In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_erro
import matplotlib.pyplot as plt
from sklearn.tree import export_graphviz
import graphviz
```

In [2]:
```python
df = pd.read_csv('data.csv')
df_edu = pd.read_csv("education_data.csv")
```

In [3]:
```python
df.shape
```

Out[3]:
```
(14534, 14)
```

In [4]:
```python
#Checking all data
df_edu
```

Out[4]:

	unique_id	education	education-num
0	b788ecae-24d3-4d2f-be70-291c656cfe35	Some-college	10
1	c50f5eec-eee7-4f71-92ad-5fd2baace470	Some-college	10
2	42dbe1ab-593f-4d41-b72f-32bb2081c64c	Bachelors	13
3	008c8da5-1688-4b85-a0d0-0516405c84d0	HS-grad	9
4	32084da0-d75e-4f29-8c86-b7d120cdb759	HS-grad	9
...	...	...	...
14529	0b97458e-e408-4956-a7b2-6e5f071906c5	Some-college	10
14530	d5c5f912-84e0-4149-8bf6-c4776a1dad57	HS-grad	9
14531	81694b3e-67a3-4c81-912e-be419e1408ea	Some-college	10
14532	7cdd164a-81f3-436c-abe9-c8cb5795f586	Some-college	10
14533	7e514127-4ac2-40ec-b971-a8113e4302a0	Bachelors	13

14534 rows × 3 columns

In [5]:

```
df
```

Out[5]:

	unique_id	birth_year	workclass	fnlwgt	marital-status	occupation
0	b788ecae-24d3-4d2f-be70-291c656cfe35	2006	?	137363	Never-married	?
1	c50f5eec-eee7-4f71-92ad-5fd2baace470	1989	Private	188041	Married-civ-spouse	Exec-manageria
2	42dbe1ab-593f-4d41-b72f-32bb2081c64c	1999	Private	178478	Never-married	Adm-clerica
3	008c8da5-1688-4b85-a0d0-0516405c84d0	1981	Private	177905	Married-civ-spouse	Handlers-cleaners
4	32084da0-d75e-4f29-8c86-b7d120cdb759	1988	?	389850	Married-spouse-absent	?
...	...	...	...	...	...	...
14529	0b97458e-e408-4956-a7b2-6e5f071906c5	1984	Self-emp-not-inc	209833	Married-civ-spouse	Craft-repai
14530	d5c5f912-84e0-4149-8bf6-c4776a1dad57	1982	Private	102085	Divorced	Other-service
14531	81694b3e-67a3-4c81-912e-be419e1408ea	1994	Private	39386	Married-civ-spouse	Exec-manageria
14532	7cdd164a-81f3-436c-abe9-c8cb5795f586	1982	Private	37869	Married-civ-spouse	Craft-repai
14533	7e514127-4ac2-40ec-b971-a8113e4302a0	1982	Federal-gov	34218	Married-civ-spouse	Exec-manageria

14534 rows × 14 columns

In [6]:
```
df.shape
```

Out[6]: (14534, 14)

In [7]:
```
df_edu.shape
```

Out[7]: (14534, 3)

In [8]:
```
#Merging both dataset
merged_df = pd.merge(df, df_edu, on="unique_id", how="inner")
```

In [9]:
```
#Check if the merging works
merged_df.shape
```

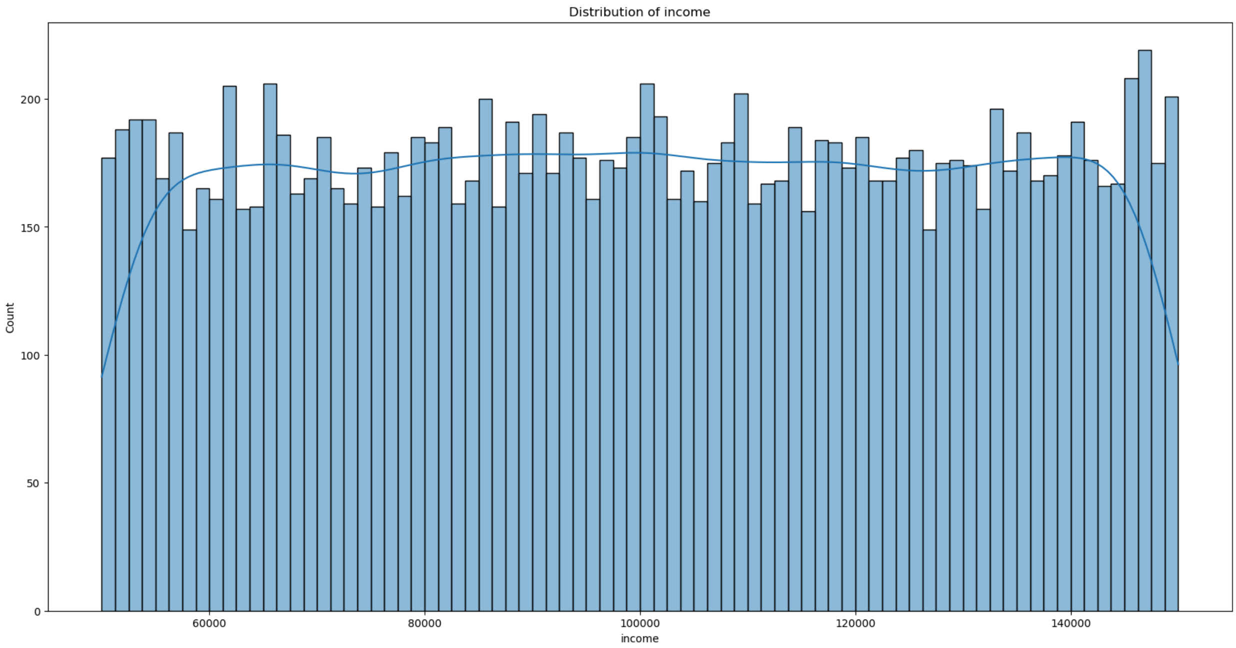
Out[9]: (14147, 16)

In [10]:
```
merged_df.isnull().sum()
```

Out[10]:
```
unique_id            0
birth_year           0
workclass          705
fnlwgt               0
marital-status       0
occupation           0
relationship         0
race                 0
sex               1757
capital-gain         0
capital-loss         0
hours-per-week       0
native-country       0
income               0
education            0
education-num        0
dtype: int64
```

Note that there are less rows when merged, because I use "inner" on "Unique_id" so if there are no similar Unique_id on both dataset it will erase the rows thus having less rows when merged

In [11]:
```
#Checking the distribution of income
plt.figure(figsize=(20, 10))
sns.histplot(merged_df['income'], kde=True, bins=80)
plt.title('Distribution of income')
plt.show()
```



1. DATA CLEANING

In [12]:
```
#Checking for missing data
merged_df.isna().sum()
```

Out[12]:
```
unique_id            0
birth_year           0
workclass          705
fnlwgt               0
marital-status       0
occupation           0
relationship         0
race                 0
sex               1757
capital-gain         0
capital-loss         0
hours-per-week       0
native-country       0
income               0
education            0
education-num        0
dtype: int64
```

In [13]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14147 entries, 0 to 14146
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   unique_id       14147 non-null  object
 1   birth_year      14147 non-null  int64 
 2   workclass       13442 non-null  object
 3   fnlwgt          14147 non-null  int64 
 4   marital-status  14147 non-null  object
 5   occupation      14147 non-null  object
 6   relationship    14147 non-null  object
 7   race            14147 non-null  object
 8   sex             12390 non-null  object
 9   capital-gain    14147 non-null  int64 
 10  capital-loss    14147 non-null  int64 
 11  hours-per-week  14147 non-null  int64 
 12  native-country  14147 non-null  object
 13  income          14147 non-null  int64 
 14  education       14147 non-null  object
 15  education-num   14147 non-null  int64 
dtypes: int64(7), object(9)
memory usage: 1.7+ MB
```

In [14]:
```
merged_df.isna().sum()
```

Out[14]:
```
unique_id            0
birth_year           0
workclass          705
fnlwgt               0
marital-status       0
occupation           0
relationship         0
race                 0
sex               1757
capital-gain         0
capital-loss         0
hours-per-week       0
native-country       0
income               0
education            0
education-num        0
dtype: int64
```

In [15]:
```
#There are some data with "?" that were not included as NA, replac
merged_df = merged_df.replace(" ?", pd.NA)
```

In [16]:
```
#Checking that data with "?" is being noticed as NA
merged_df.isna().sum()
```

Out[16]:
```
unique_id            0
birth_year           0
workclass         1495
fnlwgt               0
marital-status       0
occupation         845
relationship         0
race                 0
sex               1757
capital-gain         0
capital-loss         0
hours-per-week       0
native-country     253
income               0
education            0
education-num        0
dtype: int64
```

In [17]:
```
merged_df
```

Out[17]:

	unique_id	birth_year	workclass	fnlwgt	marital-status	occupation
0	b788ecae-24d3-4d2f-be70-291c656cfe35	2006	<NA>	137363	Never-married	<NA>
1	c50f5eec-eee7-4f71-92ad-5fd2baace470	1989	Private	188041	Married-civ-spouse	Exec-managerial
2	42dbe1ab-593f-4d41-b72f-32bb2081c64c	1999	Private	178478	Never-married	Adm-clerical
3	008c8da5-1688-4b85-a0d0-0516405c84d0	1981	Private	177905	Married-civ-spouse	Handlers-cleaners
4	32084da0-d75e-4f29-8c86-b7d120cdb759	1988	<NA>	389850	Married-spouse-absent	<NA>
...	...	...	...	...	...	...
14142	0b97458e-e408-4956-a7b2-6e5f071906c5	1984	Self-emp-not-inc	209833	Married-civ-spouse	Craft-repair
14143	d5c5f912-84e0-4149-8bf6-c4776a1dad57	1982	Private	102085	Divorced	Other-service
14144	81694b3e-67a3-4c81-912e-be419e1408ea	1994	Private	39386	Married-civ-spouse	Exec-managerial
14145	7cdd164a-81f3-436c-abe9-c8cb5795f586	1982	Private	37869	Married-civ-spouse	Craft-repair
14146	7e514127-4ac2-40ec-b971-a8113e4302a0	1982	Federal-gov	34218	Married-civ-spouse	Exec-managerial

14147 rows × 16 columns

In [18]:

```
merged_df.groupby("race")["native-country"].count()
```

Out[18]:
```
race
Amer-Indian-Eskimo      127
Asian-Pac-Islander      423
Black                  1364
Other                   107
White                 11873
Name: native-country, dtype: int64
```

In [19]:
```
new_df = merged_df.dropna(subset=["native-country"])
```

There are only 253 data which does not have "native-country", which is less than 5% of the whole dataset so dropping them would not cause too much bias

In [20]:
```
#Checking if the "native-country" imputation worked
new_df.isnull().sum()
```

Out[20]:
```
unique_id            0
birth_year           0
workclass         1473
fnlwgt               0
marital-status       0
occupation         834
relationship         0
race                 0
sex               1724
capital-gain         0
capital-loss         0
hours-per-week       0
native-country       0
income               0
education            0
education-num        0
dtype: int64
```

In [21]:
```
new_df.shape
```

Out[21]:
```
(13894, 16)
```

In [22]:
```
#to avoid any sex bias, filled all NA sex as "Unknown"
new_df["sex"] = new_df["sex"].fillna("Unknown")
```

```
/tmp/ipykernel_29/696224373.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df["sex"] = new_df["sex"].fillna("Unknown")
```

In [23]:
```
#Checking if it worked
new_df.isna().sum()
```

Out[23]:
```
unique_id            0
birth_year           0
workclass         1473
fnlwgt               0
marital-status       0
occupation         834
relationship         0
race                 0
sex                  0
capital-gain         0
capital-loss         0
hours-per-week       0
native-country       0
income               0
education            0
education-num        0
dtype: int64
```

In [24]:
```
new_df
```

Out[24]:

	unique_id	birth_year	workclass	fnlwgt	marital-status	occupation
0	b788ecae-24d3-4d2f-be70-291c656cfe35	2006	<NA>	137363	Never-married	<NA>
1	c50f5eec-eee7-4f71-92ad-5fd2baace470	1989	Private	188041	Married-civ-spouse	Exec-managerial
2	42dbe1ab-593f-4d41-b72f-32bb2081c64c	1999	Private	178478	Never-married	Adm-clerical
3	008c8da5-1688-4b85-a0d0-0516405c84d0	1981	Private	177905	Married-civ-spouse	Handlers-cleaners
4	32084da0-d75e-4f29-8c86-b7d120cdb759	1988	<NA>	389850	Married-spouse-absent	<NA>
...	...	...	...	...	...	...
14142	0b97458e-e408-4956-a7b2-6e5f071906c5	1984	Self-emp-not-inc	209833	Married-civ-spouse	Craft-repair
14143	d5c5f912-84e0-4149-8bf6-c4776a1dad57	1982	Private	102085	Divorced	Other-service
14144	81694b3e-67a3-4c81-912e-be419e1408ea	1994	Private	39386	Married-civ-spouse	Exec-managerial
14145	7cdd164a-81f3-436c-abe9-c8cb5795f586	1982	Private	37869	Married-civ-spouse	Craft-repair
14146	7e514127-4ac2-40ec-b971-a8113e4302a0	1982	Federal-gov	34218	Married-civ-spouse	Exec-managerial

13894 rows × 16 columns

In [25]:

```
# Impute workclass within each education group
new_df['workclass'] = (
```

```
    new_df.groupby(['education'], group_keys=False)['workclass']
    .apply(lambda x: x.fillna(x.mode()[0] if not x.mode().empty el
)
```

```
/tmp/ipykernel_29/1531582221.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  new_df['workclass'] = (
```

In [26]:
```
# Impute occupation within each workclass × education group
new_df['occupation'] = (
    new_df.groupby(['workclass', 'education'], group_keys=False)['
    .apply(lambda x: x.fillna(x.mode()[0] if not x.mode().empty el
)
```

```
/tmp/ipykernel_29/22025586.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  new_df['occupation'] = (
```

To find the other 2 missing rows which are "workclass" and "occupation", to find the missing values for "workclass" I used groupby "education". Since "occupation" is more niche than "workclass", I found "workclass" first then use groupby "workclass" and "education" to find "occupation",

In [27]:
```
#Checking if the imputation worked
new_df.isnull().sum()
```

Out[27]:
```
unique_id         0
birth_year        0
workclass         0
fnlwgt            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
education         0
education-num     0
dtype: int64
```

In [28]:
```
#Checking the new imputed data
```

```
new_df
```

Out[28]:

	unique_id	birth_year	workclass	fnlwgt	marital-status	occupation
0	b788ecae-24d3-4d2f-be70-291c656cfe35	2006	Private	137363	Never-married	Adm-clerical
1	c50f5eec-eee7-4f71-92ad-5fd2baace470	1989	Private	188041	Married-civ-spouse	Exec-managerial
2	42dbe1ab-593f-4d41-b72f-32bb2081c64c	1999	Private	178478	Never-married	Adm-clerical
3	008c8da5-1688-4b85-a0d0-0516405c84d0	1981	Private	177905	Married-civ-spouse	Handlers-cleaners
4	32084da0-d75e-4f29-8c86-b7d120cdb759	1988	Private	389850	Married-spouse-absent	Craft-repair
...	...	...	...	...	...	...
14142	0b97458e-e408-4956-a7b2-6e5f071906c5	1984	Self-emp-not-inc	209833	Married-civ-spouse	Craft-repair
14143	d5c5f912-84e0-4149-8bf6-c4776a1dad57	1982	Private	102085	Divorced	Other-service
14144	81694b3e-67a3-4c81-912e-be419e1408ea	1994	Private	39386	Married-civ-spouse	Exec-managerial
14145	7cdd164a-81f3-436c-abe9-c8cb5795f586	1982	Private	37869	Married-civ-spouse	Craft-repair
14146	7e514127-4ac2-40ec-b971-a8113e4302a0	1982	Federal-gov	34218	Married-civ-spouse	Exec-managerial

13894 rows × 16 columns

In [29]:
```python
for col in ["workclass", "occupation"]:
    print(f"\nValue counts for {col}:")
    print(merged_df[col].value_counts(dropna=False).head(20))
```

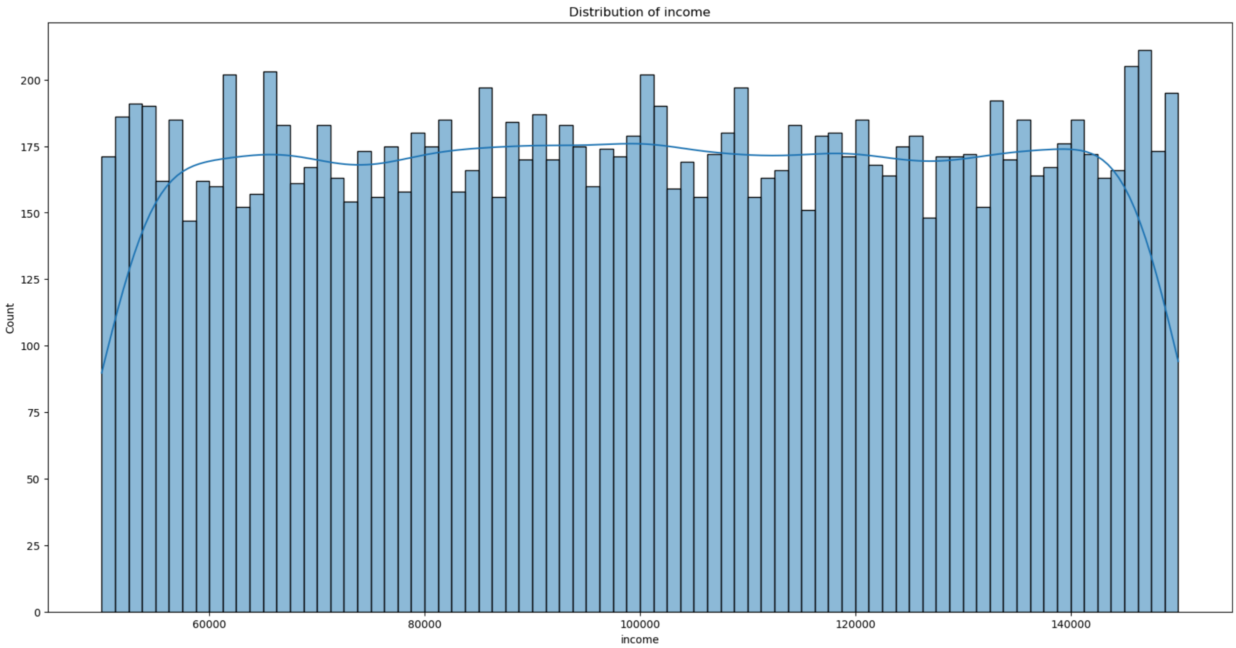
```
Value counts for workclass:
workclass
 Private             9362
 Self-emp-not-inc    1009
 Local-gov            879
<NA>                  790
NaN                   705
 State-gov            520
 Self-emp-inc         470
 Federal-gov          399
 Never-worked           7
 Without-pay            6
Name: count, dtype: int64

Value counts for occupation:
occupation
 Prof-specialty       1811
 Craft-repair         1741
 Exec-managerial      1733
 Adm-clerical         1625
 Sales                1582
 Other-service        1454
 Machine-op-inspct     848
<NA>                   845
 Transport-moving      692
 Handlers-cleaners     625
 Farming-fishing       416
 Tech-support          402
 Protective-serv       290
 Priv-house-serv        79
 Armed-Forces            4
Name: count, dtype: int64
```

In [30]:
```python
#Checking value counts for "workclass" and "occupation"
for col in ["workclass", "occupation"]:
    print(f"\nValue counts for {col}:")
    print(new_df[col].value_counts(dropna=False).head(20))
```

```
Value counts for workclass:
workclass
Private              10665
Self-emp-not-inc      1000
Local-gov              864
State-gov              511
Self-emp-inc           449
Federal-gov            392
Never-worked             7
Without-pay              6
Name: count, dtype: int64

Value counts for occupation:
occupation
 Craft-repair         1968
 Adm-clerical         1869
 Prof-specialty       1802
 Exec-managerial      1773
 Other-service        1583
 Sales                1554
 Machine-op-inspct     866
 Transport-moving      680
 Handlers-cleaners     615
 Farming-fishing       416
 Tech-support          393
 Protective-serv       288
 Priv-house-serv        76
Unknown                  7
 Armed-Forces            4
Name: count, dtype: int64
```

Checking whether my imputation inflates one variable greatly. In the new dataset note that workclass "private" has 10,665 rows which is dominating the whole dataset, however originally in workclass "private" already has 9,362 rows, meaning that the imputation is not inflating "private" by a significant amount. Every other variable is not greatly inflated, meaning the imputation will not result into any bias.

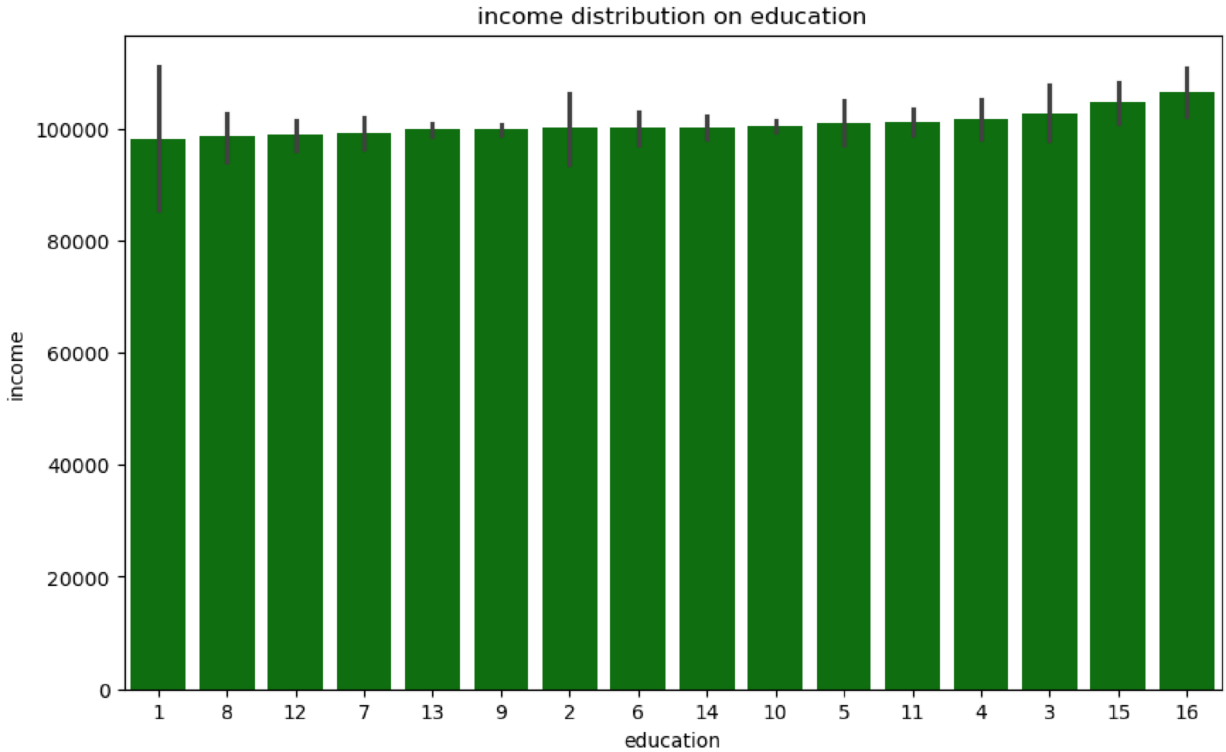
In [58]:
```
plt.figure(figsize=(20, 10))
sns.histplot(new_df['income'], kde=True, bins=80)
plt.title('Distribution of income')
plt.show()
```



Making sure the data cleaning does not change the income distribution shape.

2. EXPLORATORY DATA ANALYSIS

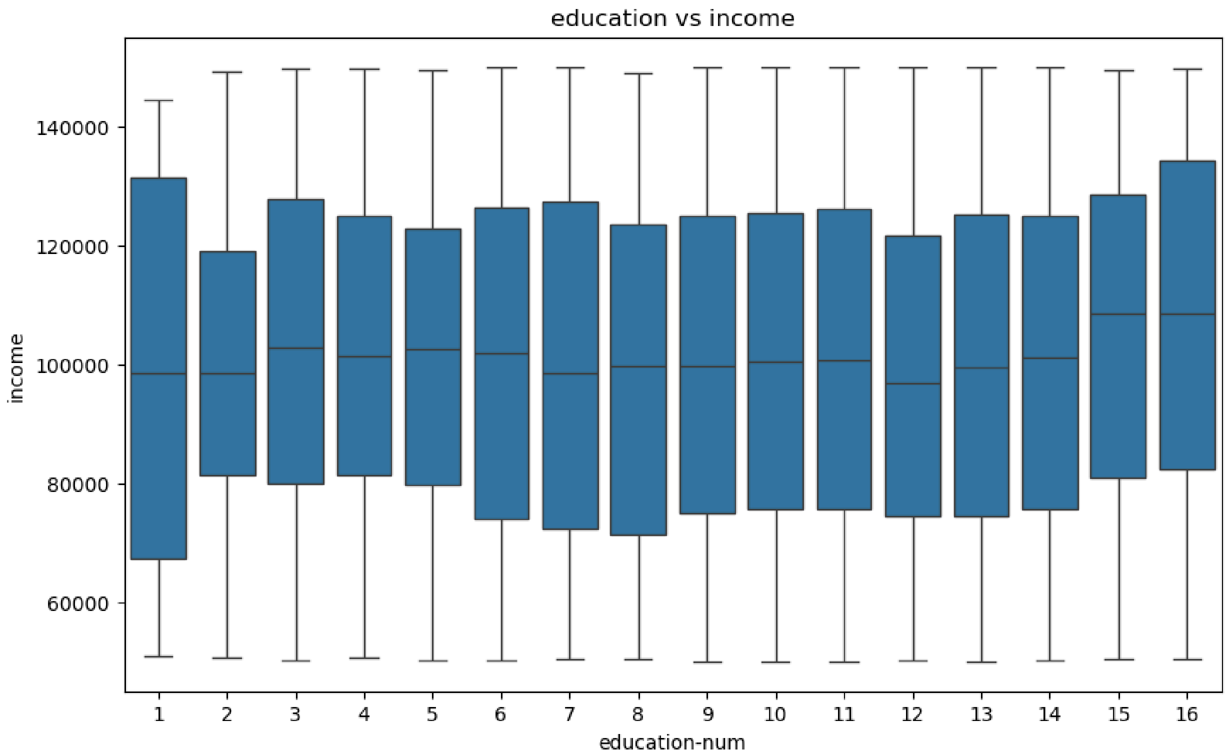
In [31]:
```
#Education income analysis
order1 = new_df.groupby("education-num")["income"].mean().sort_val

plt.figure(figsize=(10, 6))
sns.barplot(x='education-num', y='income', data=new_df, order=orde
plt.title("income distribution on education")
plt.xlabel('education')
plt.ylabel('income')
plt.show()
```



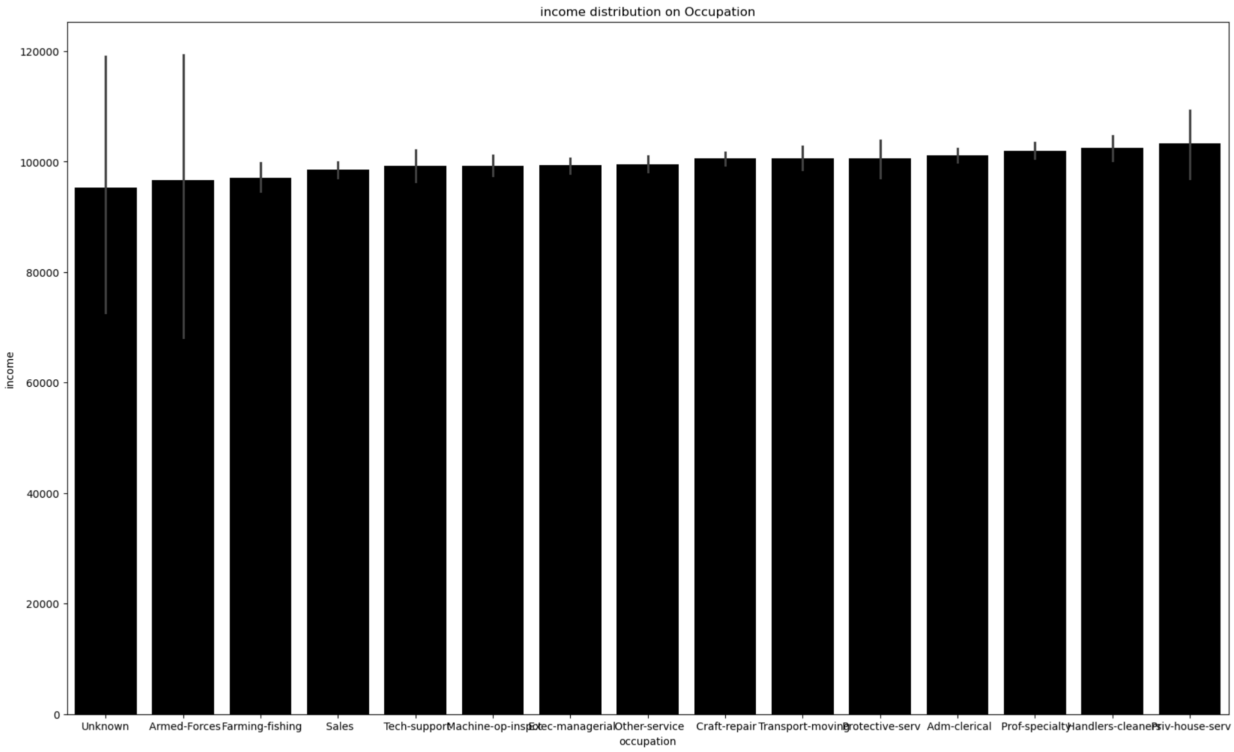
Income distribution on Education - Probably if not the most then one of the biggest drivers of Income. The higher Education shows higher income, in this case Education is categorized as numbers, the higher the number the higher the Education level, 1 being lowest and 16 being highest. On this graph 15 and 16 has the highest income on average and 1 having the lowest income which makes sense, however in between 1-15 all the numbers are quite unstructured and does not have a clear pattern, meaning someone with education of 3 has a higher average income than someone who has an education of 8 or even 12.

In [32]:
```
#Checking if there are any outliers
plt.figure(figsize=(10, 6))
plt.figure(figsize=(10, 6))
sns.boxplot(x='education-num', y='income', data=new_df)
plt.title('education vs income')
plt.show()
```

<Figure size 1000x600 with 0 Axes>



In [33]:
```
#Occupation income analysis
order2 = new_df.groupby("occupation")["income"].mean().sort_values
plt.figure(figsize=(20, 12))
sns.barplot(x='occupation', y='income', data=new_df, color="black"
plt.title("income distribution on Occupation")
plt.xlabel('occupation')
plt.ylabel('income')
plt.show()
```



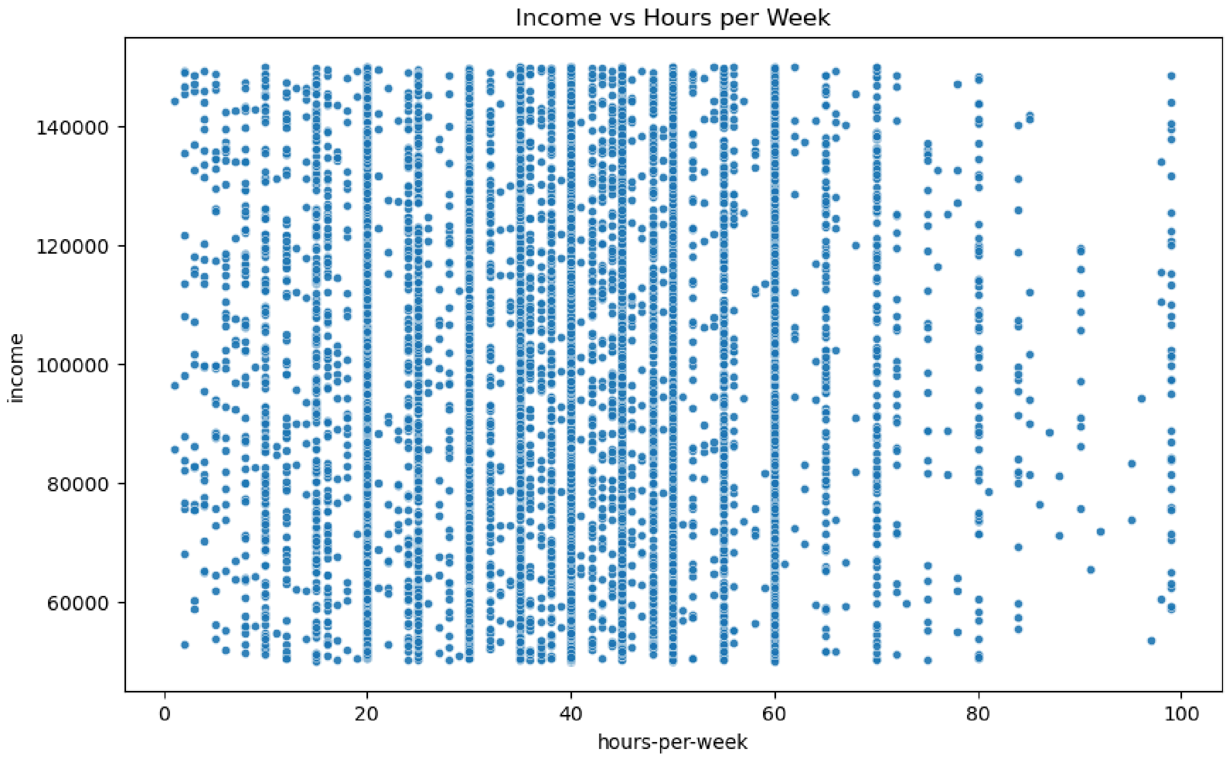
This shows the occupation to income graph, private house services shows the most earnings compared to every other occupation.

In [34]:
```
#Hours worked and occupation to income analysis
avg_income = new_df.groupby(["occupation","hours-per-week"])["inco

avg_income.sort_values("income", ascending=False).head(10)
```

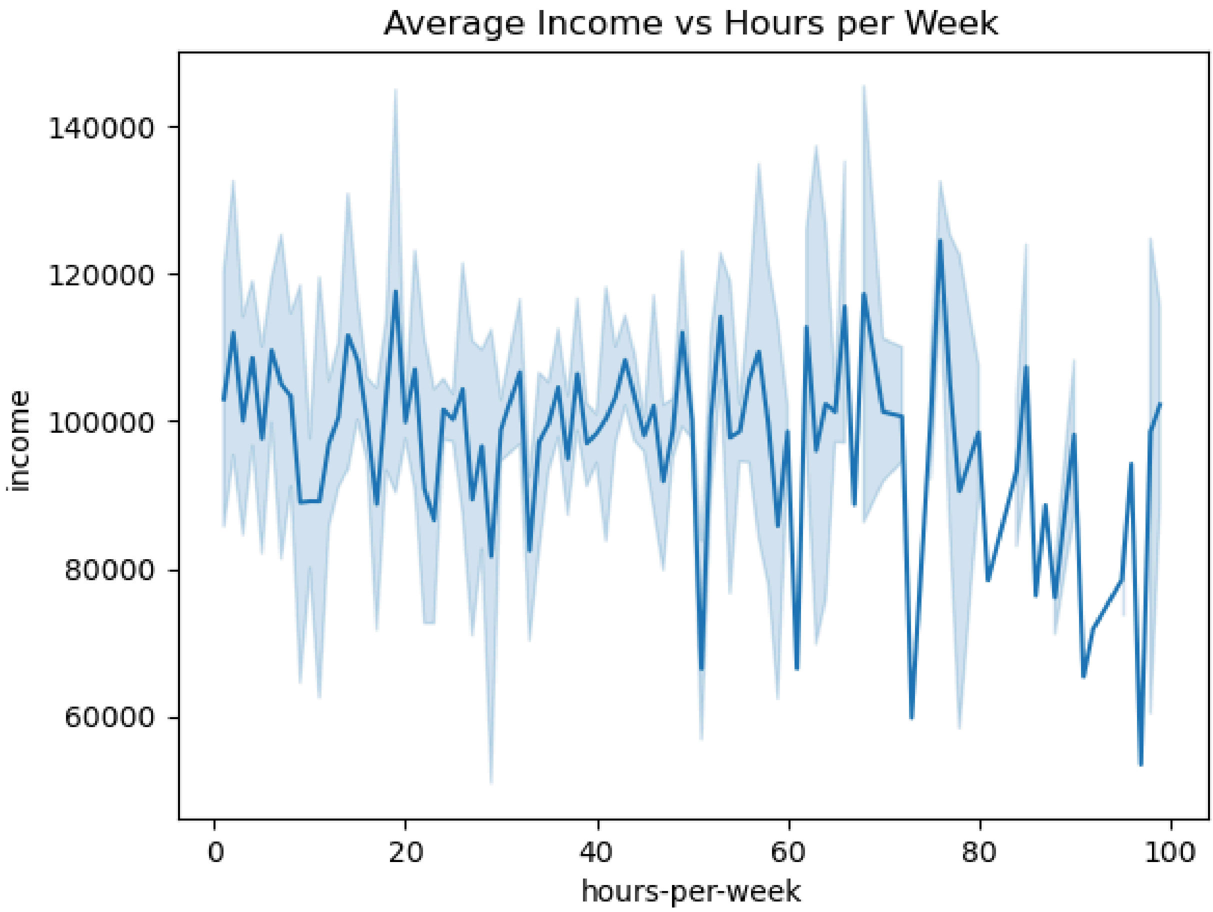
Out[34]:

	occupation	hours-per-week	income
194	Farming-fishing	38	149462.0
209	Farming-fishing	66	149141.0
267	Machine-op-inspct	2	149006.0
181	Farming-fishing	15	146859.0
537	Sales	46	146312.0
572	Tech-support	22	146266.0
491	Protective-serv	68	145376.0
408	Prof-specialty	19	144859.0
577	Tech-support	32	144759.0
460	Protective-serv	14	144466.0

In [35]:
```
#Scatterplot for hours worked analysis
plt.figure(figsize=(10,6))
sns.scatterplot(x="hours-per-week", y="income", data=new_df,
                alpha=0.9, s=20)
plt.title("Income vs Hours per Week")
plt.show()
```



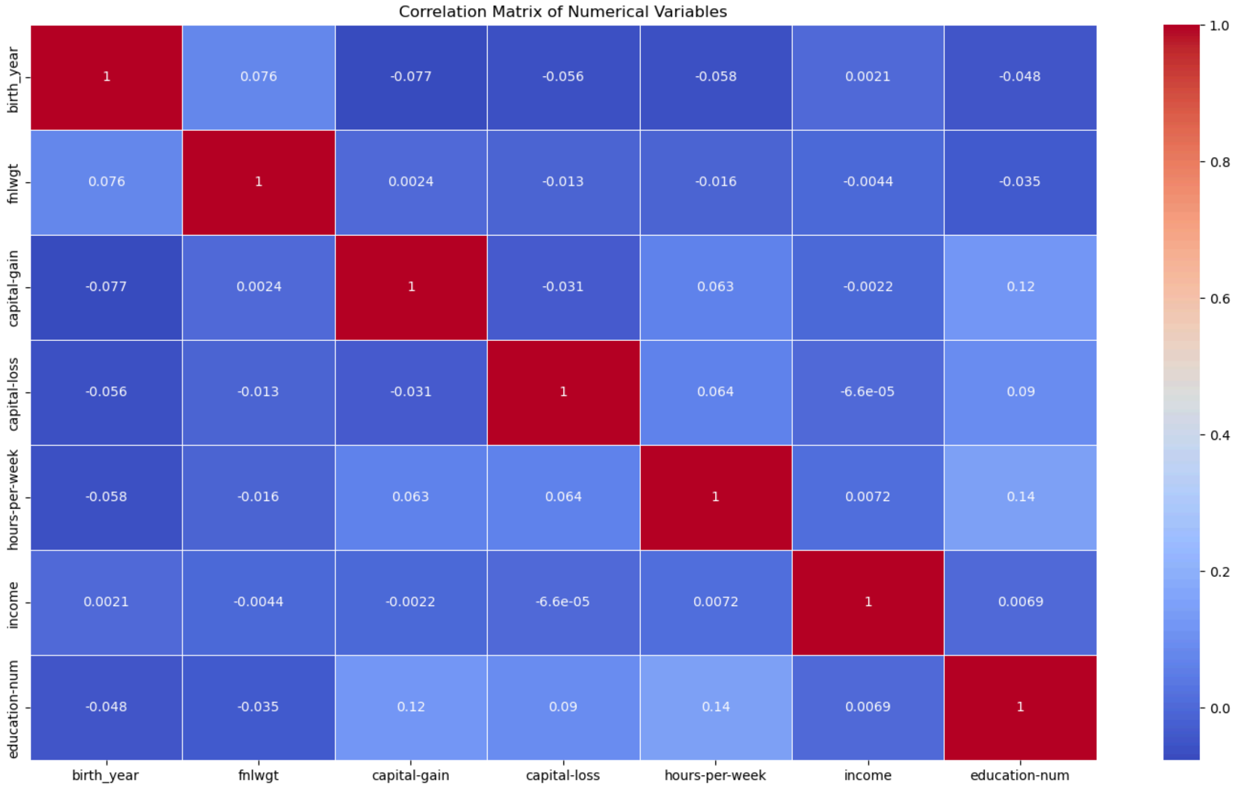
In [36]:
```
#Line graph for hours worked analysis
sns.lineplot(x="hours-per-week", y="income", data=avg_income)
plt.title("Average Income vs Hours per Week")
plt.show()
```



Hours worked and occupation analysis shows the correlation of each occupation and how much they work and their mean income, there isn't a clear pattern of any occupations that has more hours will guarantee more income. Some occupation works for 15 hours but would earn more than an employee that works 68 hours. There is also no clear pattern if hours actually effect income.

In [37]:
```
# Select numerical columns
numerical_columns = new_df.select_dtypes(include=[np.number]).colu

# Compute correlation matrix
correlation_matrix = new_df[numerical_columns].corr()
```

In [38]:
```
# Visualize correlation matrix
plt.figure(figsize=(18, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linew
plt.title('Correlation Matrix of Numerical Variables')
plt.show()
```

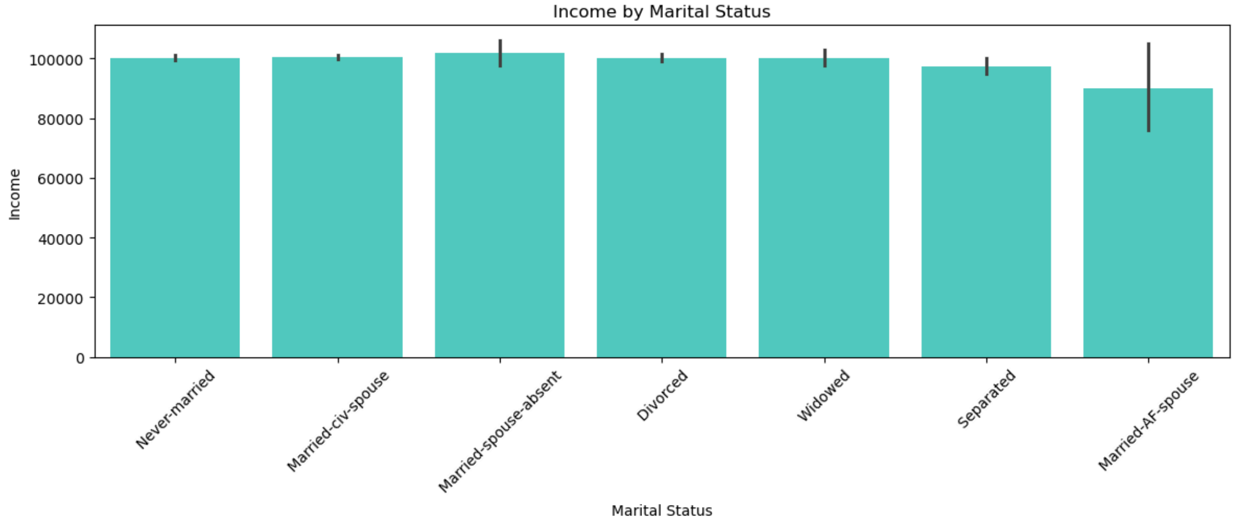


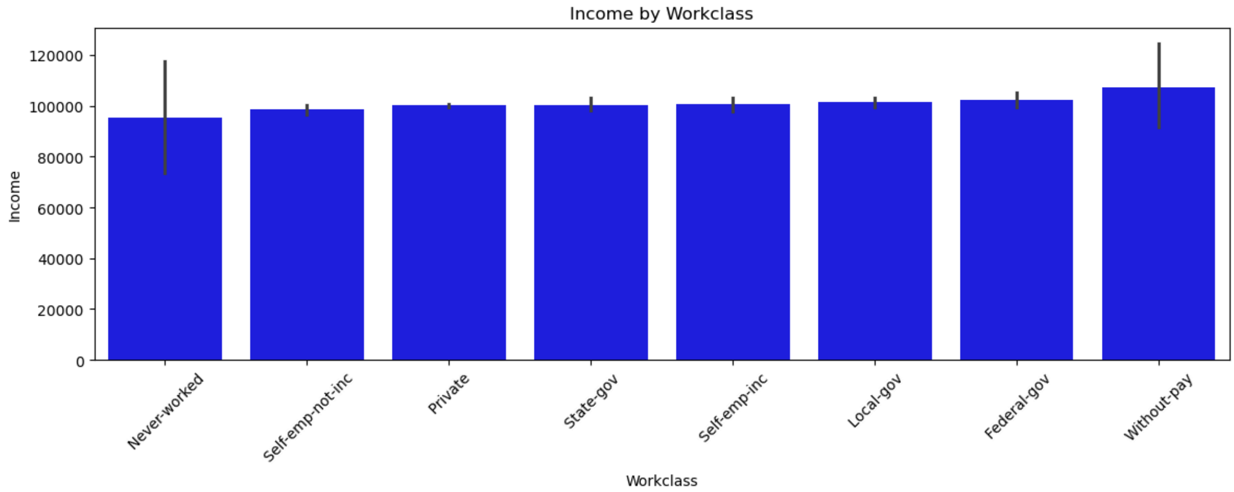
In [39]:
```
avg_metrics = new_df.groupby('education-num').agg({
    'income': 'mean',
    'hours-per-week': 'mean'
}).round(2)

print("\nAverage metrics by education:")
print(avg_metrics)
```

```
Average metrics by education:
                    income  hours-per-week
education-num
1                 97990.79           39.08
2                100014.77           38.73
3                102735.34           38.52
4                101643.77           38.63
5                100793.90           38.69
6                100045.75           36.03
7                 99200.21           34.65
8                 98591.38           36.18
9                 99890.92           40.51
10               100404.00           38.73
11               101049.26           41.83
12                98866.64           40.61
13                99760.20           42.37
14               100178.58           43.50
15               104563.03           47.69
16               106481.78           45.67
```

Average metrics shows the comparison of income and hours worked accross the education levels. Working hours varies from each education levels, meaning there might be different workloads for each education levels.

In [40]:
```
fig, axes = plt.subplots(2, 1, figsize=(12,10))

sns.barplot(x='marital-status', y='income', data=new_df,
            color="turquoise", ax=axes[0])
axes[0].set_title("Income by Marital Status")
axes[0].set_xlabel("Marital Status"); axes[0].set_ylabel("Income")
axes[0].tick_params(axis='x', rotation=45)

order = new_df.groupby("workclass")["income"].mean().sort_values()
sns.barplot(x='workclass', y='income', data=new_df,
            order=order, color="blue", ax=axes[1])
axes[1].set_title("Income by Workclass")
axes[1].set_xlabel("Workclass"); axes[1].set_ylabel("Income")
axes[1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```





These 2 analysis shows the distribution of income to workclass and an employee's marital status. By average someone who is "never married" has the highest income compared to other marital status. With workclass someone who is in "without-pay" workclass earns the most compared to every other workclasses.

3. PREDICTIVE MODELING

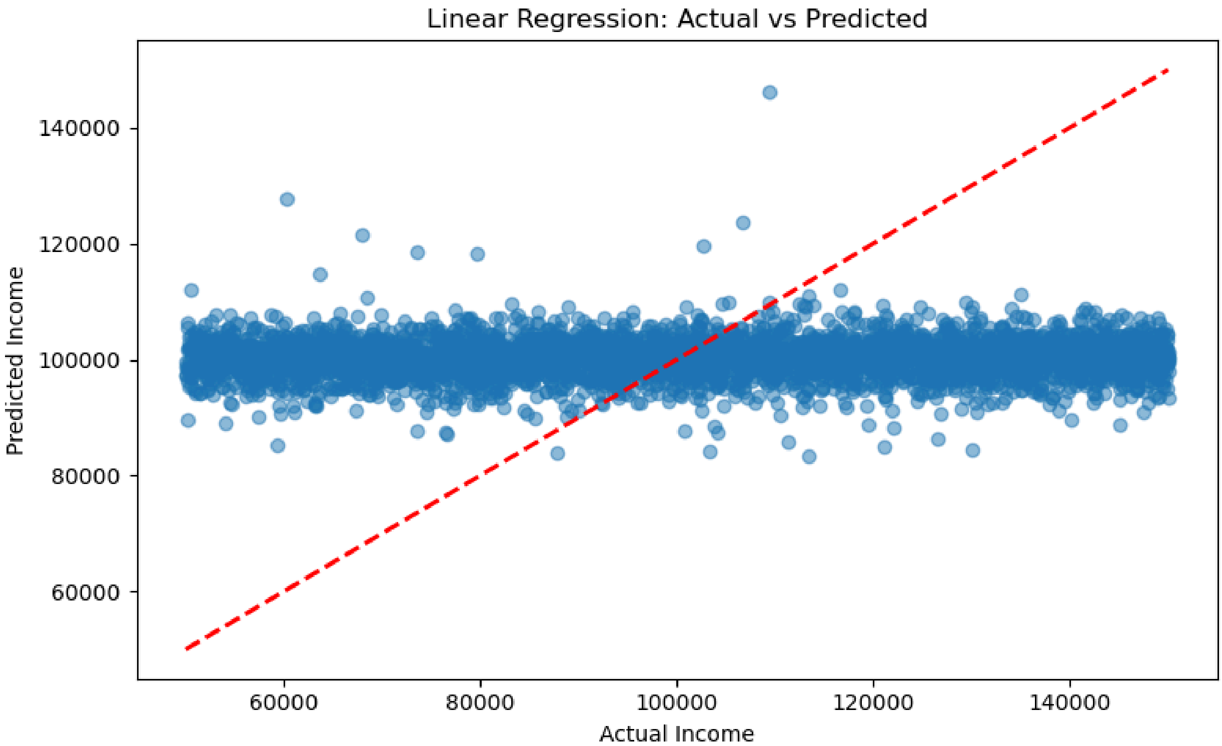
In [41]:
```
# Select features for prediction, now including neighbourhood
features = ['workclass', 'hours-per-week', 'education-num', "occup

# Create a new dataframe with only the selected features and price
df_selected = new_df[features + ['income']]


# Convert categorical variables to dummy variables
df_encoded = pd.get_dummies(df_selected, columns=['workclass', "oc

# Separate features (X) and target variable (y)
X = df_encoded.drop('income', axis=1)
y = df_encoded['income']
```

In [42]:
```
#Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_siz
```

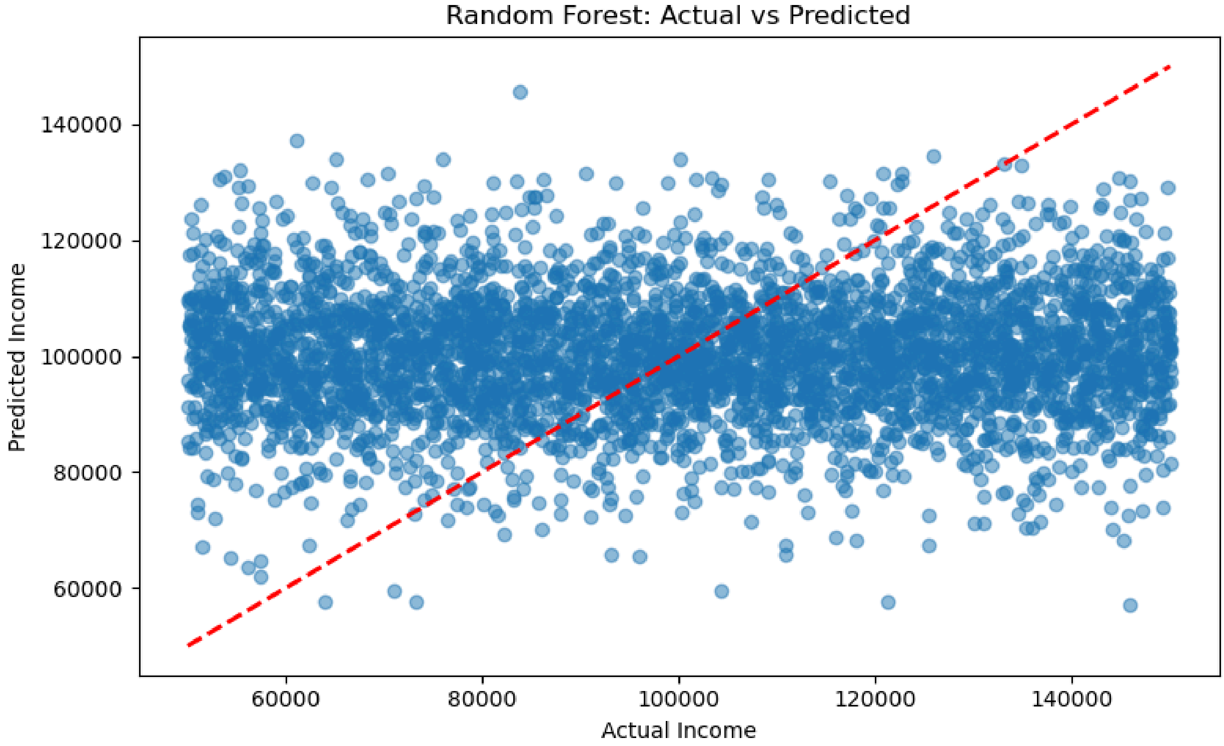
In [57]:
```
#Linear regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
```

In [44]:
```
#Random forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

In [51]:
```
plt.figure(figsize=(8, 5))
plt.scatter(y_test, lr_predictions, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()
plt.xlabel("Actual Income")
plt.ylabel("Predicted Income")
plt.title("Linear Regression: Actual vs Predicted")

plt.tight_layout()
plt.show()
```



In [50]:
```
# Visualize predictions vs actual
plt.figure(figsize=(8, 5))
plt.scatter(y_test, rf_predictions, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()
plt.xlabel("Actual Income")
plt.ylabel("Predicted Income")
plt.title("Random Forest: Actual vs Predicted")

plt.tight_layout()
plt.show()
```

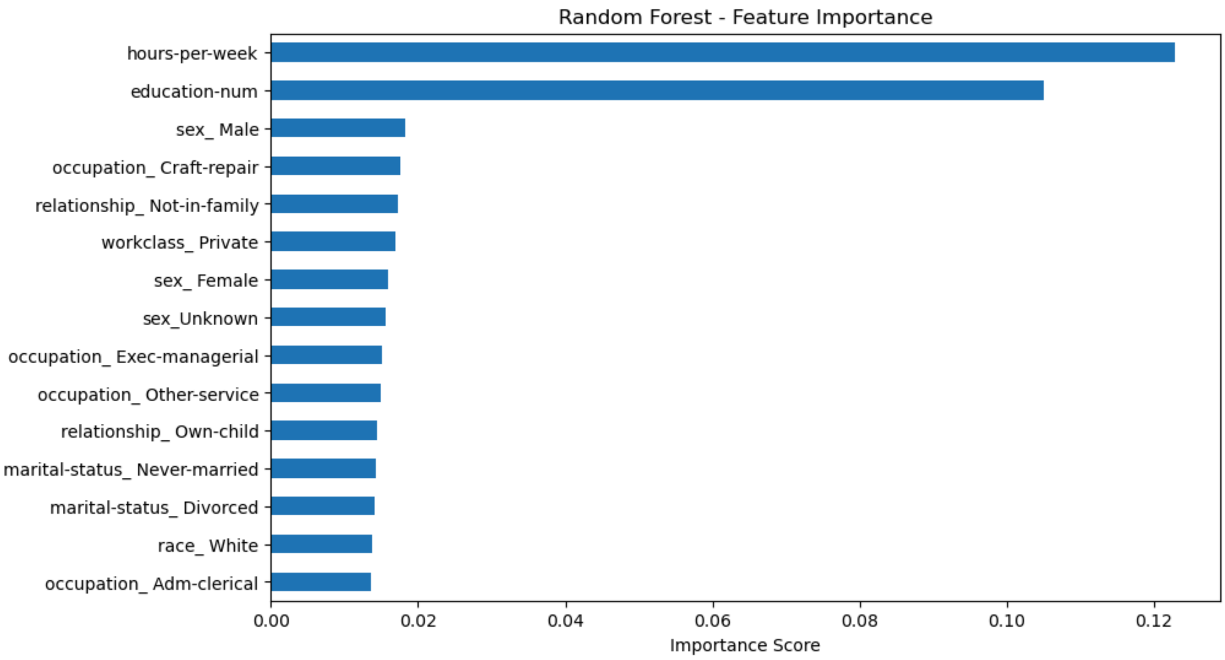


In [45]:
```
#Evaluations of the models built
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mape = mean_absolute_percentage_error(y_true, y_pred)

    print(f"{model_name} Results:")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
    print(f"Mean Absolute Percentage Error (MAPE): {mape:.2%}")
    print("-"*50)

evaluate_model(y_test, rf_predictions, "Random Forest")
evaluate_model(y_test, lr_predictions, "Linear Regression")
```

```
Random Forest Results:
Mean Absolute Error (MAE): 26149.28
Mean Squared Error (MSE): 953829481.77
Root Mean Squared Error (RMSE): 30884.13
Mean Absolute Percentage Error (MAPE): 30.06%
--------------------------------------------------
Linear Regression Results:
Mean Absolute Error (MAE): 25156.09
Mean Squared Error (MSE): 851025465.88
Root Mean Squared Error (RMSE): 29172.34
Mean Absolute Percentage Error (MAPE): 28.98%
--------------------------------------------------
```

This is the result of the models, MAPE is considered to be moderate, it does not represent the best model but it also doesn't represent the worst model, however this model is still acceptable because as the data suggests it is difficult to predict income. The distribution from the dataset itself is not normal, in the real world income is very difficult to predict there are so much more intangible variables that influences salary.

In [46]:
```
rf_importances = pd.Series(
    rf_model.feature_importances_,
    index=X_train.columns
).sort_values(ascending=False)

# Plot top 15
plt.figure(figsize=(10,6))
rf_importances.head(15).plot(kind="barh")
plt.gca().invert_yaxis()
plt.title("Random Forest - Feature Importance")
plt.xlabel("Importance Score")
plt.show()
```



EMAIL

Subject: Salary Analysis Insights – Key Drivers at TechNova

Dear Head of HR Analytics,

As part of our ongoing review of compensation equity, I have analysed employee data using Linear Regression to find coefficients interpretability and Random Forest to capture non-linear and complex patterns, both to understand what drives salary at TechNova.

Our models achieved an average prediction error (MAPE) of ~29–30%. This level of error reflects the complexity and variability of salaries, there might be variables that weren't included like intangible considerations, however it still provides reliable insight into the bigger drivers of salaries.

The Random Forest feature importance analysis highlights hours worked per week and education level as the strongest predictors of salary. Employees who consistently work longer hours and those with higher education levels are more likely to earn higher incomes. Additional drivers include occupation type, marital status, and sector of employment (workclass), which differentiate salary outcomes across the company.

Key findings:

Hours per week is the single largest contributor to salary differences, suggesting workload intensity is strongly tied to compensation.

Education has a clear positive relationship with income, confirming its role in career progression.

Occupation (specific job roles) and workclass (employment sector) provide additional explanation for salary gaps.

Demographics such as sex and relationship status appear with smaller but notable effects, reinforcing the need for equity monitoring.

Some actionable insights are around the consistency of worked hours and occupation, some individuals earn more with less hours. Age should also be a driving factor as experience should be taken into account.

In summary, TechNova's pay strategy is most influenced by work intensity and educational attainment, with occupation and sector playing secondary roles.

Best regards, Aurelius Johnsson