24.
PRN- 2024011100150

Page No.
Date

Formulate 20 problem statement of grocery using Numpy and Pandas method to find the solution for the formulated problem statements. Each one will take a real life dataset.

1) Price statistics per Category.

Problem:- Calculate the mean, median and standard deviation of product price in each category.

Method:-

```
df.groupby("Category")["Price"].agg(["mean","median","std"]).
```

2) Total Revenue per Month.

Problem:- Compute monthly revenue from transaction records.

Method:-

```
df["Date"] = pd.to_datetime(df["Date"])
df.groupby(df["Date"].dt.to_period("M"))["Revenue"].sum().
```

3) Top 5 selling products.

Problem:- Identify the top 5 products by total quantity sold.

Method:-

```
df.groupby("Product")["Quantity"].sum().nlargest(5).
```

4) Out of Stock Product.

Problem :- List all products that have zero inventory.

Method :- df[df["stock"] == 0]["Product"].

5) Average Basket Size.

Problem :- Calculate the average number of items per transaction.

Method :-
df.groupby("TransactionID")["Quantity"].sum
mean

6) Most Profitable Product.

Problem :- Identify the product with the highes profit.

Method :-
df["Profit"] = df["SellingPrice"] - df["CostPrice"]
df.groupby("Product")["Profit"].sum(). idxmax

7) Day with Highest sales.

Problem :- Find the day of the week with the highest sales volume.

Method:

```
df["Date"] = pd.to_datetime(df["Date"])
df["Day"] = df["Date"].dt.day_name()
df.groupby("Day")["Quantity"].sum().idxmax()
```

8) Seasonal Demand Patterns.

Problem: Visualize sales trends across different months.

Method:

```
df["Month"] = df["Date"].dt.month
df.groupby("Month")["Quantity"].sum().plot()
```

9) Stock Replenishment Needs.

Problem:- Identify products with stock below a threshold. (e.g = 10 units)

Method:-

```
df[df["stock"] < 10][["Product", "stock"]]
```

10) Total Discount Given.

Problem: Compute the total discount given on all transaction.

Method

```
df["Discount Amount"] = df["Original Price"] -
                        df["Selling Price"].
df["Discount Amount"].sum().
```

11) Unique Customers per Store.

Problem: Count the number of unique customers in each store.

Method:

df.groupby("Store ID")["Customer ID"].nuniq

12) Average Purchase Value.

Problem: Determine the average value of customer purchases.

Method:

df.groupby("Customer ID")["Total"].mean

13) Frequent Itemsets (Market Basket).

Problem: Identify frequently bought-togeth items using combination.

Method:

from itertools import combination.

14) Revenue by Product Category

Problem: Calculate total revenue by prod Category.

Method:

df.groupby("Category")["Revenue"].sum(

15) Highr Spending Customers.

Problem - Identify top 10 customers based on total spending.

Method.

```
df.groupby("CustomerID")["Total"].sum().
                                nlargest(10)
```

16) Products Never Sold.

Problem: List products with zero sales.

Method:

```
df.groupby("Product")["Quantity"].sum().
                                eq(0).
```

17) Peak Shopping Hours.

Problem: Find which hour of the day has the highest transaction count.

Method:

```
df["Hour"] = pd.to_datetime(df["Time"]).dt.hour
df["Hour"].value_counts().idxmax()
```

18) Expiring Products.

Problem: Identify products expiring within the next 30 days.

Method:

```
today = pd.Timestamp.now()
df["ExpiryDate"] = pd.to_datetime(df["ExpiryDate"])
df[df["ExpiryDate"] < today + pd.Timedelta(days=30)]
```

19) Weekly Sales Growth.

Problem: Calculate week-over-week growth in sales.

Method:

```
Weekly_sales = df.resample("W", on="Date")
                 ["Revenue"].sum()
weekly_sales.pct_change()
```

20) Profit Margin per Category.

Problem: Calculate average profit margin per category.

Method:

```
df["Margin"] = (df["Selling Price"] -
                df["Cost Price"]) / df["Cost Price"]
df.groupby("Category")["Margin"].mean()
```