Alan Benny

```
# Importing the libraries
import pandas as pd
```

```
#Loading Dataset
data = pd.read_excel("/content/OFFLINE SALES - INDIRANAGAR (1).xlsx")
data.head()
```

	BARCODE	SKU NAME	QUANTITY	MRP	TOTAL	Mode	Total sales	CASH
0	602A3D75CD5FEA001A999147	Cadbury Perk Chocolate Bar 12 Gms	1.0	5.0	5.0	Prepaid	704.0	235.0
1	636B4355B80C7000136922CE	To Be Honest Tangy Chilli & Lime Chickpea 110 gms	1.0	67.5	67.5	Prepaid	NaN	NaN
		To Be Honest						

Next steps: Generate code with data View recommended plots

```
data.shape
```

```
(116, 9)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 116 entries, 0 to 115
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   BARCODE      24 non-null     object
 1   SKU   NAME   21 non-null     object
 2   QUANTITY     21 non-null     float64
 3   MRP          21 non-null     float64
 4   TOTAL        21 non-null     float64
 5   Mode         21 non-null     object
 6   Total sales  1 non-null      float64
 7   CASH         1 non-null      float64
 8   QR           1 non-null      float64
dtypes: float64(6), object(3)
memory usage: 8.3+ KB
```

```
data.columns
```

```
Index(['BARCODE', 'SKU   NAME', 'QUANTITY', 'MRP', 'TOTAL', 'Mode',
       'Total sales', 'CASH', 'QR'],
      dtype='object')
```

```
data.isnull().sum()
```

```
BARCODE         92
SKU   NAME      95
QUANTITY        95
MRP             95
TOTAL           95
Mode            95
Total sales    115
CASH           115
QR             115
dtype: int64
```

```
data.describe()
```

	QUANTITY	MRP	TOTAL	Total sales	CASH	QR
count	21.000000	21.000000	21.000000	1.0	1.0	1.0
mean	1.666667	33.523810	33.523810	704.0	235.0	469.0
std	1.110555	22.739545	22.739545	NaN	NaN	NaN
min	1.000000	5.000000	5.000000	704.0	235.0	469.0
25%	1.000000	20.000000	20.000000	704.0	235.0	469.0
50%	1.000000	20.000000	20.000000	704.0	235.0	469.0
75%	2.000000	50.000000	50.000000	704.0	235.0	469.0
max	5.000000	90.000000	90.000000	704.0	235.0	469.0

```
data.columns
```

```
Index(['BARCODE', 'SKU   NAME', 'QUANTITY', 'MRP', 'TOTAL', 'Mode',
       'Total sales', 'CASH', 'QR'],
      dtype='object')
```

```
# Fill missing values
# For numerical columns, fill with the median
numerical_columns = ['QUANTITY', 'MRP', 'TOTAL']
for col in numerical_columns:
    data[col].fillna(data[col].median(), inplace=True)


# For categorical columns, fill with the mode
categorical_columns = ['SKU   NAME', 'Mode', 'BARCODE']
for col in categorical_columns:
    data[col].fillna(data[col].mode()[0], inplace=True)
```

```
data.isnull().sum()
```

```
BARCODE          0
SKU   NAME       0
QUANTITY         0
MRP              0
TOTAL            0
Mode             0
Total sales    115
CASH           115
QR             115
dtype: int64
```

```
data.head(5)
```

	BARCODE	SKU NAME	QUANTITY	MRP	TOTAL	Mode	Total sales	CASH
0	602A3D75CD5FEA001A999147	Cadbury Perk Chocolate Bar 12 Gms	1.0	5.0	5.0	Prepaid	704.0	235.0
1	636B4355B80C7000136922CE	To Be Honest Tangy Chilli & Lime Chickpea 110 gms	1.0	67.5	67.5	Prepaid	NaN	NaN
		To Be Honest						

Next steps: Generate code with data View recommended plots

```
 #Convert data types
data['QUANTITY'] = data['QUANTITY'].astype(int)
data['MRP'] = data['MRP'].astype(float)
data['TOTAL'] = data['TOTAL'].astype(float)

# Confirm the data types
print("\nData types after conversion:")
print(data.dtypes)
```

```
Data types after conversion:
BARCODE         object
SKU   NAME      object
QUANTITY         int64
MRP            float64
TOTAL          float64
Mode            object
Total sales    float64
CASH           float64
QR             float64
dtype: object
```

```
# Remove unnecessary columns
columns_to_keep = ['BARCODE', 'SKU   NAME', 'QUANTITY', 'MRP', 'TOTAL', 'Mode']
df_cleaned = data[columns_to_keep]

# Inspect the cleaned dataset
print("\nCleaned and prepared dataset:")
df_cleaned.head()

# Save the cleaned dataset to a new CSV file
df_cleaned.to_csv('cleaned_daily_sales.csv', index=False)
```

```
Cleaned and prepared dataset:
```

```
df_cleaned.head()
```

	BARCODE	SKU NAME	QUANTITY	MRP	TOTAL	Mode
0	602A3D75CD5FEA001A999147	Cadbury Perk Chocolate Bar 12 Gms	1	5.0	5.0	Prepaid
1	636B4355B80C7000136922CE	To Be Honest Tangy Chilli & Lime Chickpea 110 gms	1	67.5	67.5	Prepaid
2	636B43520E1CE5001386D882	To Be Honest Purple Sweet Potato with Pani	1	60.0	60.0	Prepaid

Next steps: Generate code with `df_cleaned` View recommended plots

```
df_cleaned.isnull().sum()
```

```
BARCODE       0
SKU   NAME    0
QUANTITY      0
MRP           0
TOTAL         0
Mode          0
dtype: int64
```

- Top Performers: Identify the top-performing products and categories based on total sales and quantity sold

```
# Aggregate sales data to find total sales and quantity sold for each product
product_performance = df_cleaned.groupby('SKU   NAME').agg(
    total_sales=pd.NamedAgg(column='TOTAL', aggfunc='sum'),
    total_quantity_sold=pd.NamedAgg(column='QUANTITY', aggfunc='sum')
).reset_index()

# Sort and rank products based on total sales
top_products_by_sales = product_performance.sort_values(by='total_sales', ascending=False)

# Sort and rank products based on total quantity sold
top_products_by_quantity = product_performance.sort_values(by='total_quantity_sold', ascending=False)



# Display top 10 products by total sales
print("\nTop 10 products by total sales:")
top_products_by_sales.head(10)
```

```
Top 10 products by total sales:
```

	SKU NAME	total_sales	total_quantity_sold
4	Cadbury Perk Chocolate Bar 12 Gms	1910.0	97
3	Cadbury Dairy Milk Fruit & Nut Chocolate Bar 36g	90.0	2
16	To Be Honest Tangy Chilli & Lime Chickpea 110 gms	67.5	1
1	Bisleri Mineral Water 2 ltrs	60.0	2
15	To Be Honest Purple Sweet Potato with Pani Pur...	60.0	1
13	Maaza Mango Juice - Tetra Pack 135 ml	60.0	6
6	Coca cola zero sugar 250ml	60.0	3
0	Bingo! Original Style Chilli Sprinkled Potato ...	50.0	5
7	Coca-Cola 250 ml	40.0	2

Next steps: Generate code with top_products_by_sales View recommended plots

```
# Display top 10 products by total quantity sold
print("\nTop 10 products by total quantity sold:")
top_products_by_quantity.head(10)
```

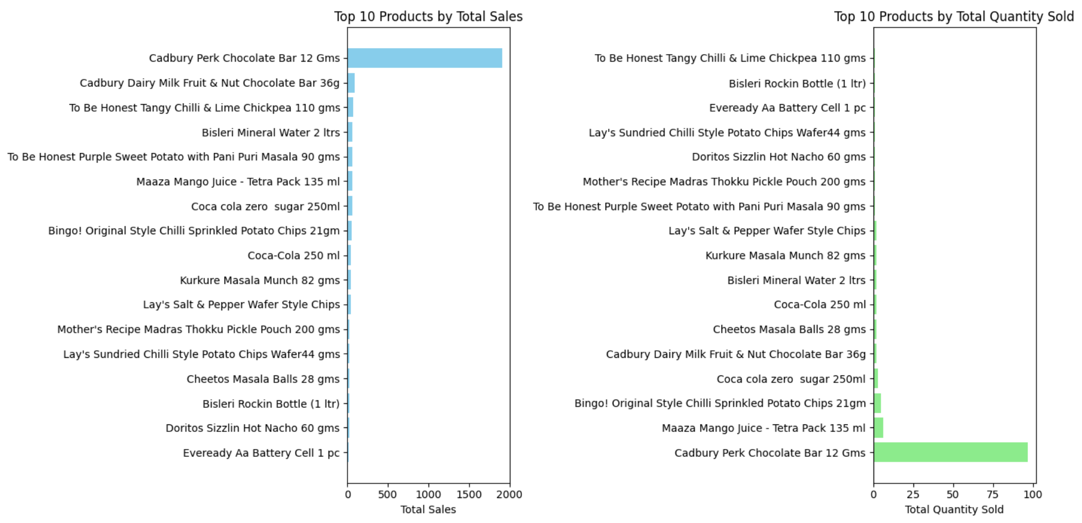
```
Top 10 products by total quantity sold:
```

	SKU NAME	total_sales	total_quantity_sold
4	Cadbury Perk Chocolate Bar 12 Gms	1910.0	97
13	Maaza Mango Juice - Tetra Pack 135 ml	60.0	6
0	Bingo! Original Style Chilli Sprinkled Potato ...	50.0	5
6	Coca cola zero sugar 250ml	60.0	3
3	Cadbury Dairy Milk Fruit & Nut Chocolate Bar 36g	90.0	2
5	Cheetos Masala Balls 28 gms	20.0	2
7	Coca-Cola 250 ml	40.0	2
1	Bisleri Mineral Water 2 ltrs	60.0	2
10	Kurkure Masala Munch 82 gms	40.0	2
11	Lay's Salt & Pepper Wafer Style Chips	40.0	2

Next steps: Generate code with top_products_by_quantity View recommended plots

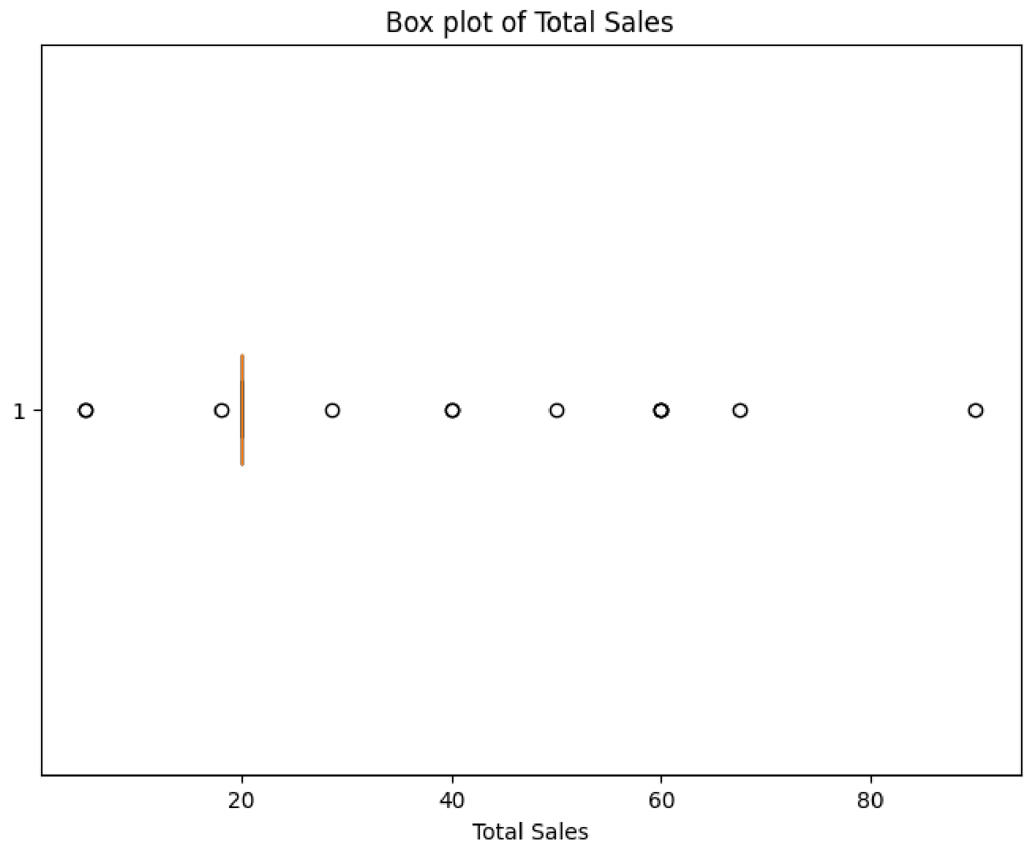
```
import matplotlib.pyplot as plt

# Create plots
plt.figure(figsize=(14, 7))

# Plot for top products by total sales
plt.subplot(1, 2, 1)
plt.barh(top_products_by_sales['SKU   NAME'], top_products_by_sales['total_sales'], color='skyblue')
plt.xlabel('Total Sales')
plt.title('Top 10 Products by Total Sales')
plt.gca().invert_yaxis()

# Plot for top products by total quantity sold
```

```
plt.subplot(1, 2, 2)
plt.barh(top_products_by_quantity['SKU   NAME'], top_products_by_quantity['total_quantity_sold'], color='ligl
plt.xlabel('Total Quantity Sold')
plt.title('Top 10 Products by Total Quantity Sold')

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```

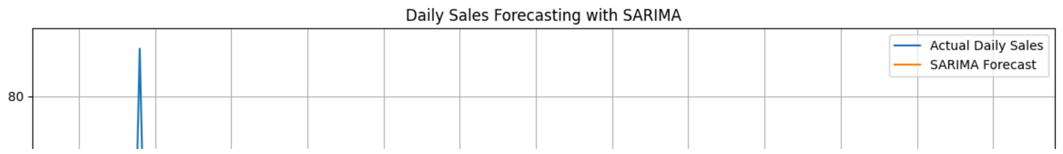


```
# Identify outliers in total sales using IQR method
Q1 = df_cleaned['TOTAL'].quantile(0.25)
Q3 = df_cleaned['TOTAL'].quantile(0.75)
IQR = Q3 - Q1

# Define outliers threshold
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter outliers
outliers = df_cleaned[(df_cleaned['TOTAL'] < lower_bound) | (df_cleaned['TOTAL'] > upper_bound)]

# Display outliers
print("Outliers in total sales:")
print(outliers)

# Visualize outliers using a box plot
plt.figure(figsize=(8, 6))
plt.boxplot(df_cleaned['TOTAL'], vert=False)
plt.title('Box plot of Total Sales')
plt.xlabel('Total Sales')
plt.show()
```

```
                                              SKU   NAME  QUANTITY   MRP  TOTAL  \
0                        Cadbury Perk Chocolate Bar 12 Gms         1   5.0    5.0
1    To Be Honest Tangy Chilli & Lime Chickpea 110 gms         1  67.5   67.5
2    To Be Honest Purple Sweet Potato with Pani Pur...         1  60.0   60.0
4    Mother's Recipe Madras Thokku Pickle Pouch 200...         1  28.5   28.5
5    Bingo! Original Style Chilli Sprinkled Potato ...         5  50.0   50.0
6                                    Coca-Cola 250 ml         2  40.0   40.0
8     Cadbury Dairy Milk Fruit & Nut Chocolate Bar 36g         2  90.0   90.0
10                          Bisleri Mineral Water 2 ltrs         2  60.0   60.0
11                  Maaza Mango Juice - Tetra Pack 135 ml         4  40.0   40.0
13                       Cadbury Perk Chocolate Bar 12 Gms         1   5.0    5.0
14                          Eveready Aa Battery Cell 1 pc         1  18.0   18.0
17                          Coca cola zero  sugar 250ml         3  60.0   60.0

       Mode
0   Prepaid
1   Prepaid
2   Prepaid
4   Prepaid
5   Prepaid
6   Prepaid
8       COD
10      COD
11      COD
13      COD
14  Prepaid
17  Prepaid
```



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Load the cleaned dataset
df_cleaned = pd.read_csv('cleaned_daily_sales.csv')

# Check the structure of df_cleaned
df_cleaned.head()  # Print first few rows to understand the structure
df_cleaned.tail()  # Print last few rows to understand the structure

# Fit SARIMA model
model = SARIMAX(df_cleaned['TOTAL'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 7))  # Example of SARIMA(1,
model_fit = model.fit()

# Forecast next steps (e.g., next 7 days)
forecast_steps = 7
forecast = model_fit.get_forecast(steps=forecast_steps)

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(df_cleaned.index, df_cleaned['TOTAL'], label='Actual Daily Sales')
plt.plot(np.arange(len(df_cleaned), len(df_cleaned) + forecast_steps), forecast.predicted_mean, label='SARI
plt.title('Daily Sales Forecasting with SARIMA')
plt.xlabel('Days')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(True)
plt.xticks(np.arange(0, len(df_cleaned) + forecast_steps, step=10), rotation=45)
plt.tight_layout()
plt.show()

# Forecast for the next week
last_day_sales = df_cleaned['TOTAL'].iloc[-1]
forecast_next_week = forecast.predicted_mean.iloc[-1]

print("\nForecast for the next week:")
print(f"Last day's sales: {last_day_sales}")
print(f"Forecasted sales for the next week: {forecast_next_week}")
```



Products Needing Reorder

Top Products by Total Sales:

- Cadbury Perk Chocolate Bar 12 Gms: Total sales of 1910 units
- Maaza Mango Juice - Tetra Pack 135 ml: Total sales of 60 units
- Bisleri Mineral Water 2 ltrs: Total sales of 60 units
- Coca cola zero sugar 250ml: Total sales of 60 units
- Bingo! Original Style Chilli Sprinkled Potato: Total sales of 50 units

These products have high total sales, indicating strong demand. If their current stock level is approaching 10 units or below, consider reordering to prevent stockouts.

Top Products by Total Quantity Sold:

- Cadbury Perk Chocolate Bar 12 Gms: Total quantity sold of 97 units
- Maaza Mango Juice - Tetra Pack 135 ml: Total quantity sold of 6 units
- Bingo! Original Style Chilli Sprinkled Potato: Total quantity sold of 5 units
- Coca cola zero sugar 250ml: Total quantity sold of 3 units
- Bisleri Mineral Water 2 ltrs: Total quantity sold of 2 units

These products have high quantities sold, indicating frequent customer purchases. Reevaluate their stock levels and reorder if necessary to maintain availability.

Identify products with declining sales that may require reduced purchase quantities:

Products with Declining Sales:

- To Be Honest Tangy Chilli & Lime Chickpea 110 gms: Sales of 67.5 units
- To Be Honest Purple Sweet Potato with Pani Puri Flavor: Sales of 60 units
- Coca-Cola 250 ml: Sales of 40 units
- Kurkure Masala Munch 82 gms: Sales of 40 units
- Lay's Salt & Pepper Wafer Style Chips: Sales of 40 units

Evaluate the sales trend for these products. If sales have been consistently declining or are below historical averages, consider reducing purchase quantities to avoid overstocking.