```
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
```

```
import pandas as pd
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
census_income = fetch_ucirepo(id=20)

# data (as pandas dataframes)
X = census_income.data.features
y = census_income.data.targets

# metadata
print(census_income.metadata)

# variable information
print(census_income.variables)
```

```
{'uci_id': 20, 'name': 'Census Income', 'repository_url': 'https://archive.ics.uci.edu/dataset/20/census+income', 'data_url': 'https://archive.ics.uci.edu/st
              name     role         type      demographic  \
0              age  Feature      Integer              Age
1        workclass  Feature  Categorical           Income
2           fnlwgt  Feature      Integer             None
3        education  Feature  Categorical  Education Level
4    education-num  Feature      Integer  Education Level
5   marital-status  Feature  Categorical            Other
6       occupation  Feature  Categorical            Other
7     relationship  Feature  Categorical            Other
8             race  Feature  Categorical             Race
9              sex  Feature       Binary              Sex
10    capital-gain  Feature      Integer             None
11    capital-loss  Feature      Integer             None
12  hours-per-week  Feature      Integer             None
13  native-country  Feature  Categorical            Other
14          income   Target       Binary           Income

                                          description units missing_values
0                                                 N/A  None             no
1   Private, Self-emp-not-inc, Self-emp-inc, Feder...  None            yes
2                                                None  None             no
3    Bachelors, Some-college, 11th, HS-grad, Prof-...  None             no
4                                                None  None             no
5   Married-civ-spouse, Divorced, Never-married, S...  None             no
6   Tech-support, Craft-repair, Other-service, Sal...  None            yes
7   Wife, Own-child, Husband, Not-in-family, Other...  None             no
8   White, Asian-Pac-Islander, Amer-Indian-Eskimo,...  None             no
9                                       Female, Male.  None             no
10                                               None  None             no
11                                               None  None             no
12                                               None  None             no
13  United-States, Cambodia, England, Puerto-Rico,...  None            yes
14                                        >50K, <=50K.  None             no
```

```
X
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	s
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema
...	...	...	...	...	...	...	...	...	...	

Next steps: View recommended plots

```
y
```

	income
0	<=50K
1	<=50K
2	<=50K
3	<=50K
4	<=50K
...	...
48837	<=50K.
48838	<=50K.
48839	<=50K.
48840	<=50K.
48841	>50K.

48842 rows × 1 columns

Next steps: View recommended plots

```
df = pd.concat((X, y), axis = 1)
df
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	s
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema
...	...	...	...	...	...	...	...	...	...	

Next steps: View recommended plots

Data cleaning

Checks for null values

```
print(df.isnull().sum())
```

```
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
```

Checks for values on each columns

```
for i in df.columns:
  print(df[i].value_counts(), '\n')
```

```
Japan                           92
Guatemala                       88
Poland                          87
Vietnam                         86
Columbia                        85
Haiti                           75
Portugal                        67
Taiwan                          65
Iran                            59
Greece                          49
Nicaragua                       49
Peru                            46
Ecuador                         45
France                          38
Ireland                         37
Hong                            30
Thailand                        30
Cambodia                        28
Trinadad&Tobago                 27
Laos                            23
Yugoslavia                      23
Outlying-US(Guam-USVI-etc)      23
Scotland                        21
Honduras                        20
Hungary                         19
Holand-Netherlands               1
Name: count, dtype: int64

income
<=50K     24720
<=50K.    12435
>50K       7841
>50K.      3846
Name: count, dtype: int64
```

Saw a '?' values so I wanted to know them

```
for i in df.columns:
    if df[i].dtype == object:
        if (df[i] == '?').any():
            print(i)
            print(df[i][df[i] == '?'].count())
            print()
```

```
workclass
1836

occupation
1843

native-country
583
```

Peek on rows with null values

```
null_mask = df.isnull().any(axis=1)
null_rows = df[null_mask]

null_rows
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	s
32565	18	NaN	103497	Some-college	10	Never-married	NaN	Own-child	White	Fema
32567	29	NaN	227026	HS-grad	9	Never-married	NaN	Unmarried	Black	Ma
32574	58	NaN	299831	HS-grad	9	Married-civ-spouse	NaN	Husband	White	Ma
32580	40	Private	85019	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Ma
32583	72	NaN	132015	7th-8th	4	Divorced	NaN	Not-in-family	White	Fema
...	...	...	...	...	...	...	...	...	...	
48769	21	NaN	212661	Some-	10	Never-	NaN	Own-child	White	Fema

Next steps: View recommended plots

Checks if when workclass is NaN occupation is also NaN since workclass == 963 and occupation is 966

```
occupation = df[df['occupation'].isnull() & df['workclass'].isnull()]
occupation.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex
32565	18	NaN	103497	Some-college	10	Never-married	NaN	Own-child	White	Female
32567	29	NaN	227026	HS-grad	9	Never-married	NaN	Unmarried	Black	Male
						Married-				

Peek on the 3 difference on the workclass and occupation

```
i = df[df['occupation'].isnull() & df['workclass'].notnull()]
i
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex
41346	17	Never-worked	131593	11th	7	Never-married	NaN	Own-child	Black	Female

since columns with values of '?' and 'NaN' are the same columns[workclass, occupation, native-country], and We can both conclude or rename them as 'unknown' since they also represent unknown values; to make the representation of data more accurate since it is the data fetched in surveys

```
df.fillna('unknown', inplace=True) # For NaN values
df.replace('?', 'unknown', inplace=True) # For "?" values
```

checks for NaN values and '?' values

```
print(df.isnull().sum())
```

```
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
```

```
for i in df.columns:
    if df[i].dtype == object:
        if (df[i] == '?').any():
            print(i)
            print(df[i][df[i] == '?'].count())
        else:
          print(f'{i}: None')
```

```
workclass: None
education: None
marital-status: None
occupation: None
relationship: None
race: None
sex: None
native-country: None
income: None
```

Peeks at duplicated values and then drop them

```
dup = df.duplicated().sum()
dup
```

```
29
```

```
df.drop_duplicates(inplace=True)
```

Checks for duplicates

```
dup = df.duplicated().sum()
dup
```

```
0
```

DATA NUMERICAL REPRESENTATION

```
df # peek for the set
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	s
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema
...	...	...	...	...	...	...	...	...	...	

Next steps: View recommended plots

Checks for dirty uniques

```
for i in df.columns[1:]:
  if df[i].dtype == object:
    print(i)
    print(df[i].unique())
    print("\n")
```

```
workclass
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
 'unknown' 'Self-emp-inc' 'Without-pay' 'Never-worked']


education
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']


marital-status
['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']


occupation
['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' 'unknown'
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']


relationship
['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relative']


race
['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']


sex
['Male' 'Female']


native-country
['United-States' 'Cuba' 'Jamaica' 'India' 'unknown' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']


income
['<=50K' '>50K' '<=50K.' '>50K.']
```

```
# Only income is somewhat dirty
print(list(df['income'].unique()))
```

```
['<=50K', '>50K', '<=50K.', '>50K.']
```

```
df['income'].replace({'<=50K.' : '<=50K', '>50K.' : '>50K'}, inplace = True)
```

```
df['income'].value_counts()
```

```
income
<=50K    37128
>50K     11685
Name: count, dtype: int64
```

```
# Stores the names for later purposes
names = {}
for i in df.columns:
  names[i] = df[i].unique()
# print(names['income'])
```

```
print(df['workclass'].unique())
print(names['workclass'])
```

```
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
 'unknown' 'Self-emp-inc' 'Without-pay' 'Never-worked']
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
 'unknown' 'Self-emp-inc' 'Without-pay' 'Never-worked']
```

```
df[['education', 'education-num']].value_counts()
```

```
education     education-num
HS-grad       9                15777
Some-college  10               10869
Bachelors     13                8020
Masters       14                2656
Assoc-voc     11                2060
11th          7                 1812
Assoc-acdm    12                1601
10th          6                 1389
7th-8th       4                  954
Prof-school   15                 834
9th           5                  756
12th          8                  656
Doctorate     16                 594
5th-6th       3                  508
1st-4th       2                  245
Preschool     1                   82
Name: count, dtype: int64
```

```
df['workclass'].value_counts()
```

```
workclass
Private             33879
Self-emp-not-inc     3861
Local-gov            3136
unknown              2799
State-gov            1981
Self-emp-inc         1694
Federal-gov          1432
Without-pay            21
Never-worked           10
Name: count, dtype: int64
```

```
df['income'].value_counts()
```

```
income
<=50K    37128
>50K     11685
Name: count, dtype: int64
```

```
df['income'].value_counts()
```

```
income
<=50K    37128
>50K     11685
Name: count, dtype: int64
```

```
print(df['native-country'].unique())
print(names['native-country'])
```

```
['United-States' 'Cuba' 'Jamaica' 'India' 'unknown' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']
['United-States' 'Cuba' 'Jamaica' 'India' 'unknown' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']
```

```
df['native-country'].value_counts()
```

```
native-country
United-States                 43810
Mexico                          947
unknown                         856
Philippines                     295
Germany                         206
Puerto-Rico                     184
Canada                          182
El-Salvador                     155
India                           151
Cuba                            138
England                         127
China                           122
South                           115
Jamaica                         106
Italy                           105
Dominican-Republic              103
Japan                            92
Poland                           87
Guatemala                        86
Vietnam                          86
Columbia                         85
Haiti                            75
Portugal                         67
Taiwan                           65
Iran                             59
Greece                           49
Nicaragua                        49
Peru                             46
Ecuador                          45
France                           38
Ireland                          37
Hong                             30
Thailand                         30
Cambodia                         28
Trinadad&Tobago                  27
Laos                             23
```

```
Yugoslavia                     23
Outlying-US(Guam-USVI-etc)     23
Scotland                       21
Honduras                       20
Hungary                        19
Holand-Netherlands              1
Name: count, dtype: int64
```

```
for i in df.columns:
  if df[i].dtype == object:
    if i != 'education' and i != 'workclass':
      num = 1
      for j in df[i].unique():
        df[i].replace({j:num}, inplace = True)
        num += 1
```

```
df['education'].replace({'Preschool': 'Elem-grad', '1st-4th': 'Elem-grad', '5th-6th': 'Elem-grad',
                         '12th': 'HS-grad', '9th': 'Elem-grad', '7th-8th': 'Elem-grad',
                         '10th': 'Elem-grad', '11th': 'HS-grad'}, inplace=True)
```

```
df[['education', 'education-num']].value_counts()
```

```
education     education-num
HS-grad       9                15777
Some-college  10               10869
Bachelors     13                8020
Masters       14                2656
Assoc-voc     11                2060
11th          7                 1812
Assoc-acdm    12                1601
10th          6                 1389
7th-8th       4                  954
Prof-school   15                 834
9th           5                  756
12th          8                  656
Doctorate     16                 594
5th-6th       3                  508
1st-4th       2                  245
Preschool     1                   82
Name: count, dtype: int64
```

```
df['native-country'].value_counts() # Chekcs if it is the same with the previous counting and yes it is
```

Data Analysis

```
income_counts = df['income'].value_counts()
percentage_1 = (income_counts[1] / len(df)) * 100
percentage_2 = (income_counts[2] / len(df)) * 100

print("Percentage of 1:", round(percentage_1, 1))
print("Percentage of 2:", round(percentage_2, 1))
```

```
Percentage of 1: 76.1
Percentage of 2: 23.9
```

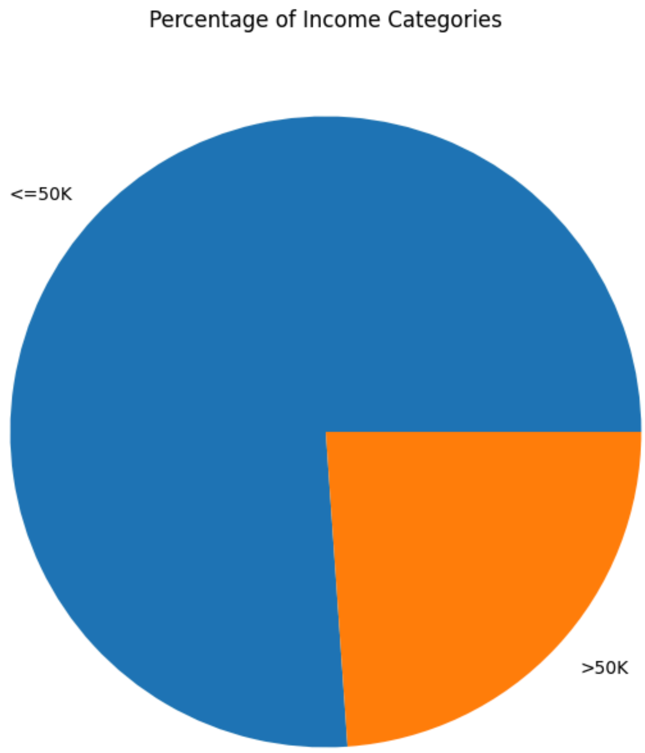
```
percent = pd.DataFrame({'Income' : names['income'], 'Count' : df['income'].value_counts().values, 'Percentage' : [round(percentage_1, 1), round(percentage_2, 1)]})
percent
```

	Income	Count	Percentage
0	<=50K	37128	76.1
1	>50K	11685	23.9

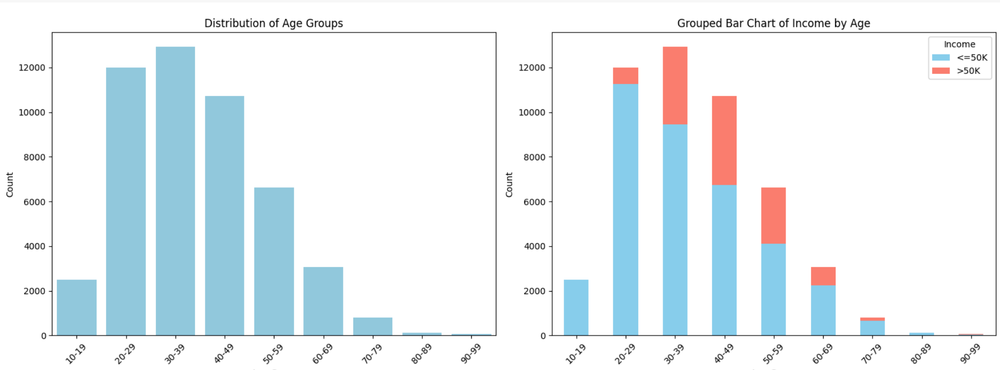
Next steps: View recommended plots

```
percent = [round(percentage_1, 1), round(percentage_2, 1)]
plt.figure(figsize=(8, 8))
plt.pie(percent, labels=names['income'])
plt.title('Percentage of Income Categories')
```

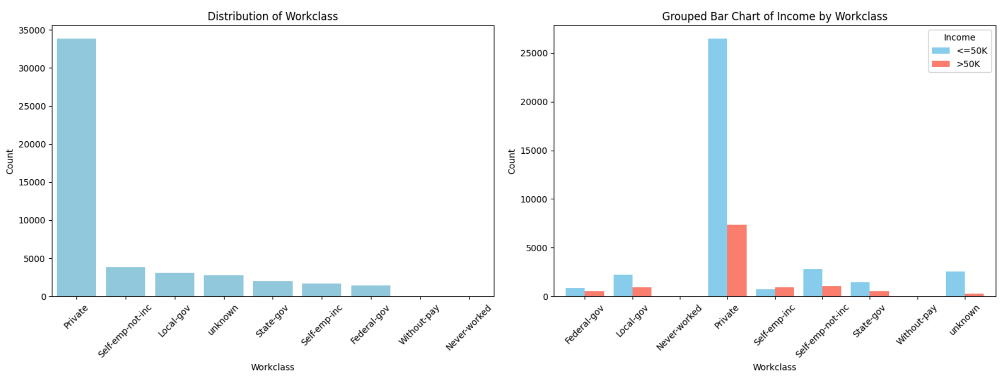
```
Text(0.5, 1.0, 'Percentage of Income Categories')
```



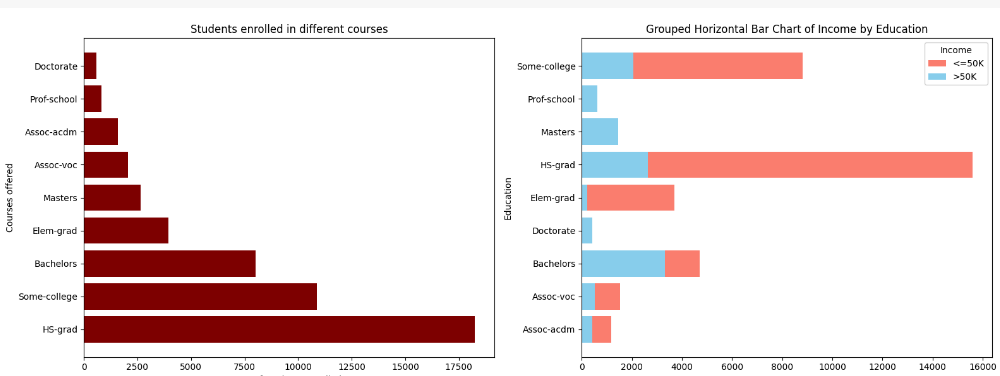

```
import numpy as np
age_groups = np.arange(10, 101, 10)
labels = [f"{age}-{age+9}" for age in age_groups[:-1]]
age_bins = pd.cut(df['age'], bins=age_groups, labels=labels, right=False)
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.barplot(x=age_bins.value_counts().index, y=age_bins.value_counts().values, color='skyblue', ax=axes[0])
axes[0].set_title('Distribution of Age Groups')
axes[0].set_ylabel('Count')
axes[0].set_xlabel('Age Group')
axes[0].tick_params(axis='x', rotation=45)
grouped_counts = df.groupby([age_bins, 'income']).size().unstack()
grouped_counts.plot(kind='bar', stacked=True, color=['skyblue', 'salmon'], ax=axes[1])
axes[1].set_title('Grouped Bar Chart of Income by Age')
axes[1].set_xlabel('Age Group')
axes[1].set_ylabel('Count')
axes[1].legend(labels=names['income'], title='Income')
axes[1].tick_params(axis='x', rotation=45)
plt.tight_layout()
plt.show()
```



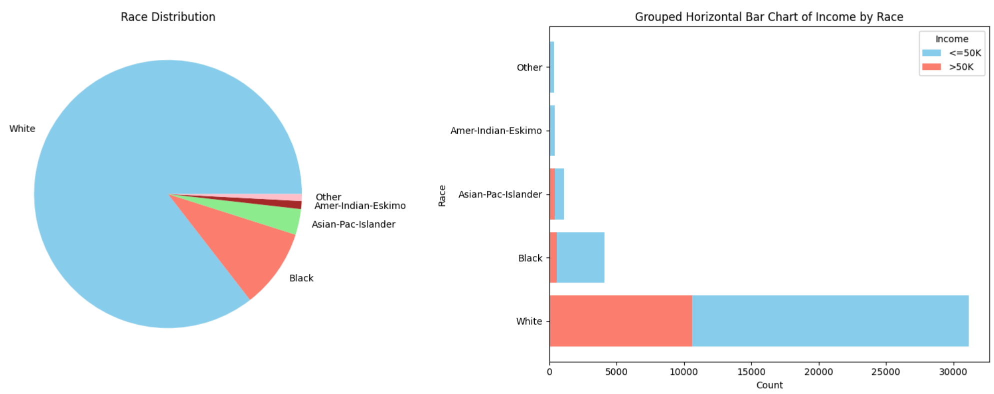
```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.barplot(x=df['workclass'].value_counts().index, y=df['workclass'].value_counts(), color='skyblue', ax=axes[0])
axes[0].set_title('Distribution of Workclass')
axes[0].set_ylabel('Count')
axes[0].set_xlabel('Workclass')
axes[0].tick_params(axis='x', rotation=45)
grouped_counts = df.groupby(['workclass', 'income']).size().unstack()
grouped_counts.plot(kind='bar', stacked=False, width=0.8, color=['skyblue', 'salmon'], ax=axes[1])
axes[1].legend(labels=names['income'], title='Income')
axes[1].set_title('Grouped Bar Chart of Income by Workclass')
axes[1].set_xlabel('Workclass')
axes[1].set_ylabel('Count')
axes[1].tick_params(axis='x', rotation=45)
plt.tight_layout()
plt.show()
```



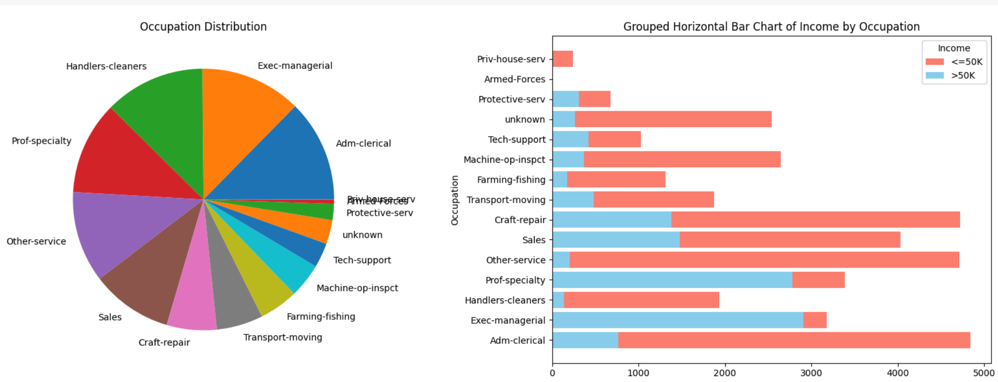

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
course_counts = df['education'].value_counts()
axes[0].barh(course_counts.index, course_counts.values, color='maroon')
axes[0].set_xlabel("No. of students enrolled")
axes[0].set_ylabel("Courses offered")
axes[0].set_title("Students enrolled in different courses")
grouped_counts = df.groupby(['education', 'income']).size().unstack()
education_levels = grouped_counts.index
colors = ['salmon', 'skyblue']
for income_category, color in zip(grouped_counts.columns, colors):
    axes[1].barh(education_levels, grouped_counts[income_category], color=color, label=income_category)
axes[1].legend(labels=names['income'], title='Income')
axes[1].set_title('Grouped Horizontal Bar Chart of Income by Education')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Education')
plt.tight_layout()
plt.show()
```



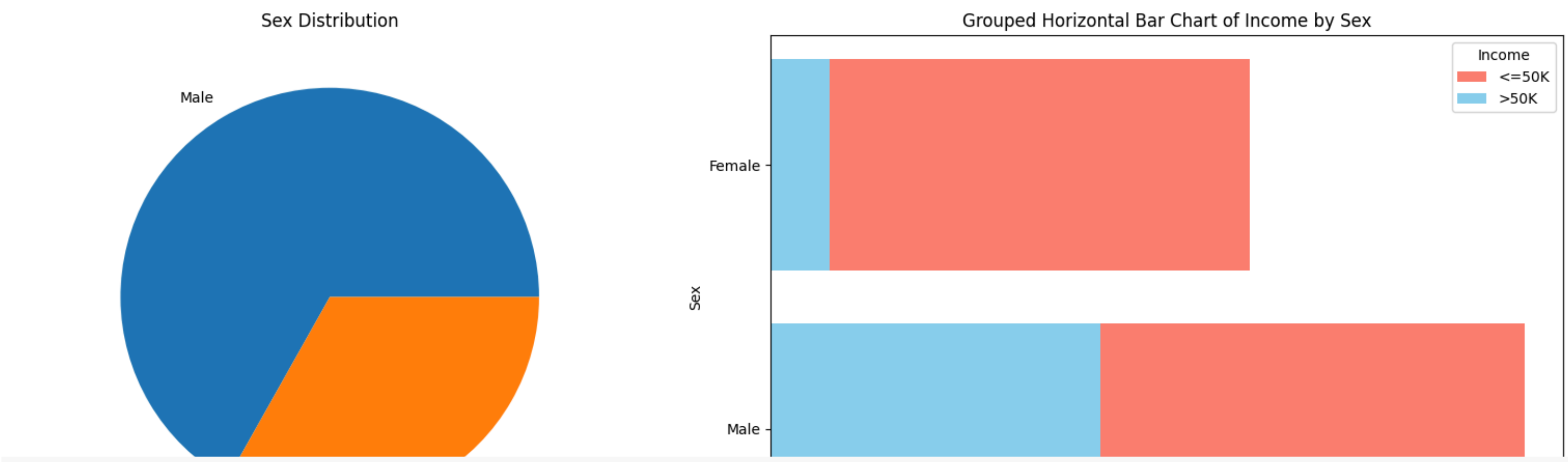
```python
custom_colors = ['skyblue', 'salmon', 'lightgreen', 'brown', 'pink']
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
axes[0].pie(df['race'].value_counts(), labels=names['race'], colors=custom_colors)
axes[0].set_title('Race Distribution')
grouped_counts = df.groupby(['race', 'income']).size().unstack()
race_names = names['race']
for income_category, color in zip(grouped_counts.columns, custom_colors[:2]):
    axes[1].barh(race_names, grouped_counts[income_category], color=color, label=income_category)
axes[1].set_title('Grouped Horizontal Bar Chart of Income by Race')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Race')
axes[1].legend(labels=names['income'], title='Income')
plt.tight_layout()
plt.show()
```



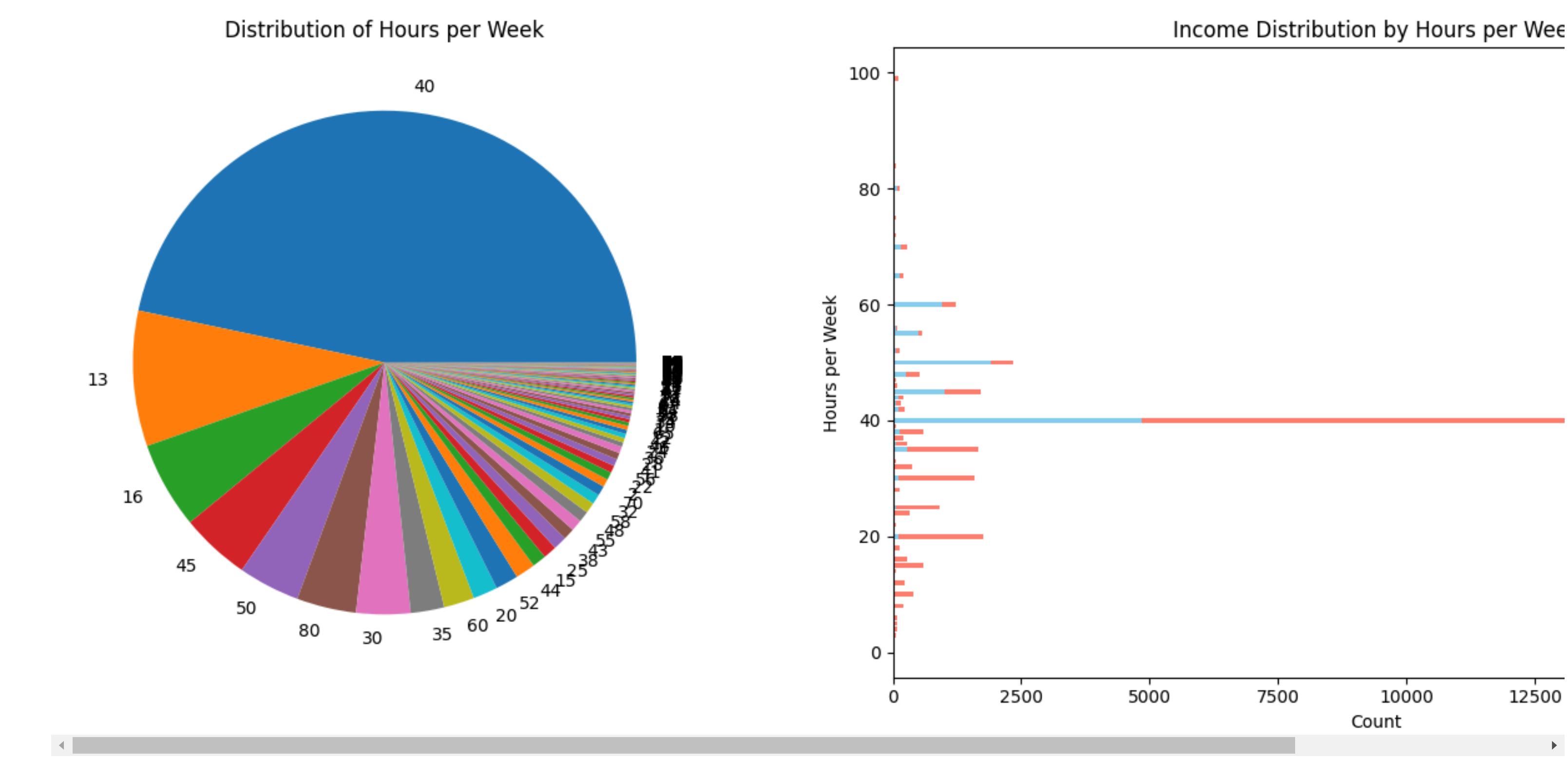

```python
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
axes[0].pie(df['occupation'].value_counts(), labels=names['occupation'])
axes[0].set_title('Occupation Distribution')
grouped_counts = df.groupby(['occupation', 'income']).size().unstack()
for income_category, color in zip(grouped_counts.columns, ['salmon', 'skyblue']):
    axes[1].barh(names['occupation'], grouped_counts[income_category], color=color, label=income_category)
axes[1].legend(labels=names['income'], title='Income')
axes[1].set_title('Grouped Horizontal Bar Chart of Income by Occupation')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Occupation')
plt.tight_layout()
plt.show()
```



```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
axes[0].pie(df['sex'].value_counts(), labels=names['sex'])
axes[0].set_title('Sex Distribution')
grouped_counts = df.groupby(['sex', 'income']).size().unstack()
for income_category, color in zip(grouped_counts.columns, ['salmon', 'skyblue']):
    axes[1].barh(names['sex'], grouped_counts[income_category], color=color, label=income_category)
axes[1].legend(labels=names['income'], title='Income')
axes[1].set_title('Grouped Horizontal Bar Chart of Income by Sex')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Sex')
plt.tight_layout()
plt.show()
```




```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
axes[0].pie(df['hours-per-week'].value_counts(), labels=df['hours-per-week'].unique())
axes[0].set_title('Distribution of Hours per Week')
grouped_counts = df.groupby(['hours-per-week', 'income']).size().unstack()
for income_category, color in zip(grouped_counts.columns, ['salmon', 'skyblue']):
    axes[1].barh(grouped_counts.index, grouped_counts[income_category], color=color, label=income_category)
axes[1].legend(labels=names['income'], title='Income')
axes[1].set_title('Income Distribution by Hours per Week')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Hours per Week')
plt.tight_layout()
plt.show()
```




```
df
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	1	1	1	1	1	2174	0	40	1	1
1	50	Self-emp-not-inc	83311	Bachelors	13	2	2	2	1	1	0	0	13	1	1
2	38	Private	215646	HS-grad	9	3	3	1	1	1	0	0	40	1	1
3	53	Private	234721	HS-grad	7	2	3	2	2	1	0	0	40	1	1
4	28	Private	338409	Bachelors	13	2	4	3	2	2	0	0	40	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	39	Private	215419	Bachelors	13	3	4	1	1	2	0	0	36	1	1
48838	64	unknown	321403	HS-grad	9	7	12	6	2	1	0	0	40	1	1
48839	38	Private	374983	Bachelors	13	2	4	2	1	1	0	0	50	1	1
48840	44	Private	83891	Bachelors	13	3	1	4	3	1	5455	0	40	1	1
		Self-emp-													