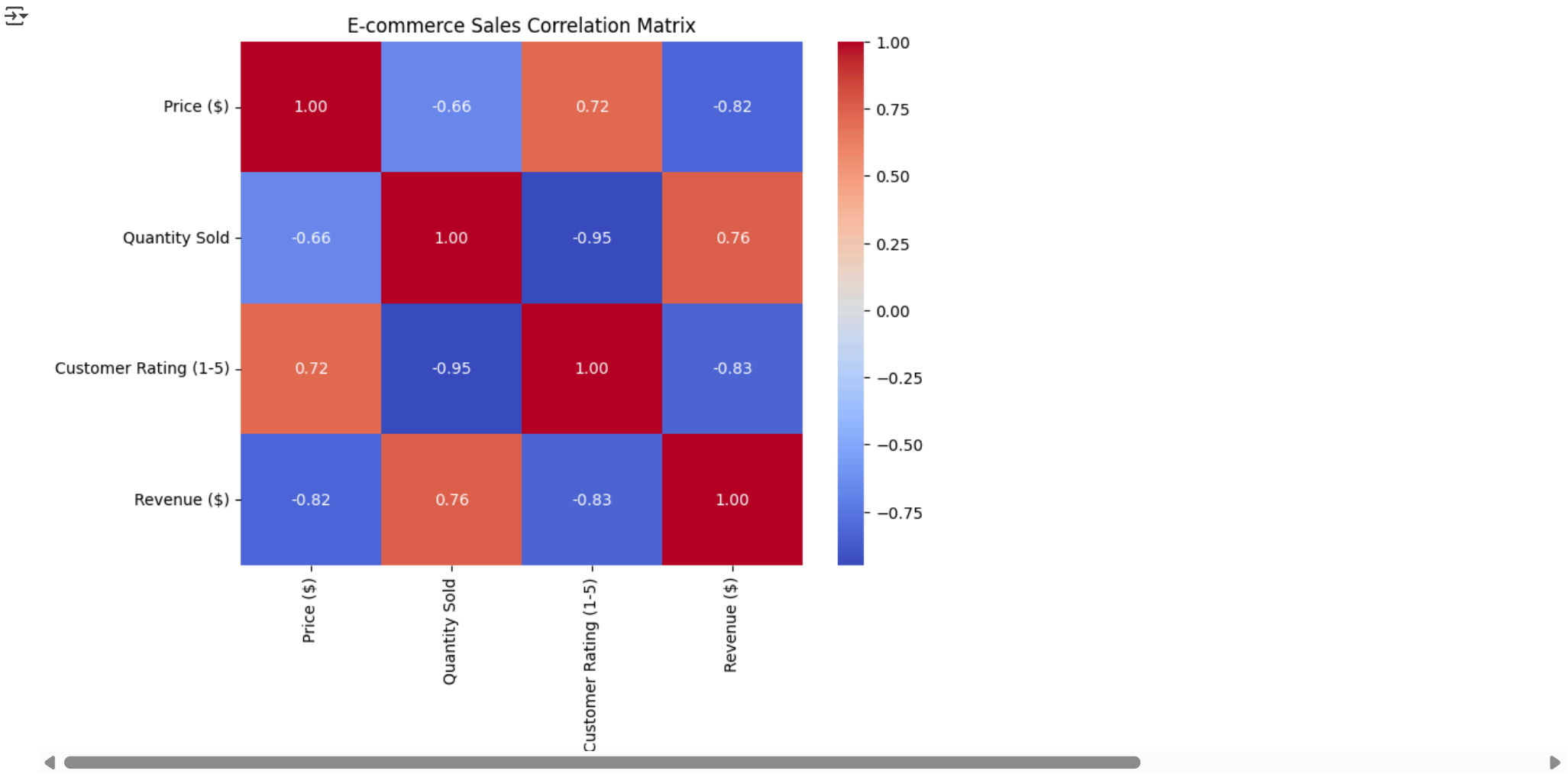
```
##CSV-formatted dataset for quantitative analysis, focusing on e-commerce sales with numeric variables (price, quantity sold, customer rating, and derived revenue).

import pandas as pd

df = pd.read_csv("D:\\LTU\\RM\\ecommerce_sales.csv")
print(df.describe())  # Summary stats (mean, min, max, etc.)
```

```
         Price ($)  Quantity Sold  Customer Rating (1-5)
count   10.000000      10.000000              10.000000
mean    60.490000     146.000000               4.330000
std     56.884386     154.124769               0.457165
min      9.990000      10.000000               3.500000
25%     22.490000      33.750000               4.050000
50%     44.990000     100.000000               4.400000
75%     74.990000     187.500000               4.675000
max    199.990000     500.000000               4.900000
```

Insight: High standard deviation in price and quantity suggests varied product types.

```
##Correlation Matrix (Seaborn)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv("D:\\LTU\\RM\\ecommerce_sales.csv").set_index('Product ID')

# Convert all columns that look like numbers with commas to proper numeric
for col in df.columns:
    df[col] = df[col].astype(str).str.replace(',', '')  # Remove commas
    df[col] = pd.to_numeric(df[col], errors='coerce')   # Convert to float, NaN if conversion fails

# Drop any columns or rows that are completely NaN (optional, depending on your data)
df = df.dropna(axis=1, how='all')
df = df.dropna(axis=0, how='all')

# Compute correlation matrix
corr_matrix = df.corr()

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("E-commerce Sales Correlation Matrix")
plt.show()
```




```
## Data Preparation

import pandas as pd

# Load data
df = pd.read_csv("D:\\LTU\\RM\\ecommerce_sales.csv")

# Clean data: remove commas from 'Revenue ($)' and convert to float
df['Revenue ($)'] = df['Revenue ($)'].str.replace(',', '', regex=False).astype(float)

# Optional: convert other numeric columns if needed
# df['Price ($)'] = df['Price ($)'].astype(float)
# df['Quantity Sold'] = df['Quantity Sold'].astype(int)
```

```
# df['Customer Rating (1-5)'] = df['Customer Rating (1-5)'].astype(float)

# Save cleaned data to a new file
output_path = "D:\\LTU\\RM\\ecommerce_sales_cleaned.csv"
df.to_csv(output_path, index=False)

print(f"✅ Cleaned data saved to: {output_path}")
```

```
✅ Cleaned data saved to: D:\LTU\RM\ecommerce_sales_cleaned.csv
```

```
#Normalization
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Load the cleaned dataset
input_path = "D:\\LTU\\RM\\ecommerce_sales_cleaned.csv"
df = pd.read_csv(input_path)

# Select columns to normalize
columns_to_normalize = ['Price ($)', 'Quantity Sold', 'Customer Rating (1-5)', 'Revenue ($)']

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Apply Min-Max normalization
df_normalized = df.copy()
df_normalized[columns_to_normalize] = scaler.fit_transform(df[columns_to_normalize])

# Save the normalized data
output_path = "D:\\LTU\\RM\\ecommerce_sales_normalized.csv"
df_normalized.to_csv(output_path, index=False)

print(f"✅ Normalized data saved to: {output_path}")
```

```
✅ Normalized data saved to: D:\LTU\RM\ecommerce_sales_normalized.csv
```

```
# Regression Analysis using Normalized Data

import pandas as pd
from sklearn.linear_model import LinearRegression

# Load normalized data
normalized_path = "D:\\LTU\\RM\\ecommerce_sales_normalized.csv"
df = pd.read_csv(normalized_path)

# Prepare variables
X = df[['Price ($)', 'Quantity Sold', 'Customer Rating (1-5)']]
y = df['Revenue ($)']
```

```
# Build regression model
model = LinearRegression()
model.fit(X, y)

# Get coefficients and intercept
coefficients = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_
})
intercept = model.intercept_

# Display results
print("✅ Regression Results (Normalized Data):")
print("Intercept:", intercept)
print(coefficients)

# Save results to file
output_path = "D:\\LTU\\RM\\regression_results_normalized.txt"
with open(output_path, 'w') as f:
    f.write("Multiple Linear Regression (Normalized Data)\n")
    f.write(f"Intercept: {intercept:.4f}\n")
    f.write("Coefficients:\n")
    for index, row in coefficients.iterrows():
        f.write(f"  {row['Feature']}: {row['Coefficient']:.4f}\n")

print(f"📄 Regression results saved to: {output_path}")
```

```
✅ Regression Results (Normalized Data):
Intercept: 1.2672765307640863
                  Feature  Coefficient
0             Price ($)    -0.571897
1          Quantity Sold    -0.273127
2  Customer Rating (1-5)    -0.830484
📄 Regression results saved to: D:\LTU\RM\regression_results_normalized.txt
```

```
#Multiple Linear regression
import pandas as pd
from sklearn.linear_model import LinearRegression

# Load normalized data
input_path = "D:\\LTU\\RM\\ecommerce_sales_normalized.csv"
df = pd.read_csv(input_path)

# Prepare independent (X) and dependent (y) variables
X = df[['Price ($)', 'Quantity Sold', 'Customer Rating (1-5)']]
y = df['Revenue ($)']

# Build and fit model
model = LinearRegression()
model.fit(X, y)
```

```
# Apply regression formula to forecast revenue
df['Predicted Revenue ($)'] = model.predict(X)

# Save the file with predicted revenue
output_path = "D:\\LTU\\RM\\ecommerce_sales_forecasted.csv"
df.to_csv(output_path, index=False)

print(f"✅ Forecasted revenue added and saved to: {output_path}")
```

```
✅ Forecasted revenue added and saved to: D:\LTU\RM\ecommerce_sales_forecasted.csv
```

```
#Denormalize
import pandas as pd

# Load forecasted normalized data
df_norm = pd.read_csv("D:\\LTU\\RM\\ecommerce_sales_forecasted.csv")

# Original min and max values (replace these with actual from Step 1)
min_max = {
    'Price ($)': (9.99, 199.99),
    'Quantity Sold': (10, 500),
    'Customer Rating (1-5)': (3.5, 4.9),
    'Revenue ($)': (1999.90, 4995.00)
}

# Function to denormalize a column
def denormalize(col_name):
    min_val, max_val = min_max[col_name]
    return df_norm[col_name] * (max_val - min_val) + min_val

# Denormalize each column
for col in ['Price ($)', 'Quantity Sold', 'Customer Rating (1-5)', 'Revenue ($)']:
    df_norm[f"{col}_Original"] = denormalize(col)

# Denormalize predicted revenue too
df_norm['Predicted Revenue ($)_Original'] = df_norm['Predicted Revenue ($)'] * (min_max['Revenue ($)'][1] - min_max['Revenue ($)'][0]) + min_max['Revenue ($)'][0]

# Save denormalized data
output_path = "D:\\LTU\\RM\\ecommerce_sales_forecasted_denormalized_all.csv"
df_norm.to_csv(output_path, index=False)

print(f"✅ All columns denormalized and saved to: {output_path}")
```

```
✅ All columns denormalized and saved to: D:\LTU\RM\ecommerce_sales_forecasted_denormalized_all.csv
```

```
Start coding or generate with AI.
```