Technological Institute of the Philippines

Computer Engineering Department

Quezon city Campus

Midterm Skills Exam: Data Wrangling and Analysis

Course: CPE 311	Program: BSCpE
Course Title: Computational Thinking with Python	**Date Performed:** April 13 , 2024
Section: BSCPE22S3	**Date Submitted:** April 13, 2024
Student Name: Juliann Vincent B. Quibral	**Instructor's Name:** Engr. Roman Richard

Data Import

```
1 pip install ucimlrepo

  Collecting ucimlrepo
    Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
  Installing collected packages: ucimlrepo
  Successfully installed ucimlrepo-0.0.6
```

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from ucimlrepo import fetch_ucirepo
```

```
1 census_income = fetch_ucirepo(id=20)
```

```
1 X = census_income.data.features
2 y = census_income.data.targets
```

Initial Exploration

```
1 X.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	ra
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Wl
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Wl
							Handlers-		

Next steps: ◯ **View recommended plots**

```
1 y.head()
```

	income
0	<=50K
1	<=50K
2	<=50K
3	<=50K
4	<=50K

Next steps: ◯ **View recommended plots**

⌄ Data Concatenation

```
1 df = pd.concat([X, pd.DataFrame(y, columns=['income'])], axis=1)
```

```
1 print("Data Imported. Shape of DataFrame:", df.shape)
```

```
   Data Imported. Shape of DataFrame: (48842, 15)
```

```
1 # Save DataFrame to CSV file
2 df.to_csv('Census.csv', index=False)
```
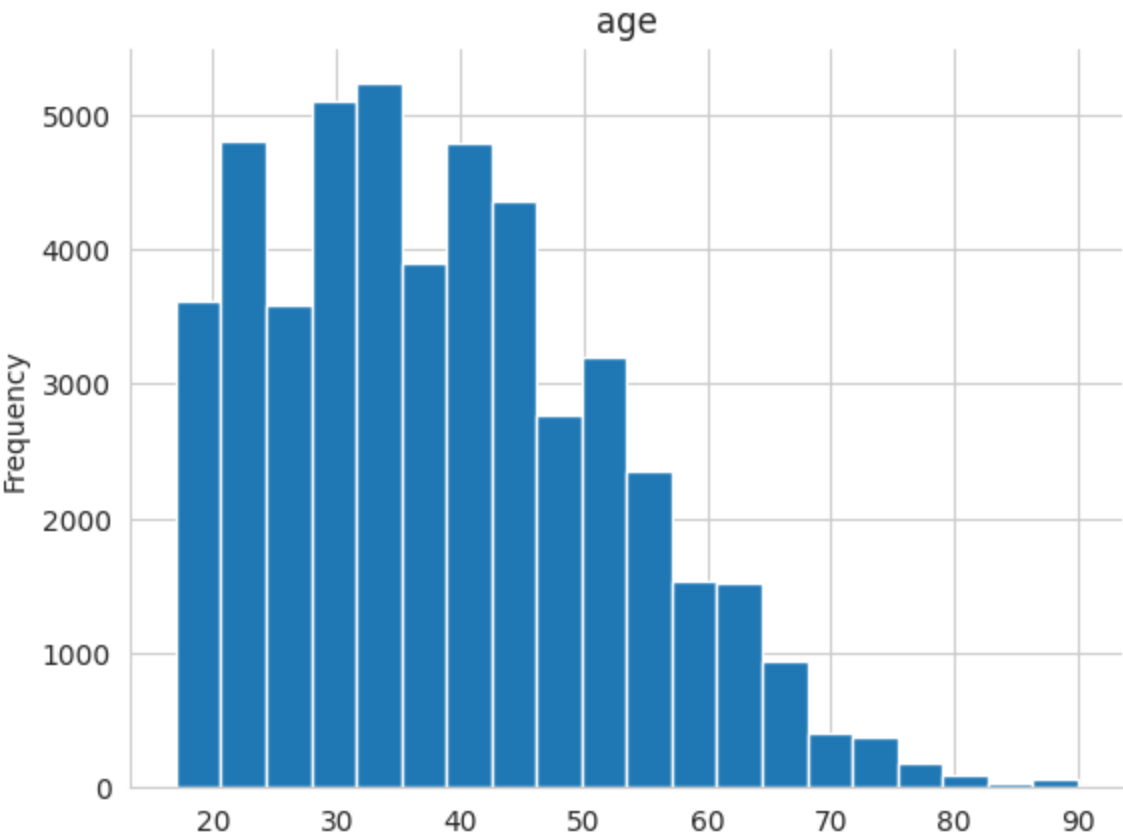
```
1 df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	ra
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Wh
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Wh

Next steps: | ◯ **View recommended plots** |

> age

Show code

age

Summary of the repondents age.

∨ **Check or missing values**

```
1 # Check for missing values
2 missing_values = df.isnull().sum()
3 print("Missing Values:\n", missing_values)
```

```
Missing Values:
 age                  0
 workclass          963
 fnlwgt               0
 education            0
 education-num        0
 marital-status       0
 occupation         966
 relationship         0
 race                 0
 sex                  0
 capital-gain         0
 capital-loss         0
 hours-per-week       0
 native-country     274
 income               0
 dtype: int64
```

```
1 # Fill missing values
2 df.fillna(method='ffill', inplace=True)  # Forward fill missing values
```

```
1 #Recheck for missing values
2 missing_values = df.isnull().sum()
3 print("Missing Values:\n", missing_values)
```

```
Missing Values:
 age                0
 workclass          0
 fnlwgt             0
 education          0
 education-num      0
 marital-status     0
 occupation         0
 relationship       0
 race               0
 sex                0
 capital-gain       0
 capital-loss       0
 hours-per-week     0
 native-country     0
 income             0
 dtype: int64
```

Check for duplicate values

```
1 # Checking for duplicate values
2 duplicates = df.duplicated().sum()
3 print("\nNumber of Duplicate Rows:", duplicates)
```
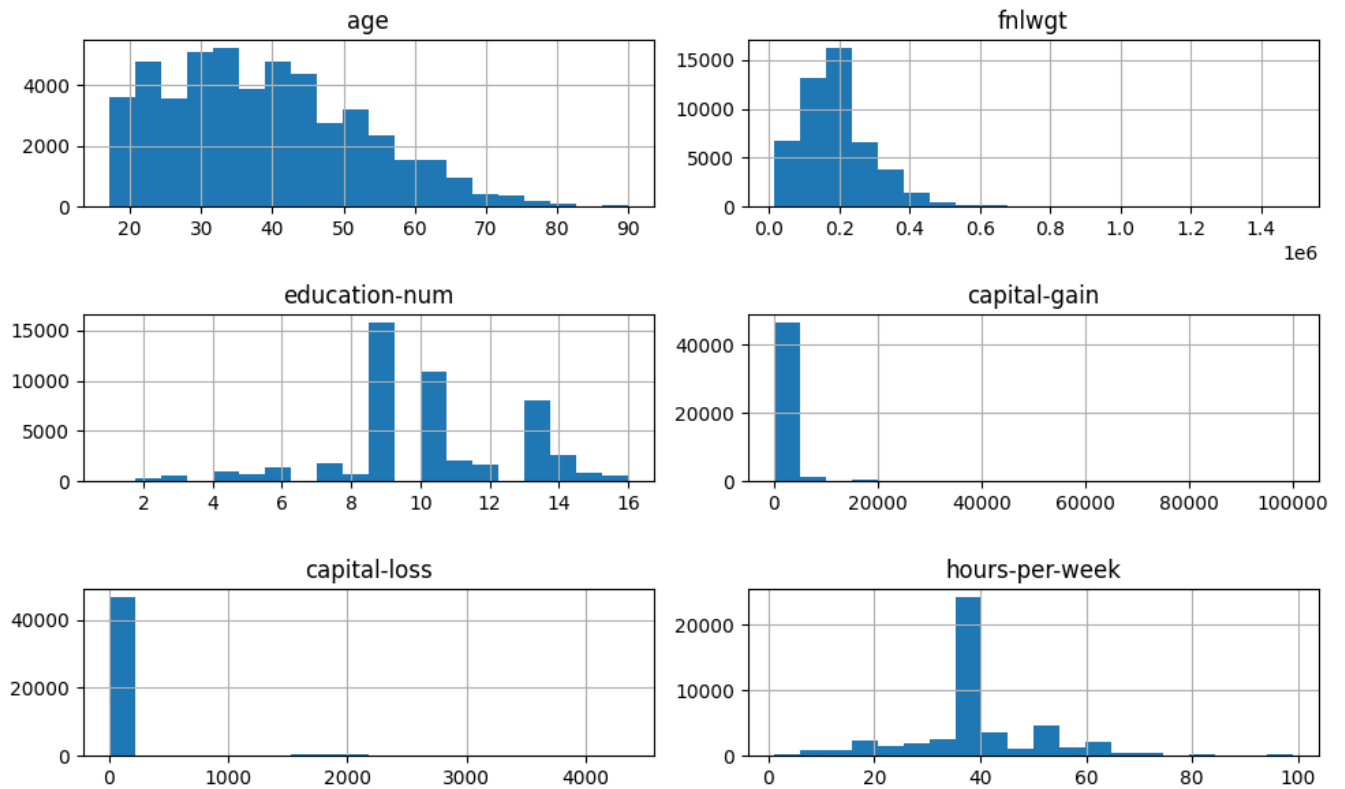
    Number of Duplicate Rows: 29


```
1 # Drop duplicate rows
2 df.drop_duplicates(inplace=True)
```


```
1 #Rechecking for duplicate values
2 duplicates = df.duplicated().sum()
3 print("\nNumber of Duplicate Rows:", duplicates)
```

    Number of Duplicate Rows: 0

∨ Initial Plot


```
1 import matplotlib.pyplot as plt
2
3 df.hist(bins=20, figsize=(10, 6))
4 plt.tight_layout()
5 plt.show()
```
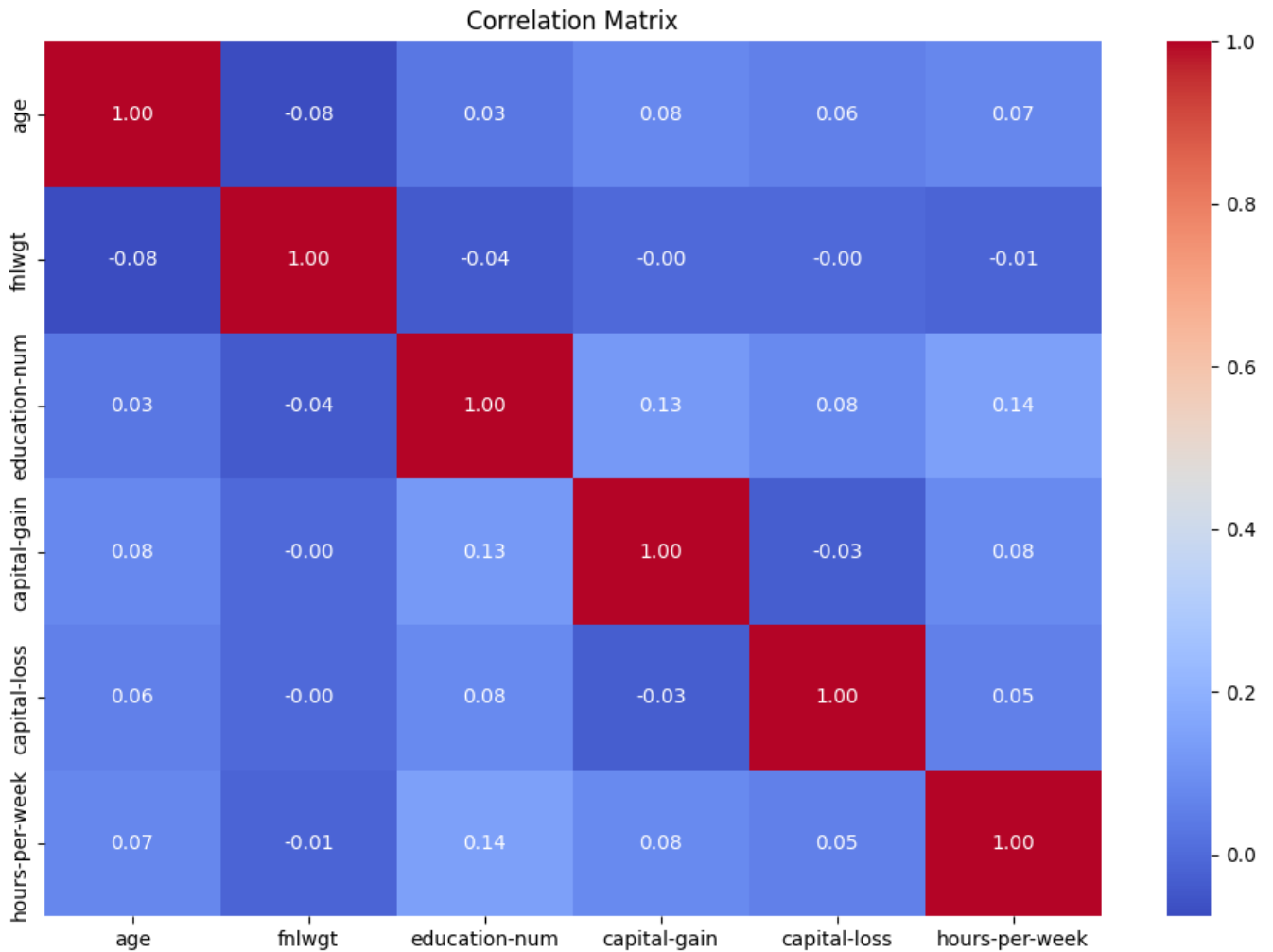
Correlation Matrix

```
1 # Correlation matrix (excluding non-numeric columns)
2 correlation_matrix = df.select_dtypes(include=np.number).corr()
3
4 plt.figure(figsize=(12, 8))
5 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
6 plt.title('Correlation Matrix')
7 plt.show()
```
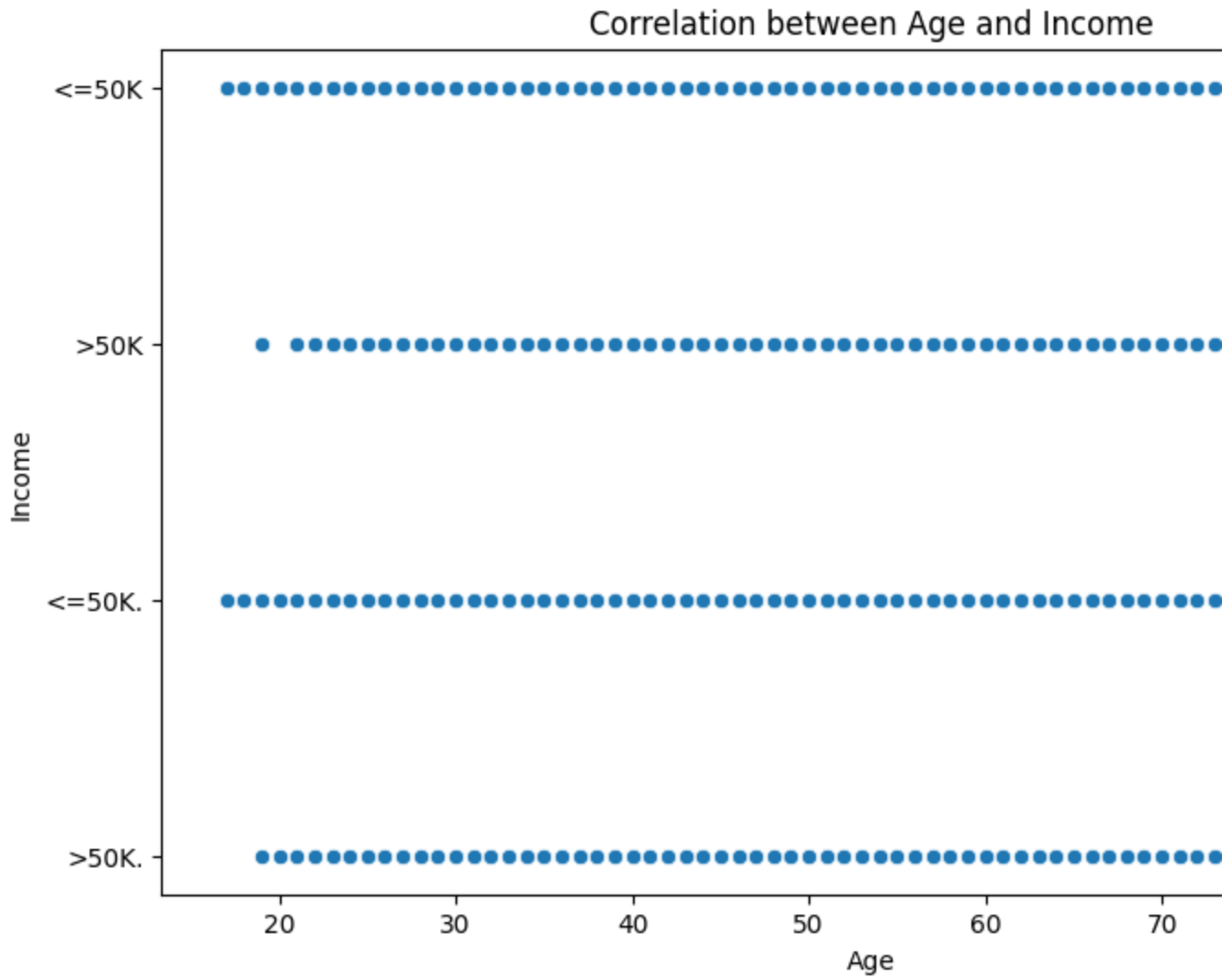
Correlation Matrix

Data Correlation/Comparions Plotting

```
1 plt.figure(figsize=(10, 6))
2
3 sns.scatterplot(x='age', y='income', data=df)
4 plt.title('Correlation between Age and Income')
5 plt.xlabel('Age')
6 plt.ylabel('Income')
7 plt.show()
```

Correlation between Age and Income:

Description: This comparison explores how age correlates with income levels. It can reveal whether there is a trend of increasing or decreasing income with age, which is important for understanding age-related income dynamics, workforce participation, and retirement planning.

```
 1 plt.figure(figsize=(10, 6))
 2
 3 sns.barplot(x='workclass', y='income_numeric', data=df, order=df['workclass'].value_count
 4 plt.title('Comparison of Workclass and Income')
 5 plt.xlabel('Workclass')
 6 plt.ylabel('Average Income')
 7 plt.xticks(rotation=45)
 8 plt.grid(axis='y')
 9 plt.show()
10
```

## Comparison of Workclass and Income
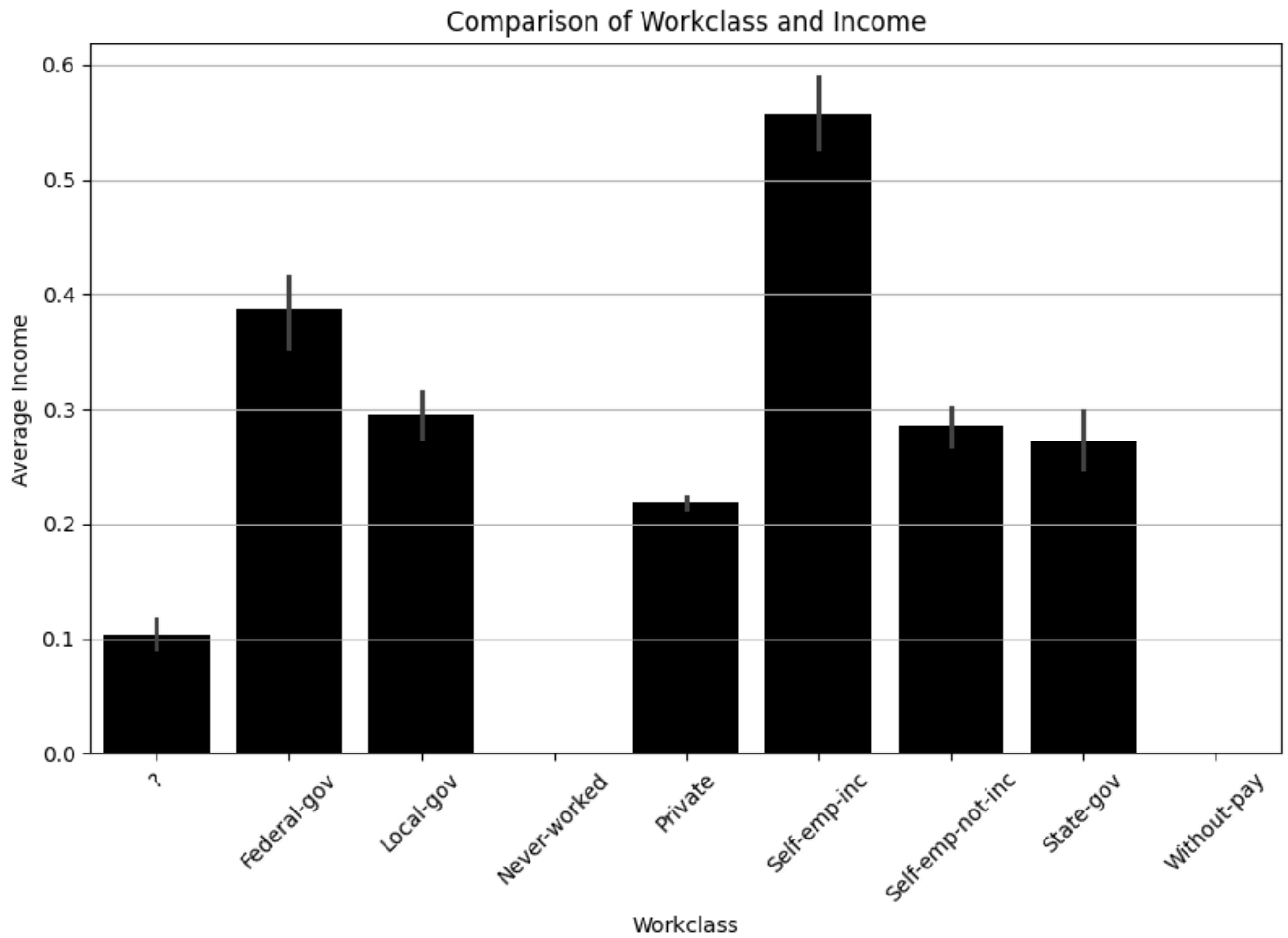
Comparison of Workclass and Income

Description: between workclass categories and income levels. It helps to understand how different types of employment, such as private sector, government, or self-employment, influence income.

```
 1 df['income_numeric'] = df['income'].map({'<=50K': 0, '>50K': 1})
 2
 3 plt.figure(figsize=(10, 6))
 4
 5 sns.barplot(x='education', y='income_numeric', data=df, order=df['education'].value_count
 6 plt.title('Comparison of Education Level and Income')
 7 plt.xlabel('Education Level')
 8 plt.ylabel('Average Income')
 9 plt.xticks(rotation=45)
10 plt.grid(axis='y')
11 plt.show()
```


Comparison of Education Level and Income
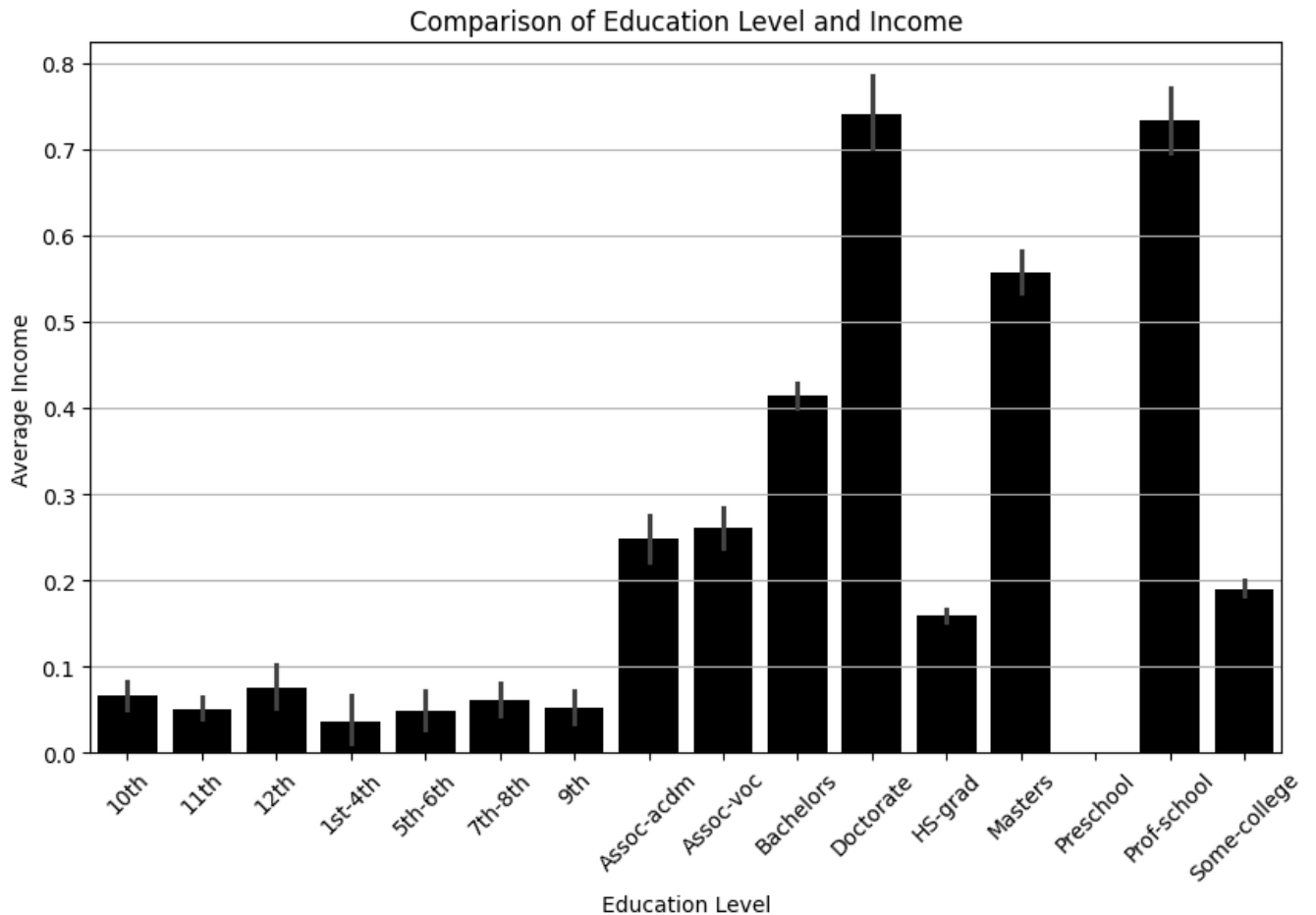
Comparison of Education Level and Income:

Description: This comparison helps to understand the relationship between education level and income. It can reveal whether higher education attainment generally leads to higher income levels, which has implications for policy-making, career choices, and socioeconomic mobility.

```
1 sns.set_theme(style="whitegrid")
2
3 plt.figure(figsize=(8, 6))
4 sns.scatterplot(x=df['hours-per-week'], y=df['income'], alpha=0.5)
5 plt.title('Correlation between Hours-per-week and Income')
6 plt.xlabel('Hours-per-week')
7 plt.ylabel('Income')
8 plt.grid(True)
9 plt.show()
```
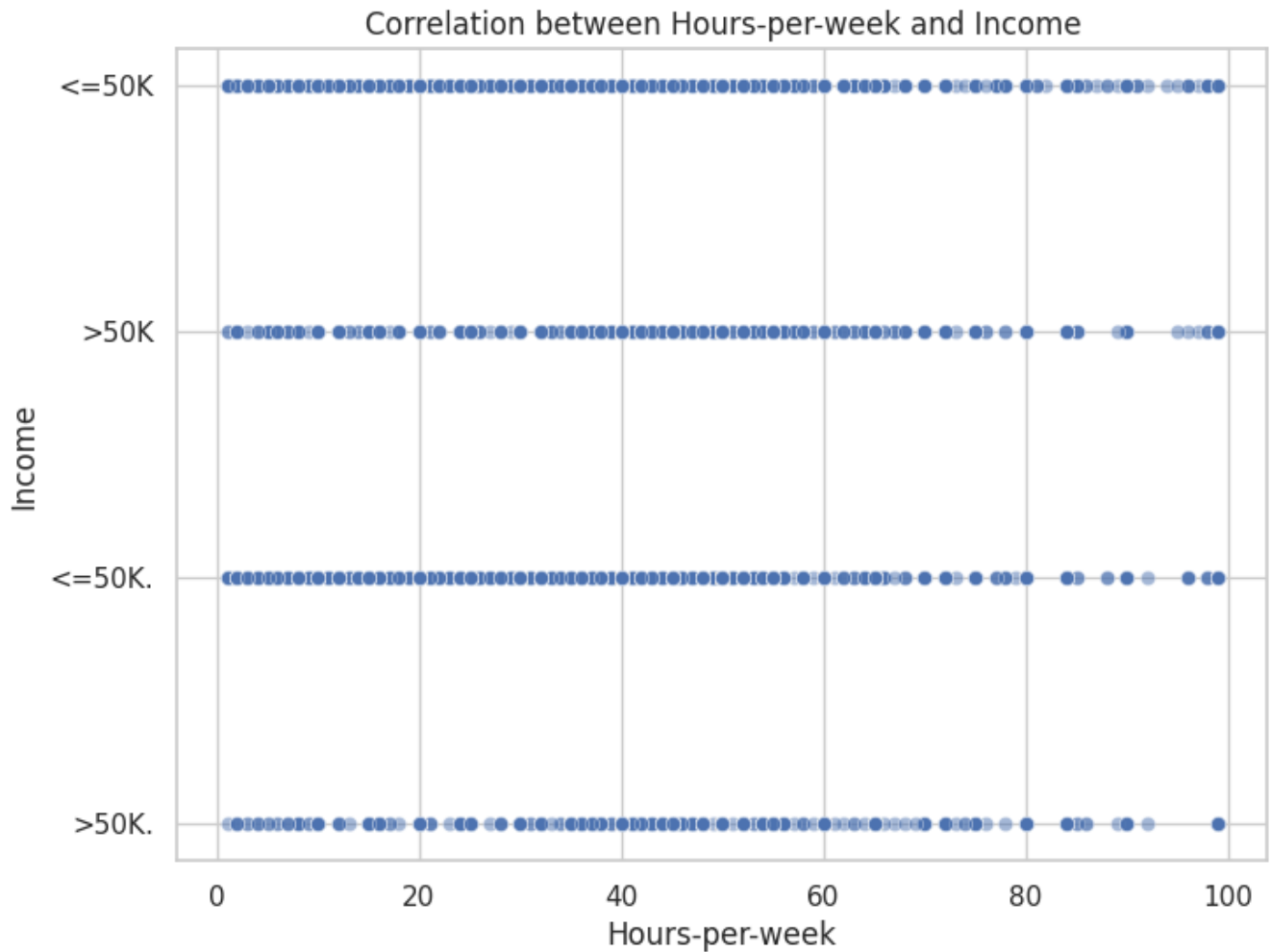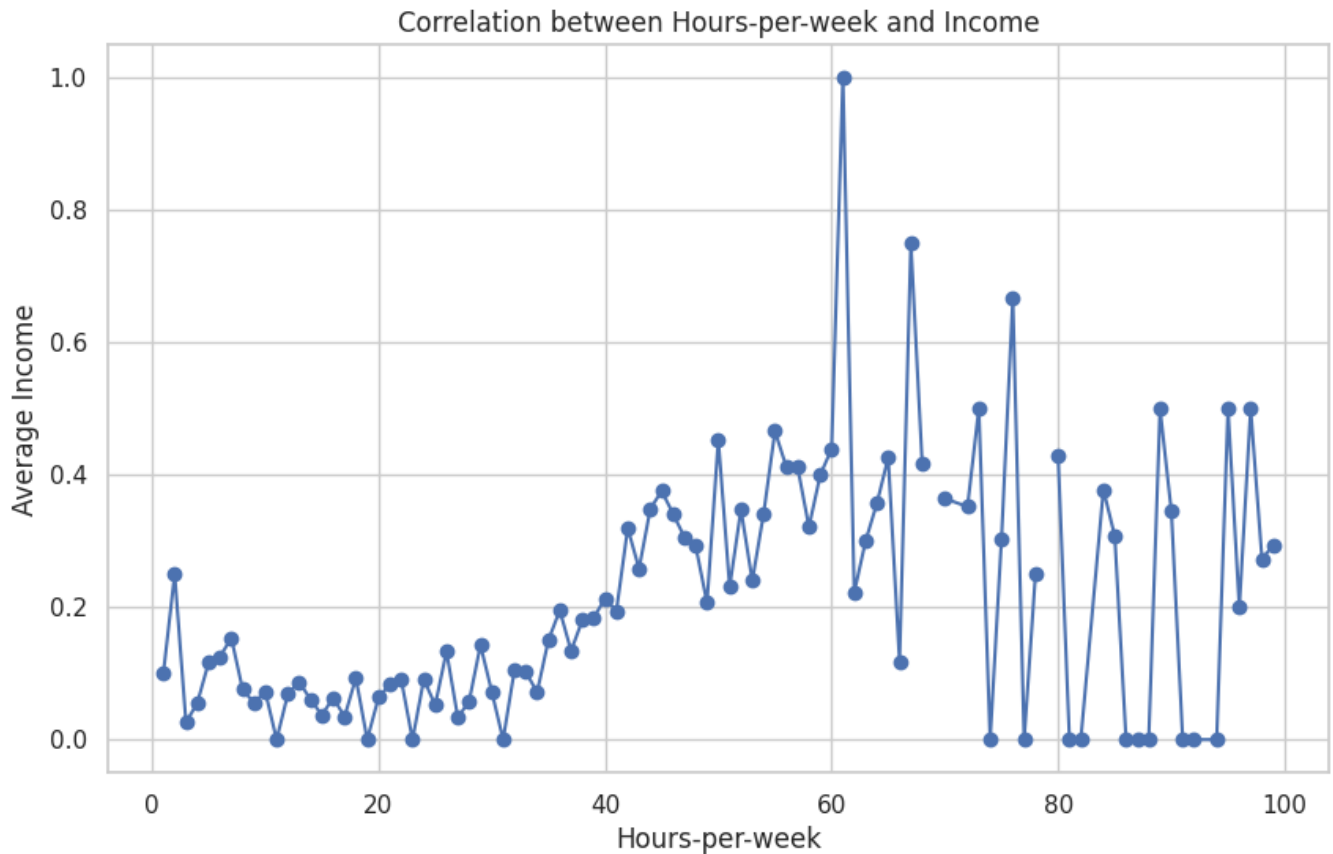


```
1 plt.figure(figsize=(10, 6))
2 plt.plot(df.groupby('hours-per-week')['income_numeric'].mean(), marker='o', linestyle='-'
3 plt.title('Correlation between Hours-per-week and Income')
4 plt.xlabel('Hours-per-week')
5 plt.ylabel('Average Income')
6 plt.grid(True)
7 plt.show()
```

Correlation between Hours-per-week and Income

Correlation between Hours-per-week and Income:

Description: This comparison examines how the number of hours worked per week correlates with income levels. It helps to understand whether working longer hours is associated with higher income and provides insights into labor market dynamics and wage structures.

```
 1 df['marital_numeric'] = df['marital-status'].map({'Married-civ-spouse': 1, 'Never-married
 2
 3 sns.set_theme(style="whitegrid")
 4
 5 plt.figure(figsize=(10, 6))
 6 sns.barplot(x='marital-status', y='income_numeric', data=df, order=df['marital-status'].v
 7 plt.title('Comparison of Marital Status and Income')
 8 plt.xlabel('Marital Status')
 9 plt.ylabel('Average Income')
10 plt.xticks(rotation=45)
11 plt.grid(axis='y')
12 plt.show()
```
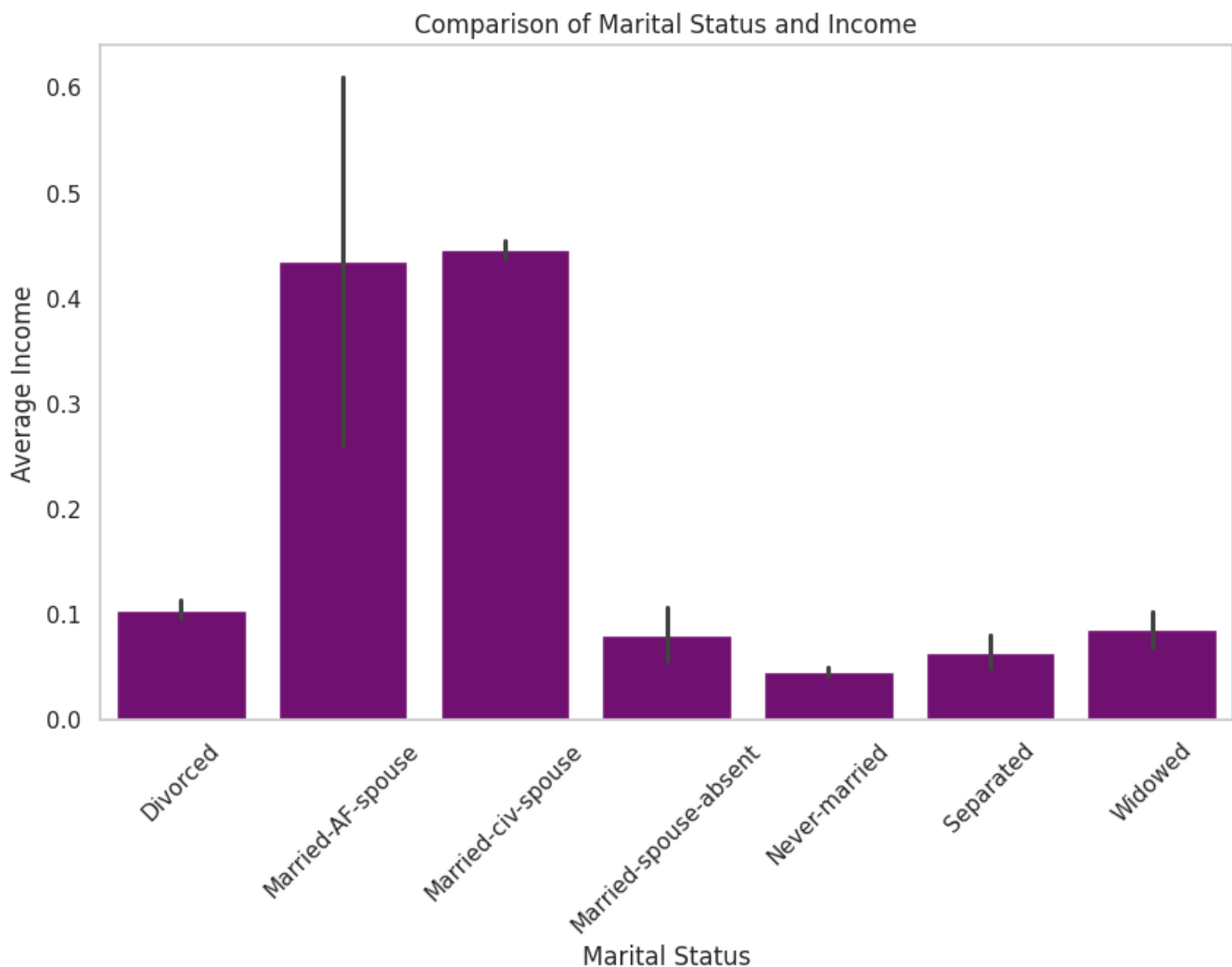
Comparison of Marital Status and Income:

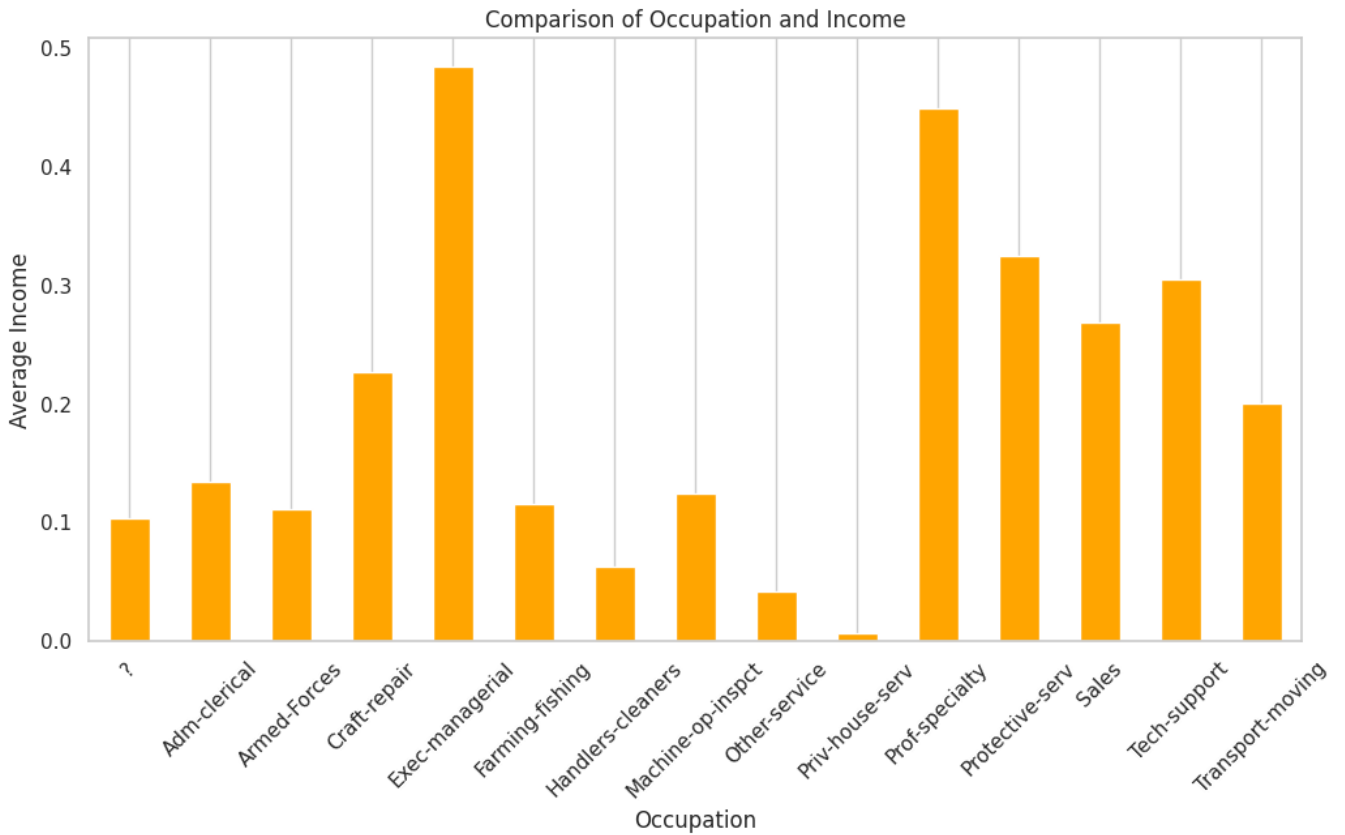**Description:** This comparison explores how marital status correlates with income levels. It can shed light on whether married individuals tend to have higher incomes compared to unmarried individuals, and it may highlight potential socioeconomic factors influencing income disparities between marital statuses.

```
1  df['occupation_numeric'] = df['occupation'].map({'Prof-specialty': 1, 'Craft-repair': 2,
2                                                    'Adm-clerical': 4, 'Sales': 5, 'Other-se
3                                                    'Machine-op-inspct': 7, 'Transport-movir
4                                                    'Farming-fishing': 10, 'Tech-support': 1
5                                                    'Priv-house-serv': 13, 'Armed-Forces': 1
6
7  plt.figure(figsize=(12, 6))
8  occupation_order = df['occupation'].value_counts().index.sort_values()
9  occupation_counts = df.groupby('occupation')['income_numeric'].mean().loc[occupation_orde
10 occupation_counts.plot(kind='bar', color='orange')
11 plt.title('Comparison of Occupation and Income')
12 plt.xlabel('Occupation')
13 plt.ylabel('Average Income')
14 plt.xticks(rotation=45)
15 plt.grid(axis='y')
16 plt.show()
```

Comparison of Occupation and Income:

Description: This comparison examines the income disparities across various occupations. It helps to identify occupations that typically offer higher salaries and those that may have lower income levels. Understanding these income differences can inform career choices, workforce development strategies, and policies aimed at reducing income inequality.

```
 1 occupation_education_pivot = df.pivot_table(index='education', columns='occupation', aggf
 2
 3 plt.figure(figsize=(14, 8))
 4 sns.heatmap(occupation_education_pivot, cmap='Blues', annot=True, fmt='g', linewidths=.5)
 5 plt.title('Distribution of Occupation by Education Level')
 6 plt.xlabel('Occupation')
 7 plt.ylabel('Education Level')
 8 plt.xticks(rotation=45)
 9 plt.yticks(rotation=0)
10 plt.show()
```

Distribution of Occupation by Education Level

| Education Level	?	Adm-clerical	Armed-Forces	Craft-repair	Exec-managerial	Farming-fishing	Handlers-cleaners	Machine-op-inspct	Other-service	Priv-house-serv	Prof-specialty	Protective-serv	Sales	Tech-support	Transport-moving
10th	102	69	0	245	48	72	110	153	287	8	22	12	126	5	130
11th	119	108	0	277	57	71	181	156	374	19	44	20	239	10	137
12th	40	56	1	94	20	29	57	62	131	8	16	12	70	4	56
1st-4th	12	6	0	29	7	33	26	36	56	12	5	1	10	0	12
5th-6th	30	10	0	71	7	53	58	97	100	20	2	1	19	1	39
7th-8th	73	23	0	178	34	107	66	132	154	17	15	12	46	8	89
9th	51	20	0	146	28	44	73	102	146	16	7	9	49	4	61
Assoc-acdm	47	283	0	167	243	28	35	57	111	3	212	50	210	117	38
Assoc-voc	61	270	1	379	238	85	43	96	162	6	246	67	166	184	56
Bachelors	173	772	1	340	2037	114	84	103	267	13	2245	150	1280	347	94
Doctorate	15	6	0	4	85	1	0	1	2	1	451	1	17	8	2
HS-grad	533	2081	5	2956	1230	579	959	1550	1967	94	370	329	1603	282	1239
Masters	48	105	3	37	784	16	6	14	37	1	1302	20	207	61	15
Preschool	5	3	0	7	1	16	5	12	23	2	2	0	3	1	2
Prof-school	18	13	1	10	69	7	0	1	9	0	668	1	23	11	3
Some-college	516	1895	4	1300	1318	268	418	519	1197	29	699	312	1532	434	428

Occupation

```
 1 sns.set_theme(style="whitegrid")
 2
 3 plt.figure(figsize=(10, 6))
 4 relationship_order = df['relationship'].value_counts().index.sort_values()
 5 relationship_counts = df.groupby('relationship')['income_numeric'].mean().loc[relationshi
 6 sns.lineplot(x=relationship_counts.index, y=relationship_counts.values, color='pink')
 7 plt.title('Comparison of Relationship and Income')
 8 plt.xlabel('Relationship')
 9 plt.ylabel('Average Income')
10 plt.xticks(rotation=45)
11 plt.grid(axis='y')
12 plt.show()
```
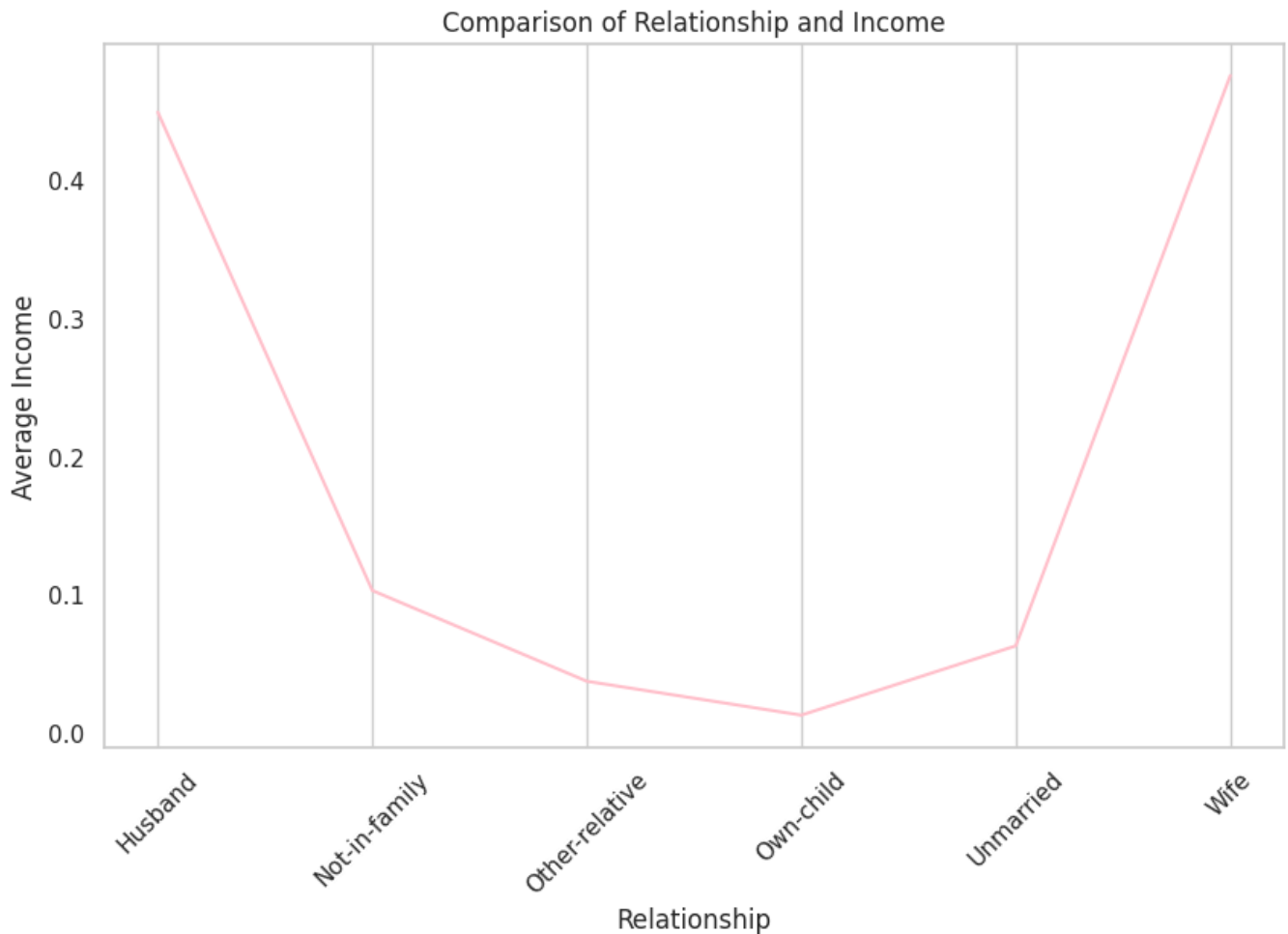

Comparison o Relationship and Income

Description: This comparison examines the correlation between relationship status and income levels. It helps to understand whether factors such as being married, single, or in other relationship statuses impact income, providing insights into household dynamics and financial well-being.

```
1  sns.set_theme(style="whitegrid")
2
3  plt.figure(figsize=(10, 6))
4  race_order = df['race'].value_counts().index.sort_values()
5  race_counts = df.groupby('race')['income_numeric'].mean().loc[race_order]
6  sns.barplot(x=race_counts.index, y=race_counts.values, color='black')
7  plt.title('Comparison of Race and Income')
8  plt.xlabel('Race')
9  plt.ylabel('Average Income')
10 plt.xticks(rotation=45)
11 plt.grid(axis='y')
12 plt.show()
```
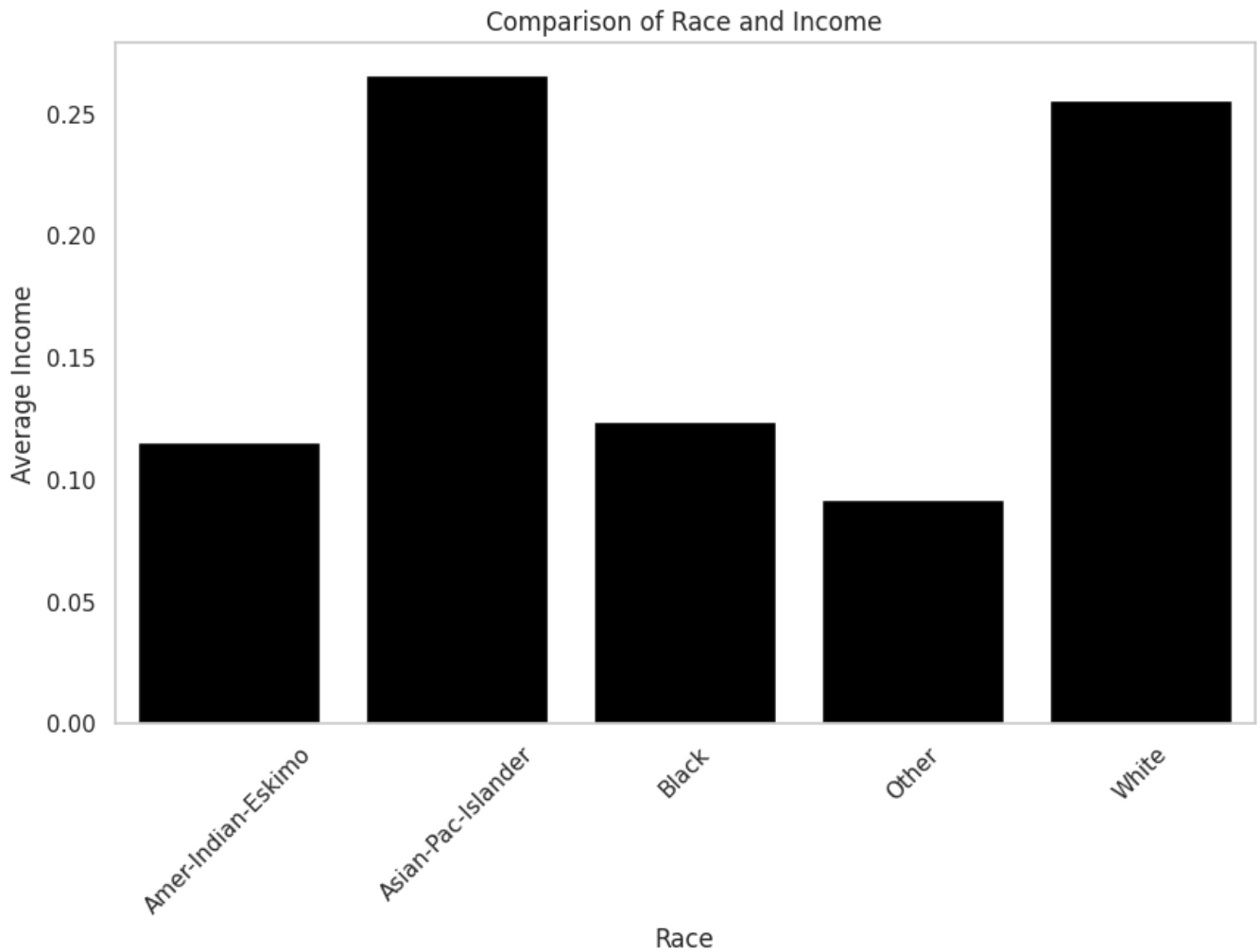
Comparison of Race and Income

Description: This comparison explores income disparities across different racial groups. It helps to identify whether certain racial demographics tend to have higher or lower incomes, highlighting potential disparities and informing efforts to promote equity and inclusion.

```
1 plt.figure(figsize=(8, 6))
2 sex_counts = df.groupby('sex')['income_numeric'].mean()
3 sex_counts.plot(kind='bar', color=['blue', 'pink'], alpha=0.7)
4 plt.title('Comparison of Sex and Income')
5 plt.xlabel('Sex')
6 plt.ylabel('Average Income')
7 plt.xticks(rotation=0)
8 plt.grid(axis='y')
9 plt.show()
```
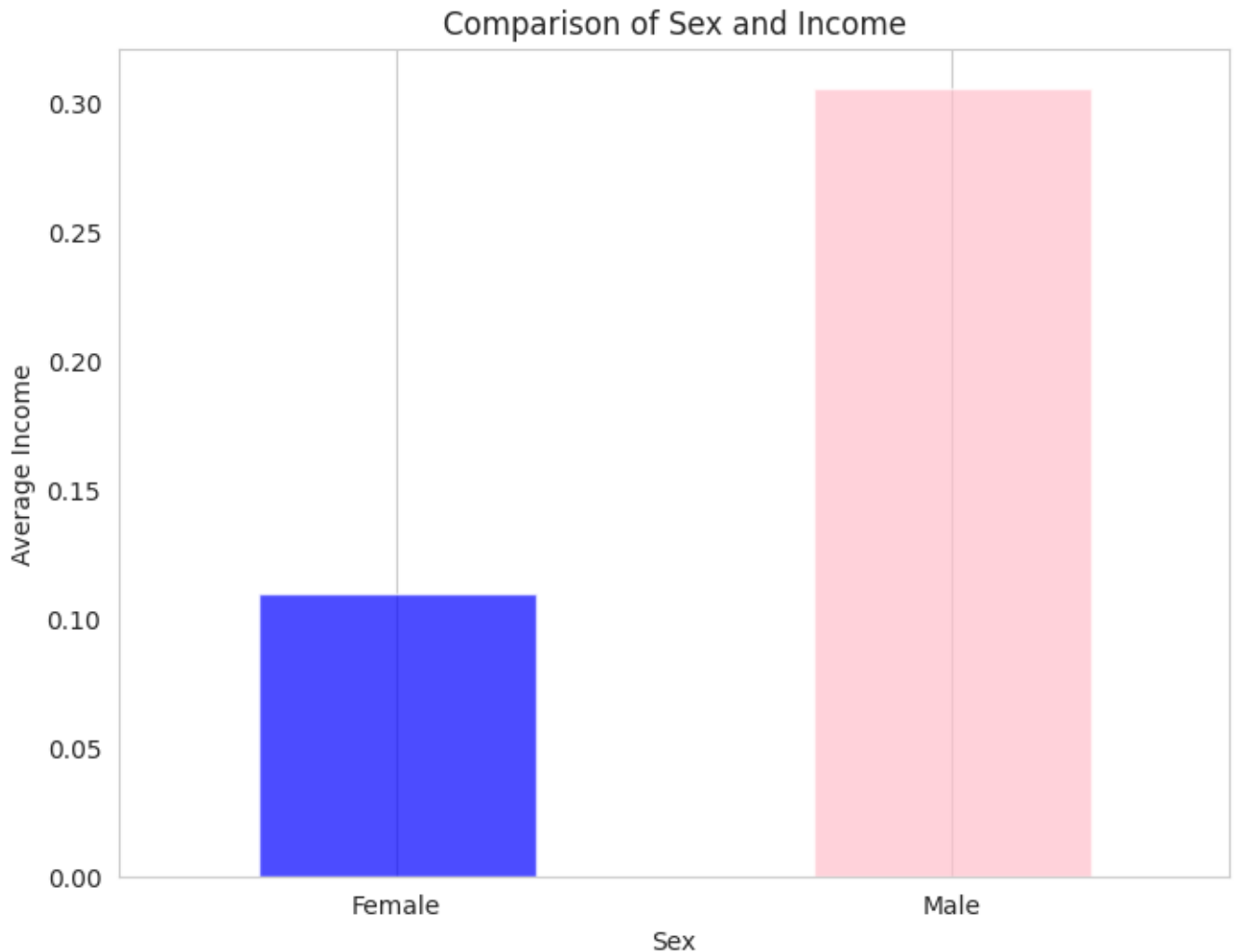


```
1 gender_workclass_pivot = df.pivot_table(index='sex', columns='workclass', aggfunc='size',
2
3 print("Employment Distribution by Gender:")
4 print(gender_workclass_pivot)
```

```
Employment Distribution by Gender:
workclass    ?  Federal-gov  Local-gov  Never-worked  Private  Self-emp-inc  \
sex
Female     839          465       1282             3    11916           225
Male       997         1004       1908             7    22682          1506
```

```
   workclass  Self-emp-not-inc  State-gov  Without-pay
   sex
   Female                  665        779            8
   Male                   3271       1241           15
```

```python
 1  import matplotlib.pyplot as plt
 2
 3  plt.figure(figsize=(10, 6))
 4
 5  gender_workclass_pivot.plot(kind='bar', stacked=True)
 6
 7  plt.title('Employment Distribution by Gender')
 8  plt.xlabel('Gender')
 9  plt.ylabel('Count')
10  plt.xticks(rotation=0)
11
12  plt.legend(title='Workclass')
13
14  plt.tight_layout()
15  plt.show()
16
```

```
<Figure size 1000x600 with 0 Axes>
```


Employment Distribution by Gender
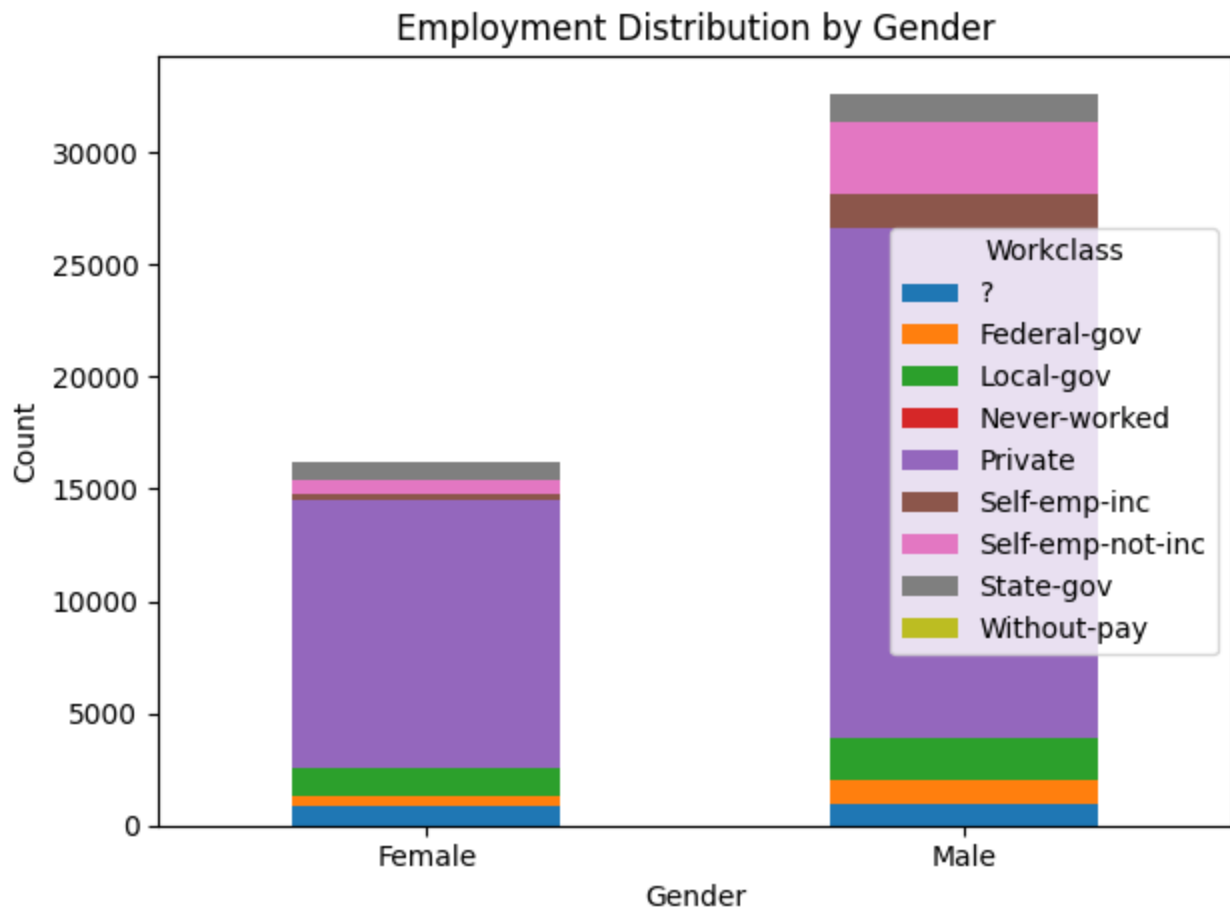
```
1 total_employment = gender_workclass_pivot.sum(axis=1)
2
3 print("Total Employment Count for Each Gender:")
4 print(total_employment)
5
```

```
Total Employment Count for Each Gender:
sex
Female    16182
Male      32631
```

Comparison o Sex and Income:

Description: This comparison investigates income differences between genders. It helps to understand whether there is a gender wage gap and how gender identity influences income levels, providing insights into gender equality and workplace diversity.

In addition it also examines the total employment count for each gender, revealing 16,182 females and 32,631 males. With a total difference of 16,449, it sheds light on potential income disparities