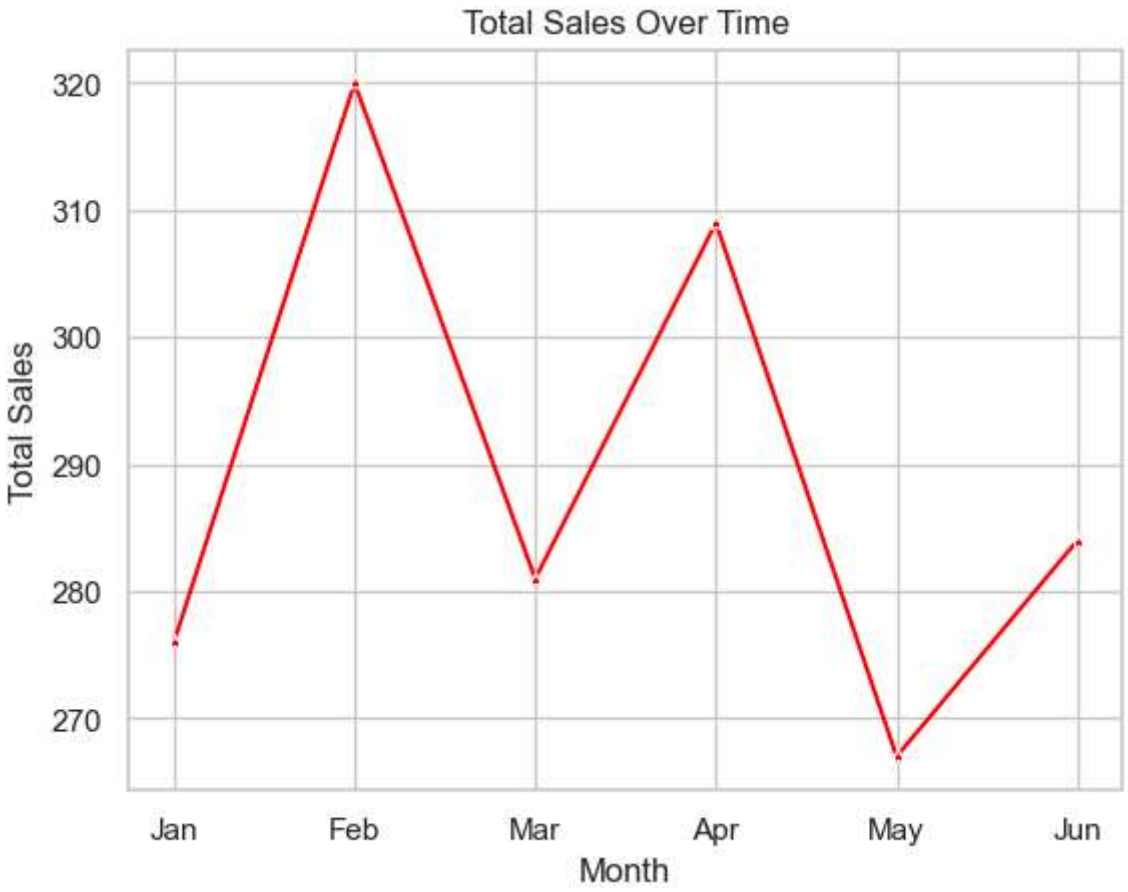
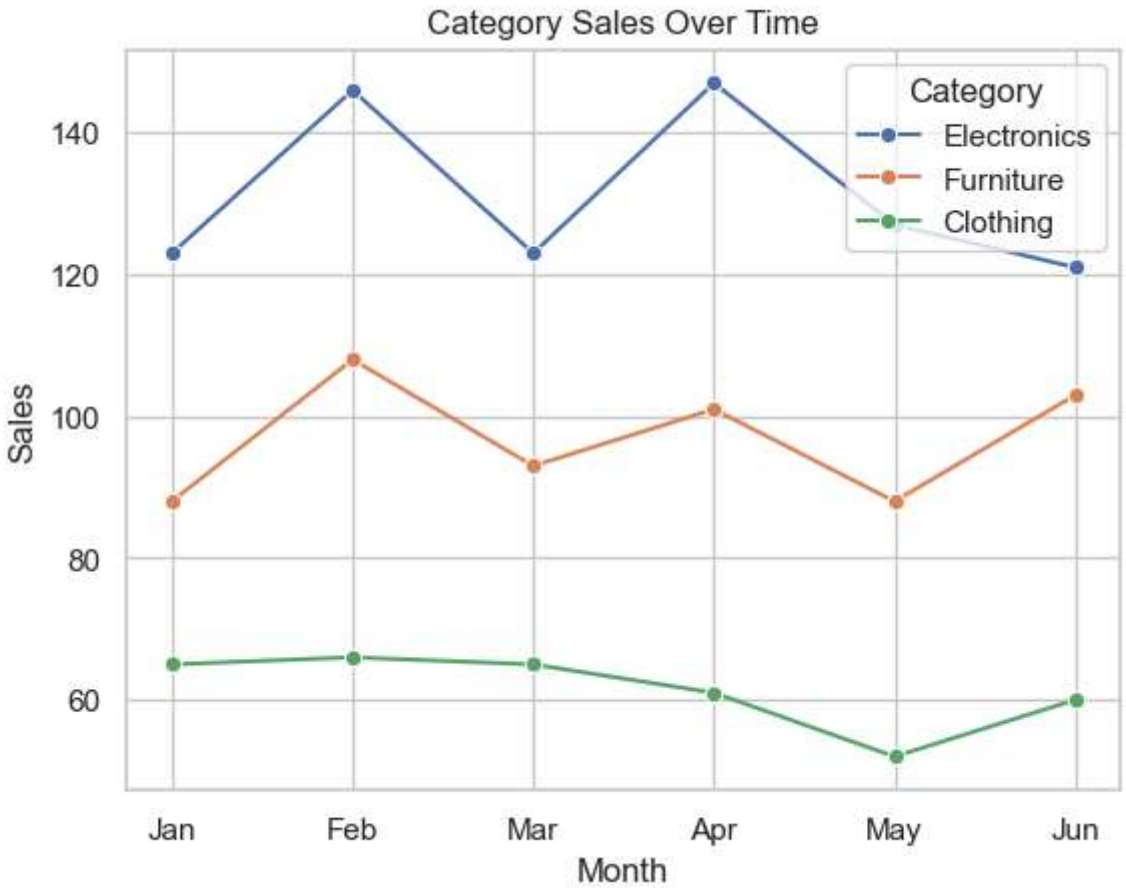
In [1]:
```
# ----------------------------------------
# DO NOT EDIT THIS CELL
# ----------------------------------------
# Load required packages (install if needed before running)
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import random

# Set random seed for reproducibility
np.random.seed(42)
random.seed(42)
```

In [11]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 1
# ----------------------------------------
# Do not alter the code below, which creates a dataframe from which you will create
# Create a line plot using category_df that displays sales over time (using Total_S

# DO NOT ALTER THE CODE HERE
# The code below generates synthetic data into an object 'category_df'
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']

electronics_sales = np.random.randint(120, 150, size=6)
furniture_sales = np.random.randint(80, 110, size=6)
clothing_sales = np.random.randint(50, 70, size=6)

category_df = pd.DataFrame({
    'Electronics': electronics_sales,
    'Furniture': furniture_sales,
    'Clothing': clothing_sales,
    'Month': months
})

category_df['Total_Sales'] = category_df[['Electronics', 'Furniture', 'Clothing']].



# PLACE YOUR CODE HERE
sns.set_theme(style="whitegrid")
sns.lineplot(data=category_df, x='Month', y='Total_Sales', marker='*', color = "red
plt.title("Total Sales Over Time")
plt.xlabel("Month")
plt.ylabel("Total Sales")
plt.show()
```

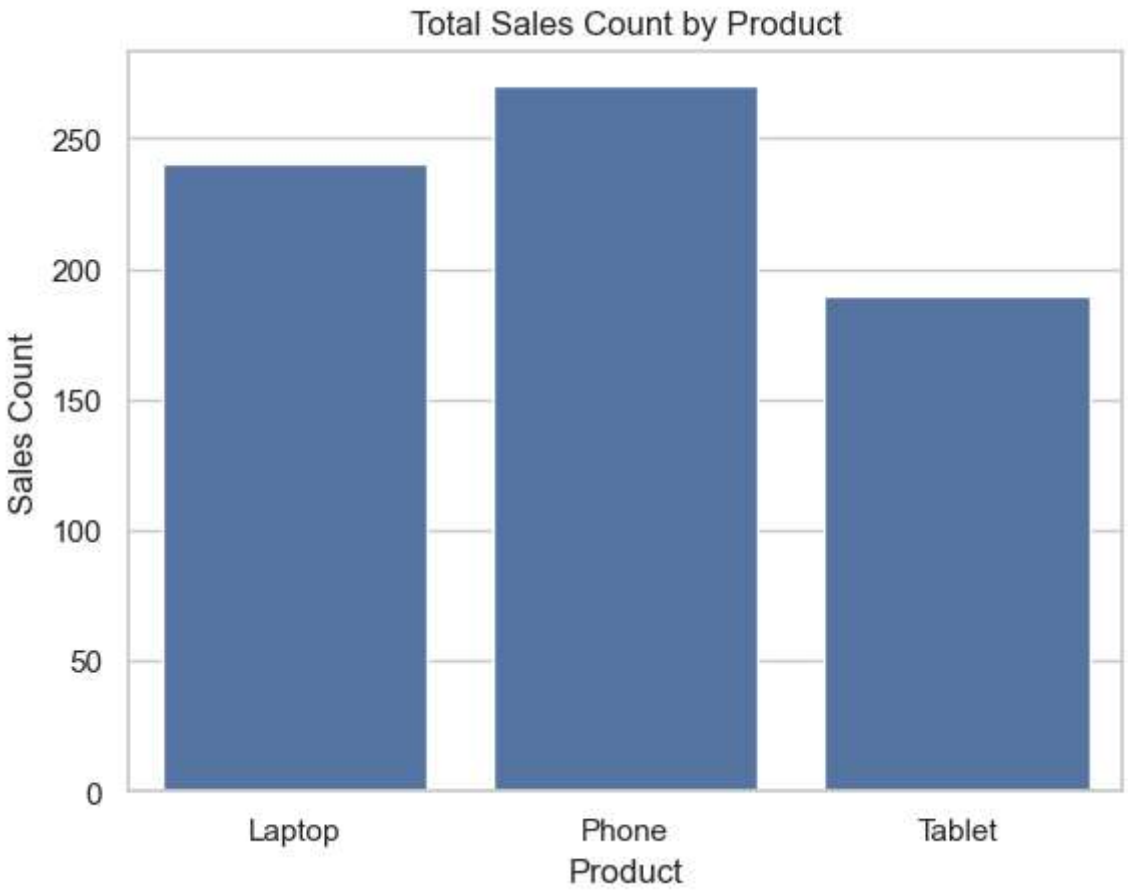


In [ ]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 2
# ----------------------------------------
# Do not alter the code below, which creates a dataframe from which you will create
# Create a line plot that displays a distinct line for sales over time for each Cat
# Each line should be a different color and use Sales and Month for the y and x axe

melted_category_df = pd.melt(category_df, id_vars=['Month'], value_vars=['Electroni
                             var_name='Category', value_name='Sales')




# PLACE YOUR CODE HERE
sns.set_theme(style="whitegrid")

# Line plot for each category's sales over time
sns.lineplot(data=melted_category_df, x='Month', y='Sales', hue='Category', marker=
plt.title("Categorical Sales Over Time")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.show()
```

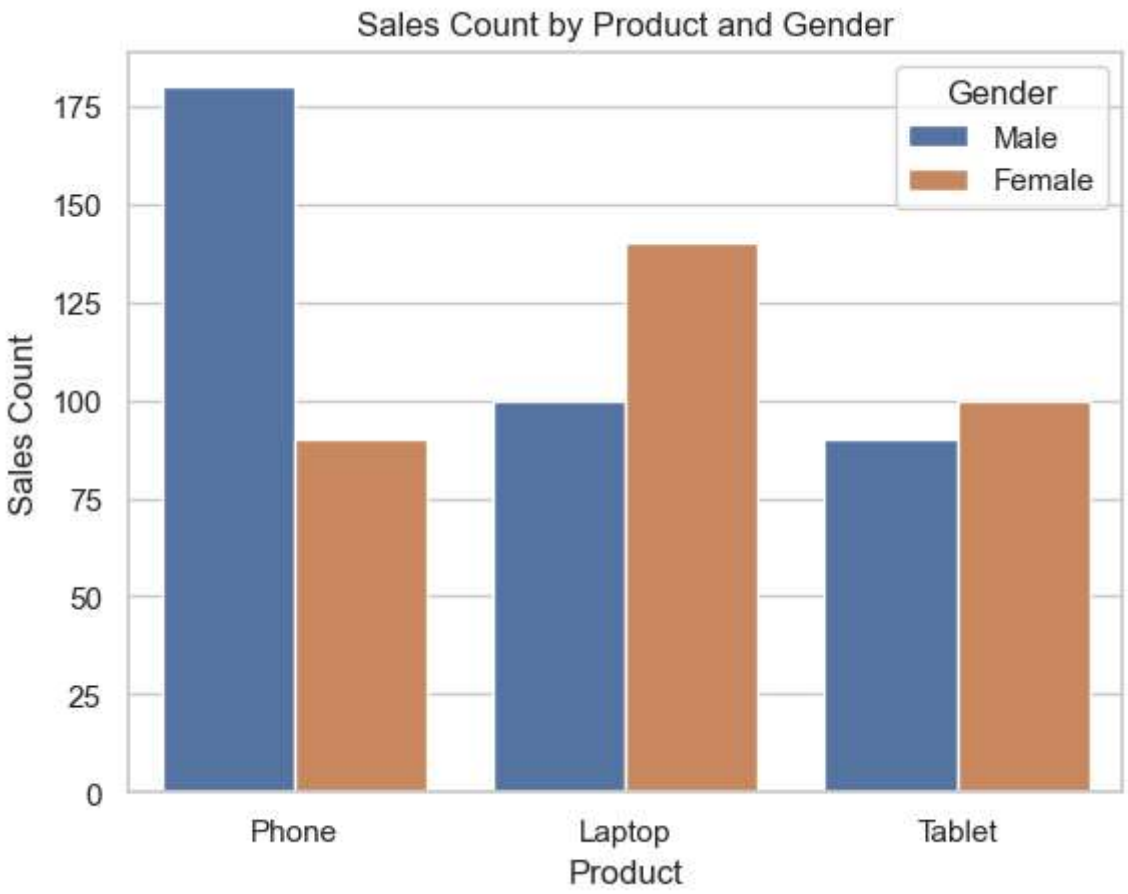


In [13]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 3
# ----------------------------------------
# Do not alter the code below, which creates a dataframe called 'simple_bar_df' fro
# Create a vertical bar plot that displays a sales count for each Product

product_types = ['Phone', 'Laptop', 'Tablet']

male_counts = np.array([180, 100, 90])
female_counts = np.array([90, 140, 100])

interaction_df = pd.DataFrame({
    'Product': product_types * 2,
    'Gender': ['Male']*3 + ['Female']*3,
    'Count': np.concatenate([male_counts, female_counts])
})

simple_bar_df = interaction_df.groupby('Product')['Count'].sum().reset_index()



# PLACE YOUR CODE HERE
sns.set_theme(style="whitegrid")
sns.barplot(data=simple_bar_df, x='Product', y='Count')
plt.title("Total Sales Count by Product")
plt.xlabel("Product")
```

```
plt.ylabel("Sales Count")
plt.show()
```

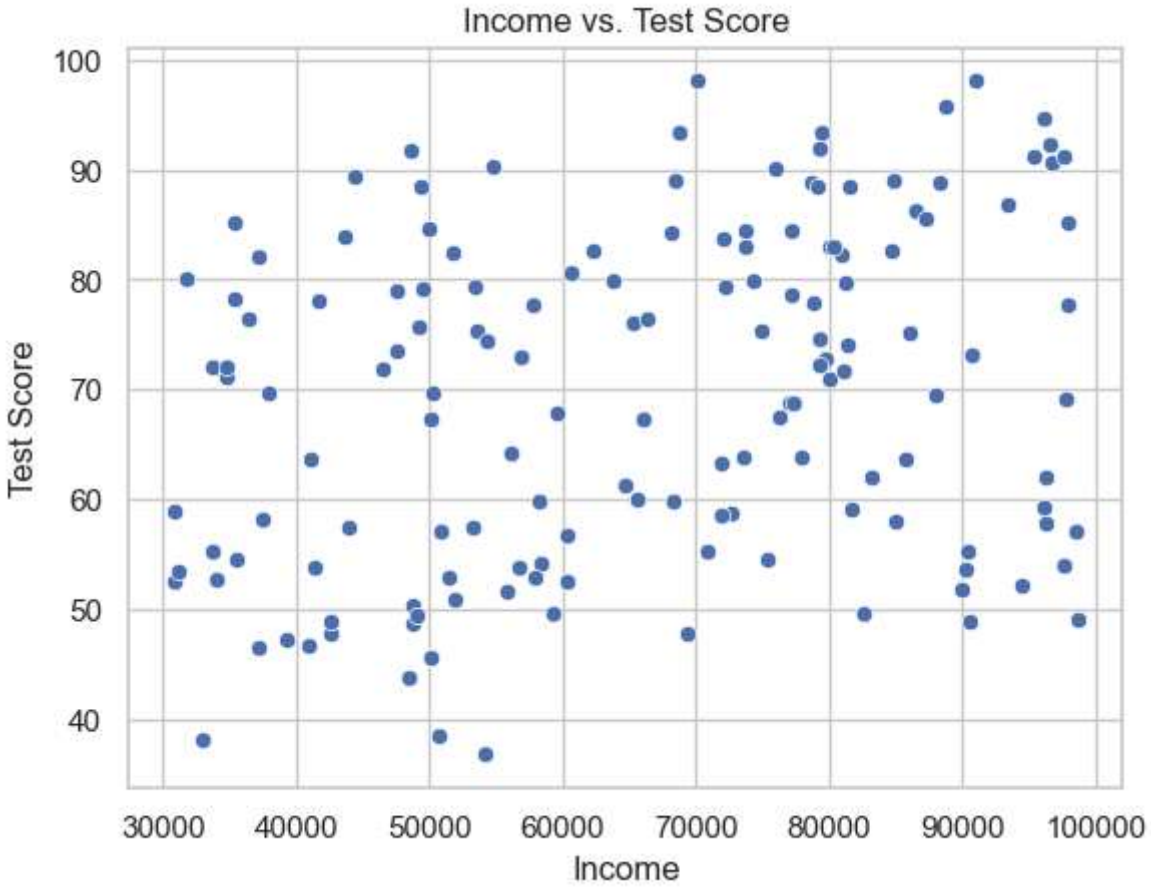


In [14]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 4
# ----------------------------------------
# Create a grouped bar plot that displays sales counts for each product type, where
# Use the interaction_df object from the prior code cell for convenience



# PLACE YOUR CODE HERE

sns.set_theme(style="whitegrid")
sns.barplot(data=interaction_df, x='Product', y='Count', hue='Gender')
plt.title("Sales Count by Product and Gender")
plt.xlabel("Product")
plt.ylabel("Sales Count")
plt.show()
```



In [ ]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 5
# ----------------------------------------
# Do not alter the code below, which creates a dataframe from which you will create
# Create a scatter plot that displays Income (x-axis) vs. Test_Score (y-axis), usin

n = 150
income = np.random.uniform(30_000, 100_000, n)

school_type = np.random.choice(['A', 'B', 'C'], size=n, p=[0.4, 0.3, 0.3])

test_score = []
for school in school_type:
    if school == 'A':
        test_score.append(np.random.normal(54, 6))
    elif school == 'B':
        test_score.append(np.random.normal(85, 6))
    else:
        test_score.append(np.random.normal(75, 6))

test_score = np.array(test_score) + (income - income.mean()) / 10_000

study_time = 20 + (income/100_000) * 80 + np.random.normal(0, 5, n)

scatter_df = pd.DataFrame({
    'Income': income,
    'Test_Score': test_score,
    'School_Type': school_type,
```

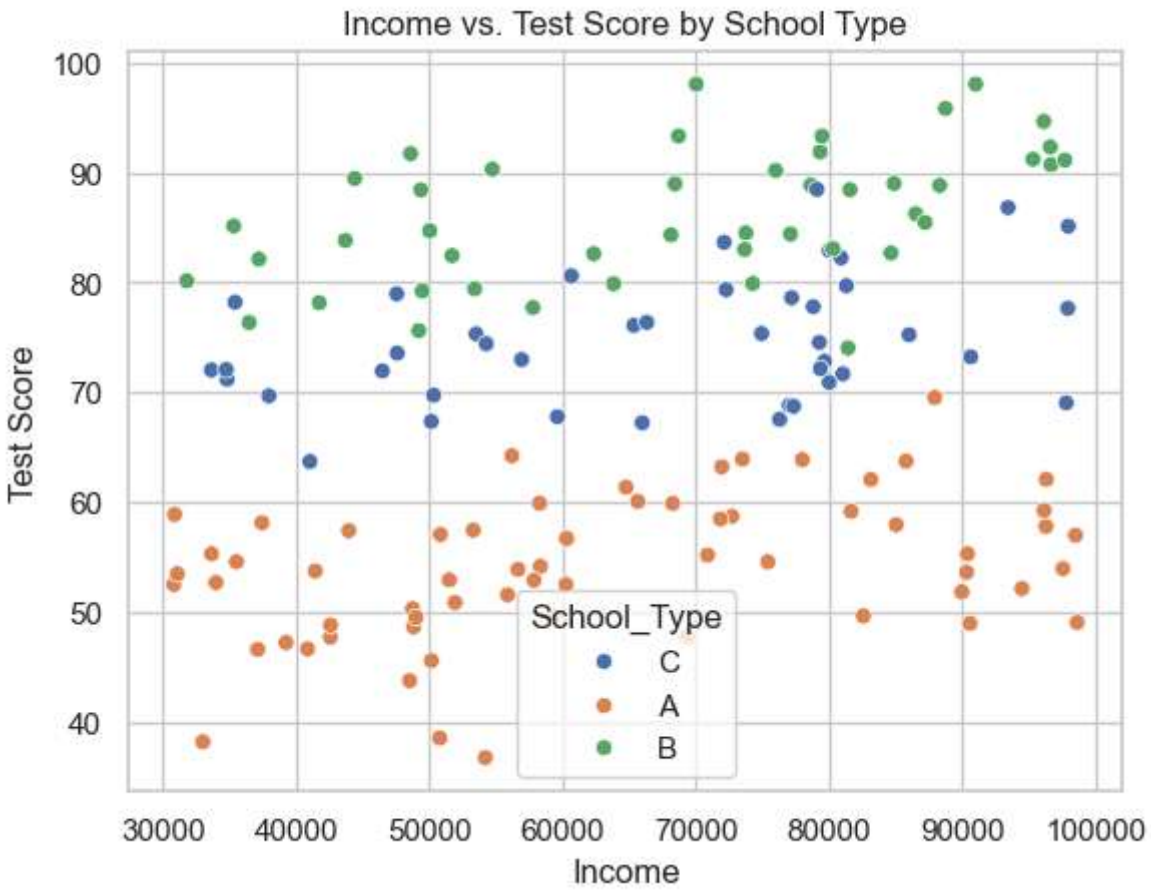
```
    'Study_Time': study_time
})



# PLACE YOUR CODE HERE

sns.set_theme(style="whitegrid")
sns.scatterplot(data=scatter_df, x='Income', y='Test_Score')
plt.title("Income vs. Test Score")
plt.xlabel("Income")
plt.ylabel("Test Score")
plt.show()
```



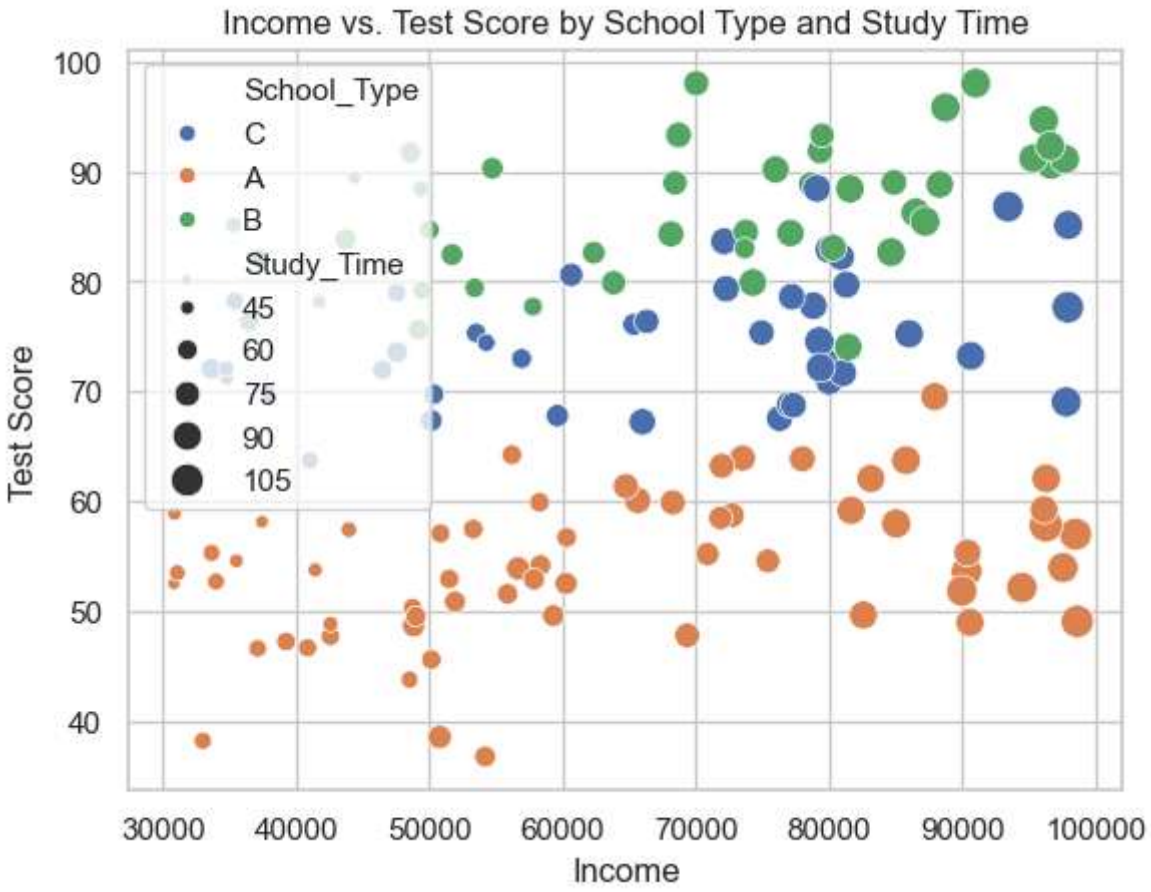
In [ ]:

```
# ----------------------------------------
# INSTRUCTIONS: TASK 6
# ----------------------------------------
# Create a scatter plot that displays income (x-axis) vs. test scores (y-axis), and
# Again, use 'scatter_df'.



# PLACE YOUR CODE HERE

sns.set_theme(style="whitegrid")
sns.scatterplot(data=scatter_df, x='Income', y='Test_Score', hue='School_Type')
plt.title("Income vs. Test Score by School Type")
plt.xlabel("Income")
```

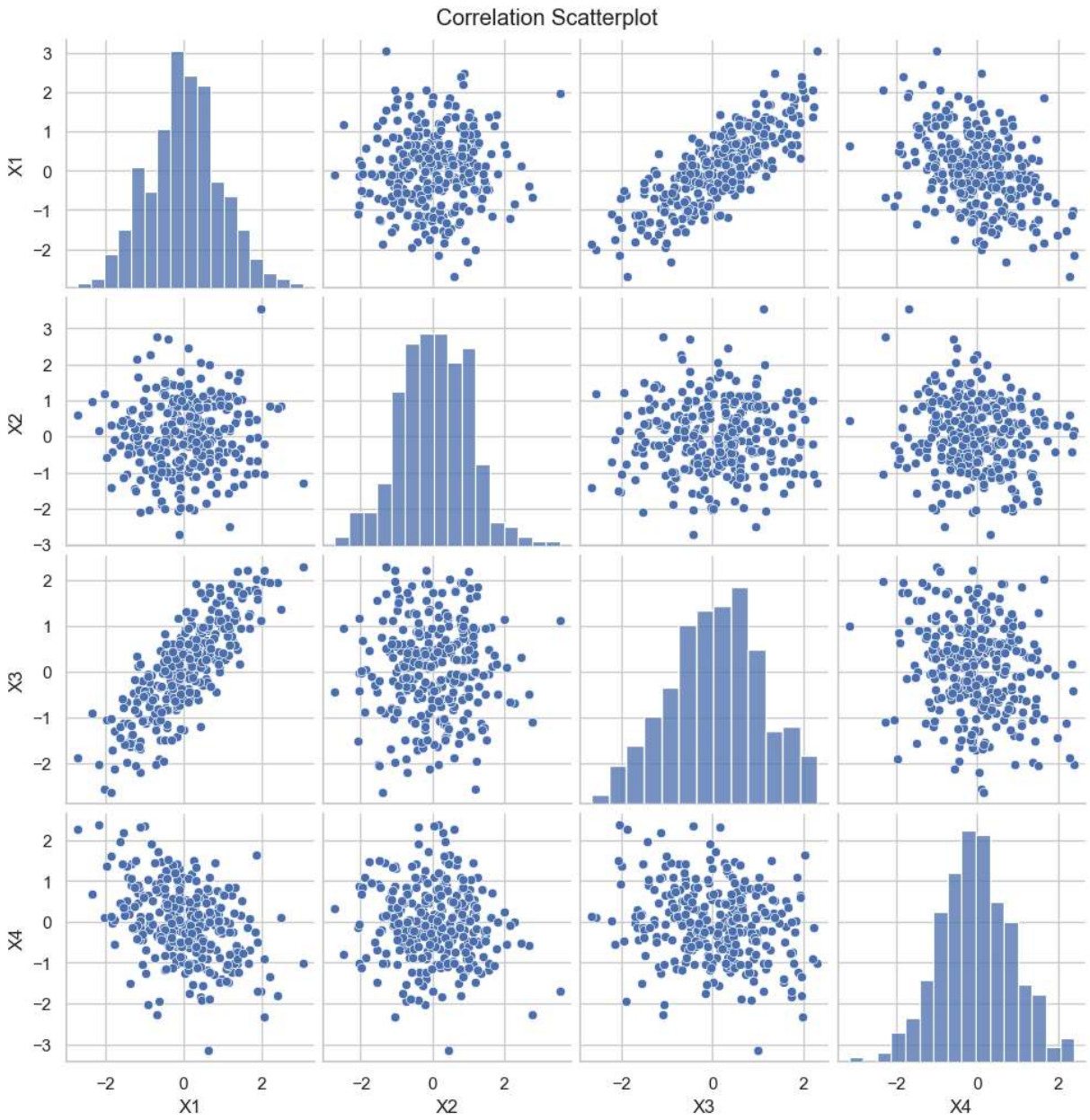
```
plt.ylabel("Test Score")
plt.show()
```



In [27]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 7
# ----------------------------------------
# Create a scatter plot that displays income (x-axis) vs. test scores (y-axis),
# and also varies color of points by school type, and varies size by study time
# Points should become larger as Study_Time increases
# Again, use 'scatter_df'.


# PLACE YOUR CODE HERE
sns.set_theme(style="whitegrid")
sns.scatterplot(data=scatter_df, x='Income', y='Test_Score', hue='School_Type', siz
plt.title("Income vs. Test Score by School Type and Study Time")
plt.xlabel("Income")
plt.ylabel("Test Score")
plt.show()
```



In [33]:
```
# ----------------------------------------
# INSTRUCTIONS: TASK 8
# ----------------------------------------
# Do not alter the code below, which creates a dataframe from which you will create
# Create a pairs plot that displays pairwise scatterplots of X1, X2, X3, and X4.
# Use the pairplot function in Seaborn and the pairs_df object to construct this pl

cov_matrix = [
    [1,  0.0,  0.8, -0.5],  # Correlations for X1
    [0.0, 1,   0.0,  0.0],  # Correlations for X2
    [0.8, 0.0, 1,  -0.3],   # Correlations for X3
    [-0.5, 0.0, -0.3, 1]    # Correlations for X4
]

mean = [0, 0, 0, 0]
n_samples = 300

multi_data = np.random.multivariate_normal(mean, cov_matrix, size=n_samples)

pairs_df = pd.DataFrame(multi_data, columns=['X1', 'X2', 'X3', 'X4'])



# PLACE YOUR CODE HERE
sns.set_theme(style="whitegrid")
sns.pairplot(pairs_df)
plt.suptitle("Correlation Scatterplot", y=1.01)
plt.show()
```


Correlation Scatterplot
X1
X2
X3
X4