```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import seaborn as sns
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r"C:\Users\jahna\OneDrive\Desktop\Project-1\
med_inventory_dataset.csv")

# Display basic information about the DataFrame
print(df.info())

# Exploratory Data Analysis (EDA)
# Check summary statistics
print(df.describe())

# Check for missing values
print(df.isnull().sum())

# Visualize distribution of numerical features
sns.histplot(df['Quantity'], bins=20, kde=True)
plt.title('Distribution of Quantity')
plt.show()

# Visualize relationships between numerical features
sns.pairplot(df[['Quantity', 'Final_Cost', 'Final_Sales']])
plt.show()

# Data Cleaning
# Convert 'DateTime' to datetime type
df['DateTime'] = pd.to_datetime(df['Dateofbill'], errors='coerce')

# Drop rows with missing values
df.dropna(inplace=True)

# Convert numeric columns to numeric types
numeric_columns = ['Quantity', 'Final_Cost', 'Final_Sales']
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, errors='coerce')

# Data Modeling
# Split data into features (X) and target variable (y)
X = df[['Quantity', 'Final_Cost']]
y = df['Final_Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a simple RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
```

```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Feature Importance
feature_importance = model.feature_importances_
print('Feature Importance:')
for feature, importance in zip(X.columns, feature_importance):
    print(f'{feature}: {importance}')

df.to_csv("preprocessed_data.csv", index=False)

import pandas as pd

# Assuming df is your preprocessed DataFrame
# Convert 'DateTime' to datetime type if not already done
df['DateTime'] = pd.to_datetime(df['DateTime'], errors='coerce')

# Set 'DateTime' as the index
df_time_series = df.set_index('DateTime')

# Display the resulting DataFrame
print(df_time_series.head())
```