

SQL AND PYTHON

Name: Jaavanaika L

Step 1: Clean Data in SQL (SQLite)

1. Remove Null or Missing Records In DB Browser > Execute SQL, run:

```
DELETE FROM superstore  
WHERE "Order ID" IS NULL  
    OR "Category" IS NULL  
    OR "Sales" IS NULL  
    OR "Profit" IS NULL;
```

2. Remove Duplicates (Based on Order ID)

Since SQLite doesn't have ROW_NUMBER(), use this workaround:

```
DELETE FROM superstore  
WHERE rowid NOT IN (  
    SELECT MIN(rowid)  
    FROM superstore  
    GROUP BY "Order ID"  
);
```

3. Verify Cleaned Table

Check the number of rows after cleaning:

```
SELECT COUNT(*) FROM superstore;
```

4. Export Cleaned Table (Optional) You

can now:

- Go to File > Export > Table as CSV
- Save as superstore_cleaned.csv
(You'll use this cleaned CSV in Python and Tableau)

Step 2: SQL Profitability Analysis (Clean Data)

A. Profit by Category

SELECT

```
"Category" AS category,  
ROUND(SUM(Sales), 2) AS total_sales,  
ROUND(SUM(Profit), 2) AS total_profit,  
ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent  
FROM superstore_cleaned  
GROUP BY "Category"  
ORDER BY profit_margin_percent ASC;
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the query code. The Results tab displays a table with three rows of data:

category	total_sales	total_profit	profit_margin_percent
Furniture	373504.72	7569.17	2.03
Office Supplies	345716.46	54789.0	15.95
Technology	380640.9	70157.57	18.43

B. Profit by Sub-Category

SELECT

```
"Sub-Category" AS sub_category,  
ROUND(SUM(Sales), 2) AS total_sales,  
ROUND(SUM(Profit), 2) AS total_profit,  
ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent  
FROM superstore_cleaned  
GROUP BY "Sub-Category"  
ORDER BY profit_margin_percent ASC;
```

DB Browser for SQLite - C:\Users\JAAVANIKA\dbc.d.b

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Execute SQL Edit Pragmas

SQL 1* SQL 2* SQL 3* SQL 4*

```

1 SELECT
2   "Sub-Category" AS sub_category,
3   ROUND(SUM(Sales), 2) AS total_sales,
4   ROUND(SUM(Profit), 2) AS total_profit,
5   ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent
6 FROM superstore_cleaned
7 GROUP BY "Sub-Category"
8 ORDER BY profit_margin_percent ASC;
9 
```

sub_category	total_sales	total_profit	profit_margin_percent
Tables	104410.66	-10997.07	-10.53
Supplies	13027.94	-754.1	-5.79
Bookcases	54547.46	-1247.24	-2.29
Chairs	166703.65	13200.89	7.92
Binders	103534.39	10027.44	9.69
Machines	75844.42	7606.82	10.03
Storage	107383.04	12060.17	11.23
Phones	162431.75	20938.78	12.89
Appliances	52911.18	6925.07	13.09
Furnishings	47842.95	6612.6	13.82
Art	14966.64	3612.16	24.13
Accessories	72235.69	19209.27	26.59
Fasteners	1606.66	496.93	30.93
Copiers	70129.03	22402.7	31.94
Envelopes	9254.82	3856.87	41.67
Paper	37897.93	16285.49	42.97
Labels	5133.85	2278.97	44.39

Remote
Identity Select an identity to connect
DBHub.io Local Current Database
Name Last modified

SQL Log Plot DB Schema Remote

Wi-AUS In 1 hour ENG IN 18:38 03-07-2023

C. Profit by Category + Sub-Category

SELECT

```

"Category" AS category,
"Sub-Category" AS sub_category,
ROUND(SUM(Sales), 2) AS total_sales,
ROUND(SUM(Profit), 2) AS total_profit,
ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent

```

FROM superstore

GROUP BY "Category", "Sub-Category"

ORDER BY profit_margin_percent ASC;

DB Browser for SQLite - C:\Users\JAAVANIKA\dbc.d.b

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Execute SQL Edit Pragmas

SQL 1* SQL 2* SQL 3* SQL 4*

```

1 ANALYTIC
2   "Category" AS category,
3   "Sub-Category" AS sub_category,
4   ROUND(SUM(Sales), 2) AS total_sales,
5   ROUND(SUM(Profit), 2) AS total_profit,
6   ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent
7 FROM superstore_cleaned
8 GROUP BY "Category", "Sub-Category"
9 ORDER BY profit_margin_percent ASC;
10 
```

category	sub_category	total_sales	total_profit	profit_margin_percent
Furniture	Tables	104410.66	-10997.07	-10.53
Office Supplies	Supplies	13027.94	-754.1	-5.79
Furniture	Bookcases	54547.46	-1247.24	-2.29
Furniture	Chairs	166703.65	13200.89	7.92
Office Supplies	Binders	103534.39	10027.44	9.69
Technology	Machines	75844.42	7606.82	10.03
Office Supplies	Storage	107383.04	12060.17	11.23
Technology	Phones	162431.75	20938.78	12.89
Office Supplies	Appliances	52911.18	6925.07	13.09
Furniture	Furnishings	47842.95	6612.6	13.82
Office Supplies	Art	14966.64	3612.16	24.13
Technology	Accessories	72235.69	19209.27	26.59
Office Supplies	Fasteners	1606.66	496.93	30.93
Technology	Copiers	70129.03	22402.7	31.94
Office Supplies	Envelopes	9254.82	3856.87	41.67
Office Supplies	Paper	37897.93	16285.49	42.97
Office Supplies	Labels	5133.85	2278.97	44.39

Remote
Identity Select an identity to connect
DBHub.io Local Current Database
Name Last modified

SQL Log Plot DB Schema Remote

Finance headline US consumer inc... 18:38 03-07-2023

D. Profit by Region

SELECT

```
"Region",
ROUND(SUM(Sales), 2) AS total_sales,
ROUND(SUM(Profit), 2) AS total_profit,
ROUND((SUM(Profit) / SUM(Sales)) * 100, 2) AS profit_margin_percent
FROM superstore
GROUP BY "Region"
ORDER BY profit_margin_percent ASC;
```

The screenshot shows the DB Browser for SQLite interface. The SQL tab contains the query provided above. The Results tab displays a table with four rows of data:

	Region	total_sales	total_profit	profit_margin_percent
1	Central	246314.52	11357.04	4.61
2	South	190409.23	21292.62	11.18
3	West	332443.71	47420.44	14.26
4	East	330694.63	52445.68	15.86

Step 3: Python – Correlation Between Inventory Days & Profitability

Visualizations (Python/Seaborn/Matplotlib) import

pandas as pd

import numpy as np

```
# Load cleaned data
df = pd.read_csv("superstore_cleaned.csv")

# Simulate Inventory Days (since not in original dataset)
np.random.seed(42)
df["Inventory Days"] = np.random.randint(10, 101, size=len(df))

# Convert date columns to datetime df["Order Date"]
= pd.to_datetime(df["Order Date"]) df["Month"] =
```

```

df["Order Date"].dt.month df["Season"] =
df["Month"].map({ 12: "Winter", 1: "Winter", 2:
"Winter",
3: "Spring", 4: "Spring", 5: "Spring",
6: "Summer", 7: "Summer", 8: "Summer",
9: "Fall", 10: "Fall", 11: "Fall"
})

```

- Scatter Plot: Inventory Days vs Profit Margin grouped =**

```

df.groupby("Sub-Category").agg({
    "Sales": "sum",
    "Profit": "sum",
    "Inventory Days": "mean"
}).reset_index()
grouped["Profit Margin (%)"] = (grouped["Profit"] / grouped["Sales"]) * 100

```

```

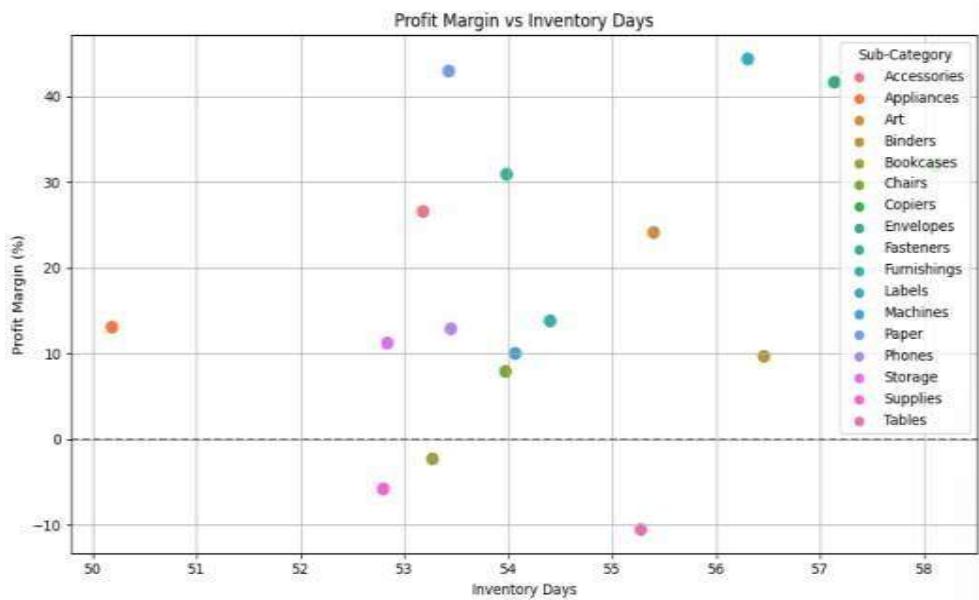
import seaborn as sns
import matplotlib.pyplot as plt

```

```

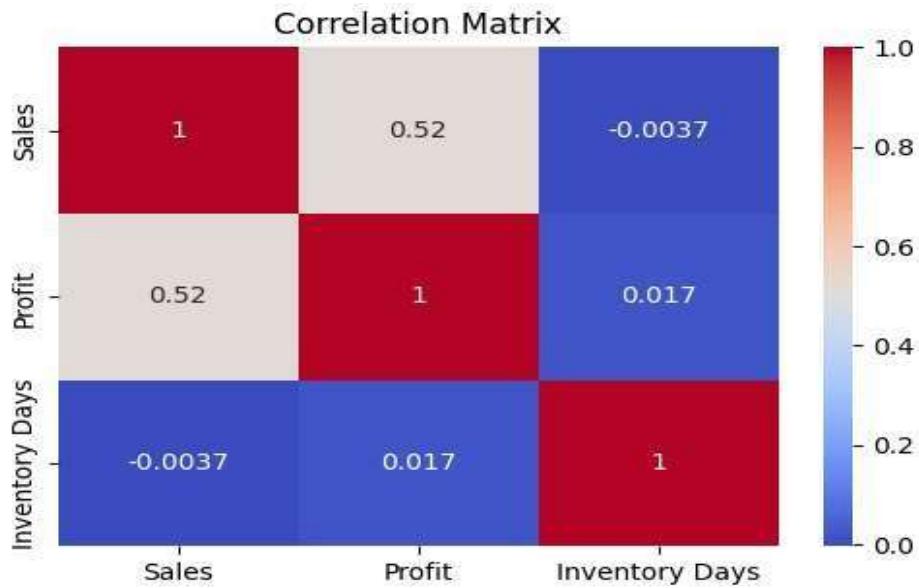
plt.figure(figsize=(10,6))
sns.scatterplot(data=grouped, x="Inventory Days", y="Profit Margin (%)",
hue="Sub-Category", s=100) plt.title("Profit Margin vs Inventory Days")
plt.axhline(0, linestyle='--', color='gray') plt.grid(True) plt.tight_layout()
plt.show()

```



2. Heatmap: Inventory Days vs Profit Margin (Correlation Matrix)

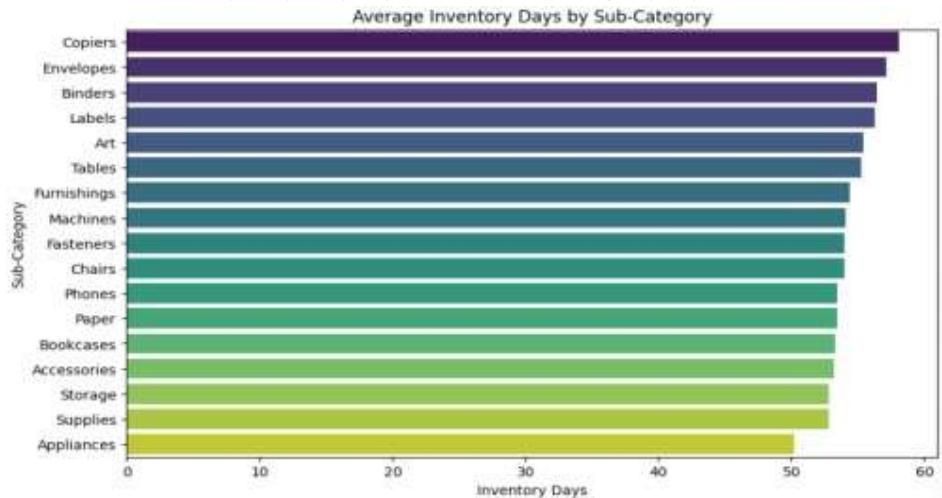
```
Correlation between numeric columns plt.figure(figsize=(6,4))  
sns.heatmap(df[["Sales", "Profit", "Inventory Days"]].corr(), annot=True,  
cmap='coolwarm')  
plt.title("Correlation Matrix") plt.show()
```



3. Bar Chart: Sub-Categories with Highest Inventory Days inv_days =

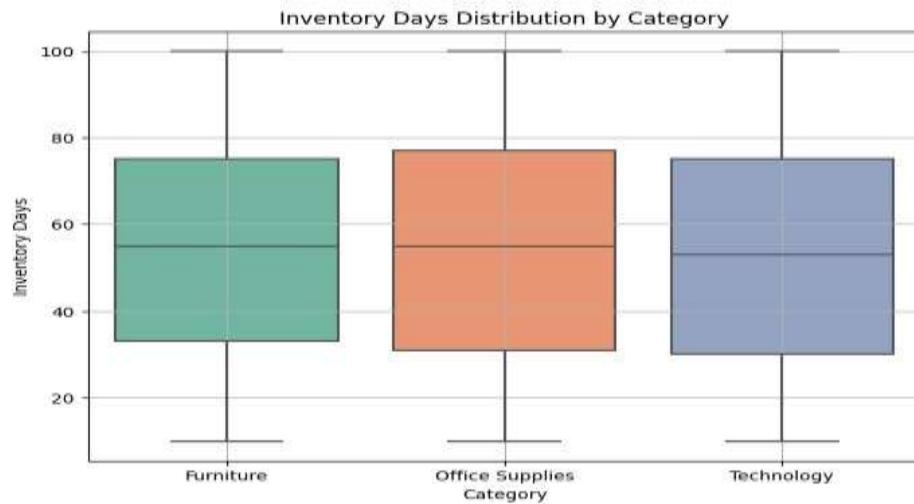
```
df.groupby("Sub-Category")["Inventory  
Days"].mean().sort_values(ascending=False)
```

```
plt.figure(figsize=(10,6))  
sns.barplot(x=inv_days.values, y=inv_days.index, palette="viridis")  
plt.title("Average Inventory Days by Sub-Category")  
plt.xlabel("Inventory Days") plt.ylabel("Sub-Category") plt.show()
```



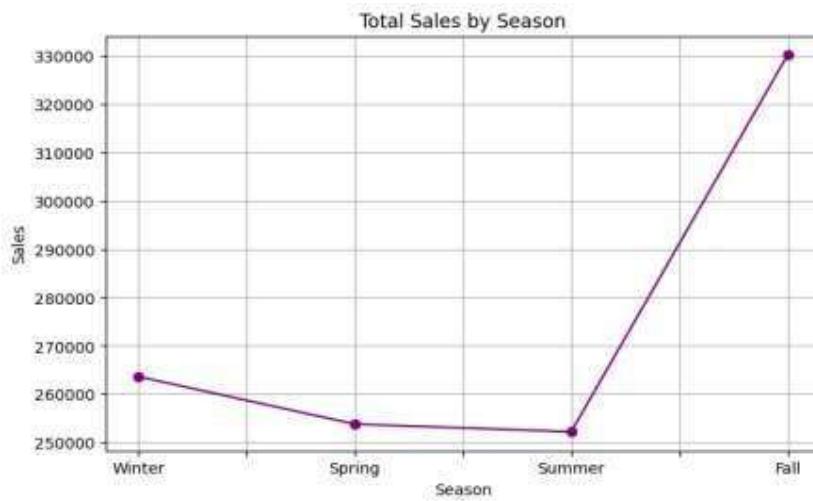
4. Box Plot: Inventory Days Distribution by Category

```
plt.figure(figsize=(8,6))
sns.boxplot(data=df, x="Category", y="Inventory Days", palette="Set2")
plt.title("Inventory Days Distribution by Category") plt.grid(True)
plt.show()
```



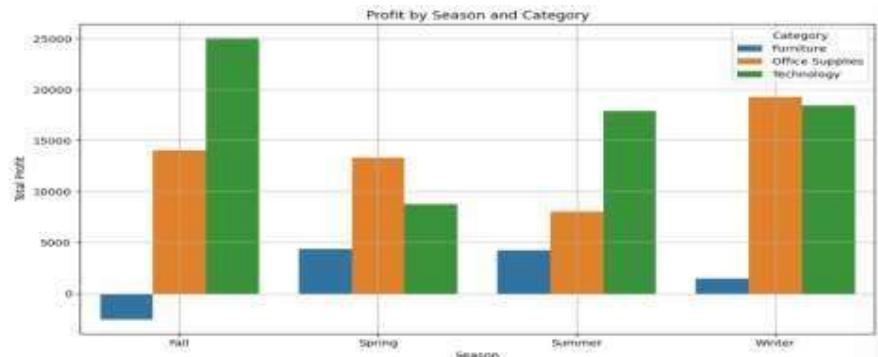
5. Seasonal Sales Trend (Line Plot by Season) season_sales =

```
df.groupby("Season")["Sales"].sum().reindex(["Winter",
"Spring", "Summer", "Fall"])
plt.figure(figsize=(8,5))
season_sales.plot(kind="line", marker='o', color="purple")
plt.title("Total Sales by Season")
plt.xlabel("Season")
plt.ylabel("Sales")
plt.grid(True)
plt.show()
```



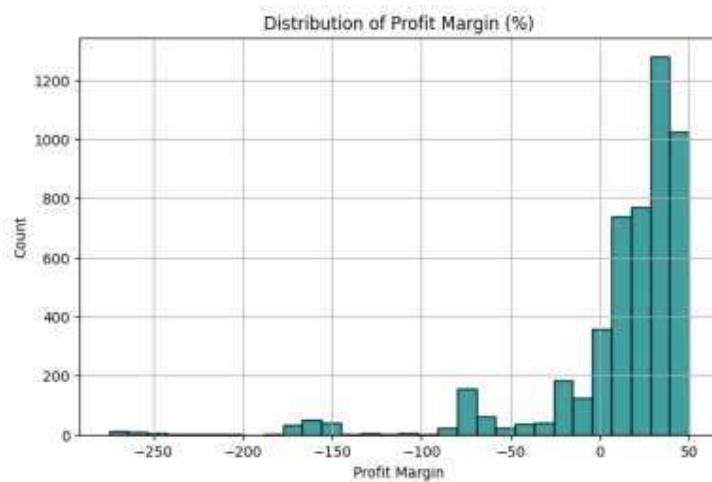
6. Profitability by Season & Category (Grouped Bar Chart)

```
season_cat = df.groupby(["Season", "Category"])["Profit"].sum().reset_index()
plt.figure(figsize=(10,6))
sns.barplot(data=season_cat, x="Season", y="Profit", hue="Category")
plt.title("Profit by Season and Category")
plt.ylabel("Total Profit")
plt.grid(True)
plt.tight_layout() plt.show()
```



7. Histogram: Distribution of Profit Margins

```
df["Profit Margin (%)"] = (df["Profit"] / df["Sales"]) * 100
plt.figure(figsize=(8,5))
sns.histplot(df["Profit Margin (%)"], bins=30, color="teal")
plt.title("Distribution of Profit Margin (%)") plt.xlabel("Profit Margin") plt.grid(True) plt.show()
```



8. Sub-Category Level Comparison (Bar Chart)

```
sub_profit = df.groupby("Sub-Category")["Profit"].sum().sort_values()
plt.figure(figsize=(10,6))
sns.barplot(x=sub_profit.values, y=sub_profit.index, palette="coolwarm")
plt.title("Total Profit by Sub-Category") plt.xlabel("Profit") plt.ylabel("Sub-Category")
plt.axvline(0, color="black", linestyle="--") plt.show()
```

