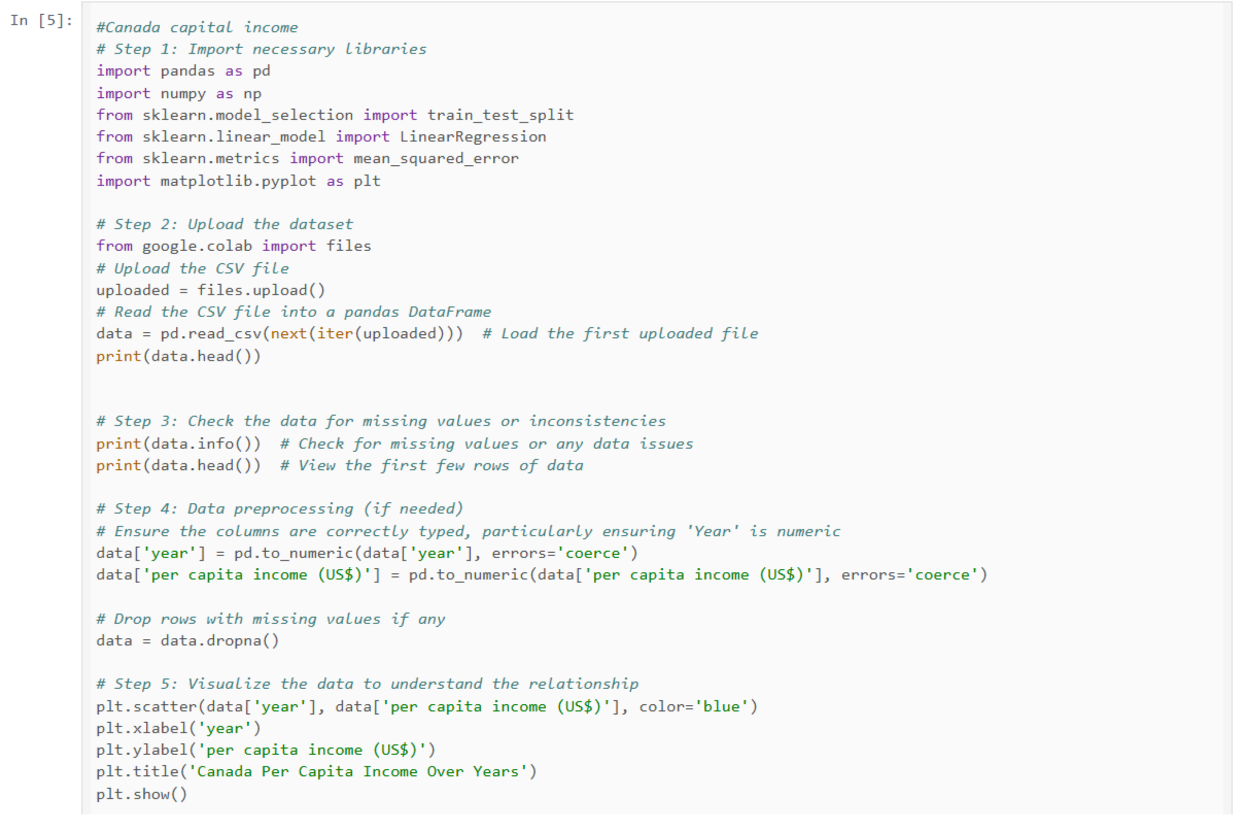
Lab 2

LINEAR MULTIPLICATION REGRESSION:



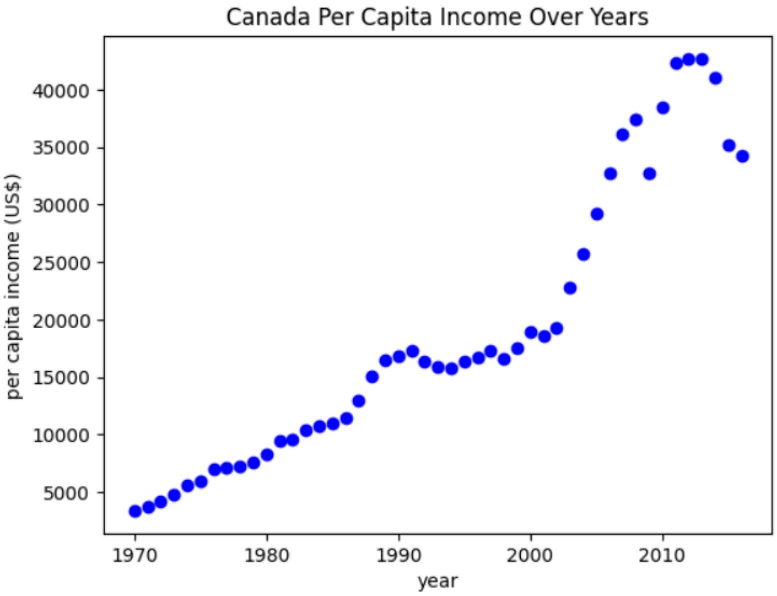
In [5]:

```
#Canada capital income
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Step 2: Upload the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
data = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(data.head())


# Step 3: Check the data for missing values or inconsistencies
print(data.info())  # Check for missing values or any data issues
print(data.head())  # View the first few rows of data

# Step 4: Data preprocessing (if needed)
# Ensure the columns are correctly typed, particularly ensuring 'Year' is numeric
data['year'] = pd.to_numeric(data['year'], errors='coerce')
data['per capita income (US$)'] = pd.to_numeric(data['per capita income (US$)'], errors='coerce')

# Drop rows with missing values if any
data = data.dropna()

# Step 5: Visualize the data to understand the relationship
plt.scatter(data['year'], data['per capita income (US$)'], color='blue')
plt.xlabel('year')
plt.ylabel('per capita income (US$)')
plt.title('Canada Per Capita Income Over Years')
plt.show()
```

```
# Step 6: Prepare data for training
X = data['year'].values.reshape(-1, 1)  # Year as feature
y = data['per capita income (US$)'].values  # Per Capita Income as target

# Step 7: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 8: Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 9: Evaluate the model (optional, just for understanding the accuracy)
y_pred = model.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")

# Step 10: Predict per capita income for the year 2020
year_2020 = np.array([[2020]])
predicted_income_2020 = model.predict(year_2020)
print(f"Predicted per capita income in 2020: ${predicted_income_2020[0]:,.2f}")

# Step 11: (Optional) Plot the regression line along with the data points
plt.scatter(X, y, color='blue')
plt.plot(X, model.predict(X), color='red', linewidth=2)
plt.xlabel('year')
plt.ylabel('per capita income (US$)')
plt.title('Linear Regression: Canada Per Capita Income Over Years')
plt.show()
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

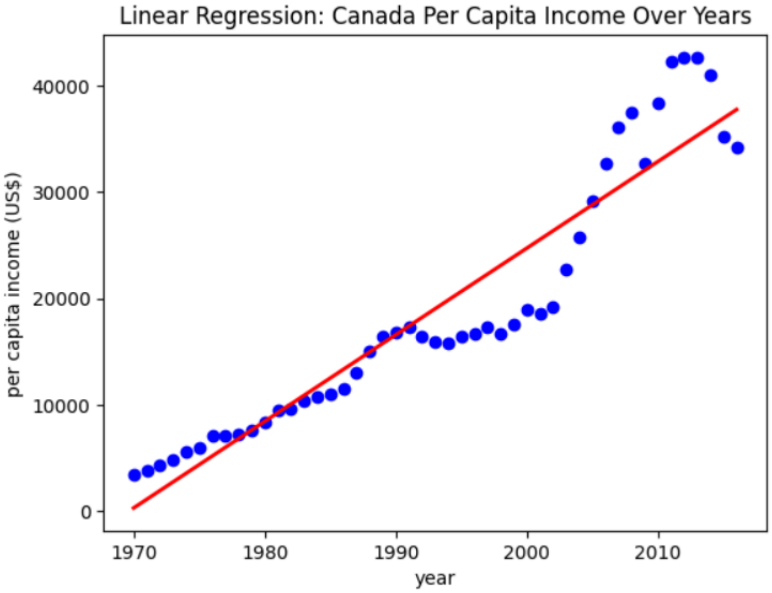
```
Saving canada_per_capita_income.csv to canada_per_capita_income (2).csv
   year  per capita income (US$)
0  1970              3399.299037
1  1971              3768.297935
2  1972              4251.175484
3  1973              4804.463248
4  1974              5576.514583
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 2 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   year                     47 non-null     int64  
 1   per capita income (US$)  47 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 884.0 bytes
None
   year  per capita income (US$)
0  1970              3399.299037
1  1971              3768.297935
2  1972              4251.175484
3  1973              4804.463248
4  1974              5576.514583
```



```
Mean Squared Error: 15147815.5477862
Predicted per capita income in 2020: $41,027.68
```

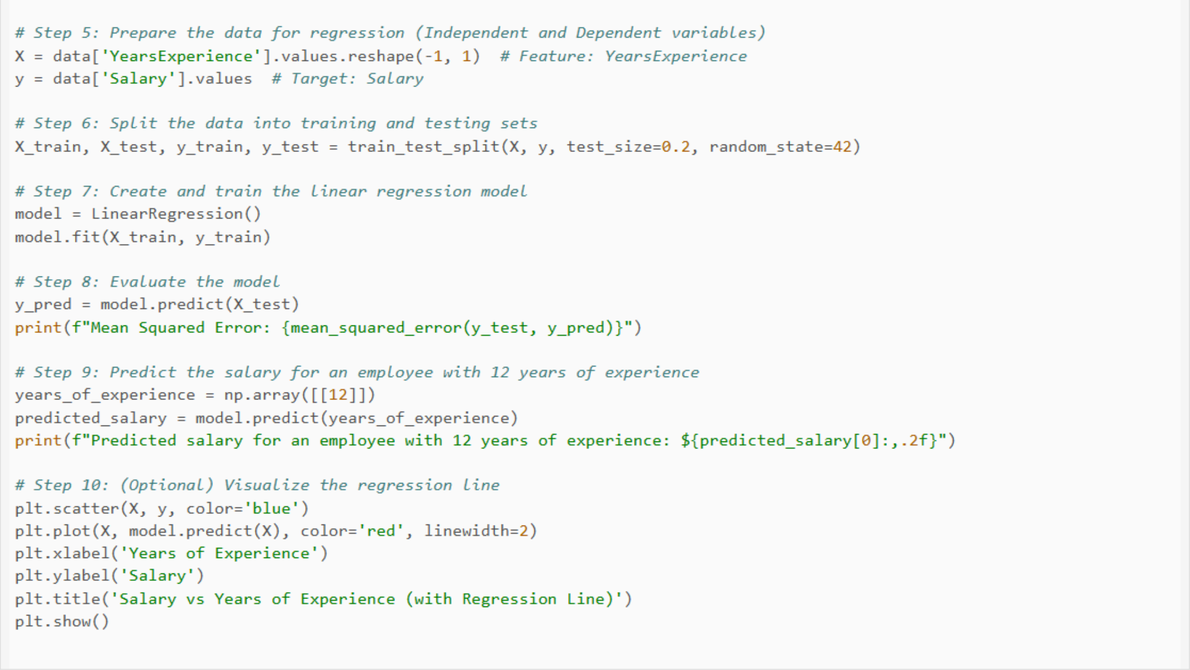
year

```
Mean Squared Error: 15147815.5477862
Predicted per capita income in 2020: $41,027.68
```

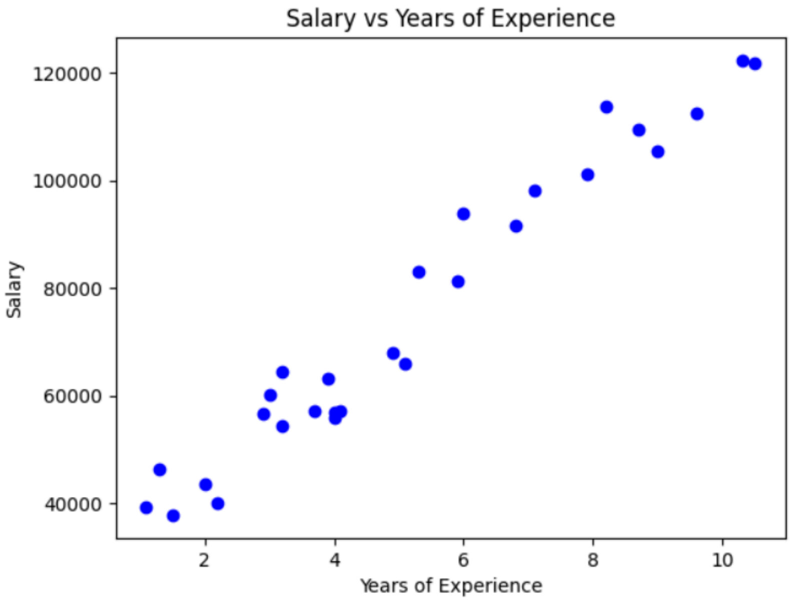


In [6]:

```
#salary
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Step 2: Load the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
data = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(data.head())

# Step 3: Data preprocessing
# Check for any missing values
print(data.info())  # Check the data info for missing values and types
print(data.head())  # Preview the first few rows

# Ensure the columns are of numeric types (YearsExperience and Salary)
data['YearsExperience'] = pd.to_numeric(data['YearsExperience'], errors='coerce')
data['Salary'] = pd.to_numeric(data['Salary'], errors='coerce')

# Drop rows with missing values if any
data = data.dropna()

# Step 4: Visualize the data
plt.scatter(data['YearsExperience'], data['Salary'], color='blue')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience')
plt.show()
```

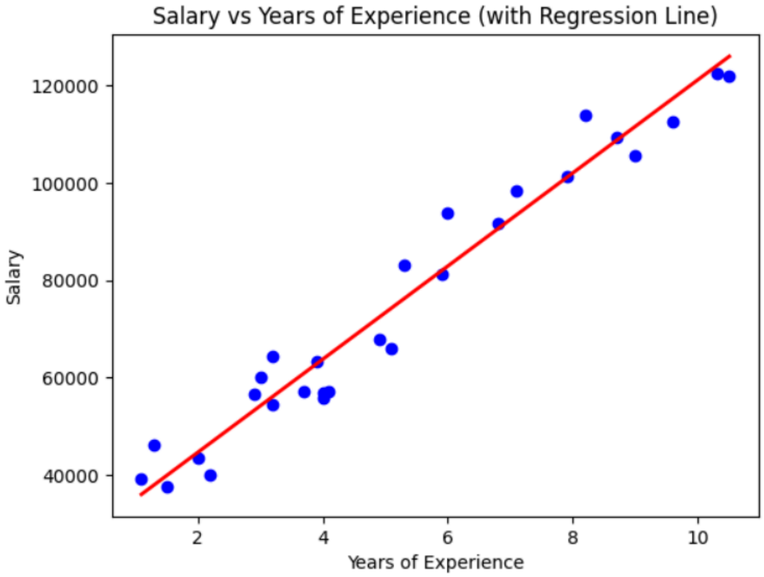


```
# Step 5: Prepare the data for regression (Independent and Dependent variables)
X = data['YearsExperience'].values.reshape(-1, 1)  # Feature: YearsExperience
y = data['Salary'].values  # Target: Salary

# Step 6: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 7: Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 8: Evaluate the model
y_pred = model.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")

# Step 9: Predict the salary for an employee with 12 years of experience
years_of_experience = np.array([[12]])
predicted_salary = model.predict(years_of_experience)
print(f"Predicted salary for an employee with 12 years of experience: ${predicted_salary[0]:,.2f}")

# Step 10: (Optional) Visualize the regression line
plt.scatter(X, y, color='blue')
plt.plot(X, model.predict(X), color='red', linewidth=2)
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience (with Regression Line)')
plt.show()
```



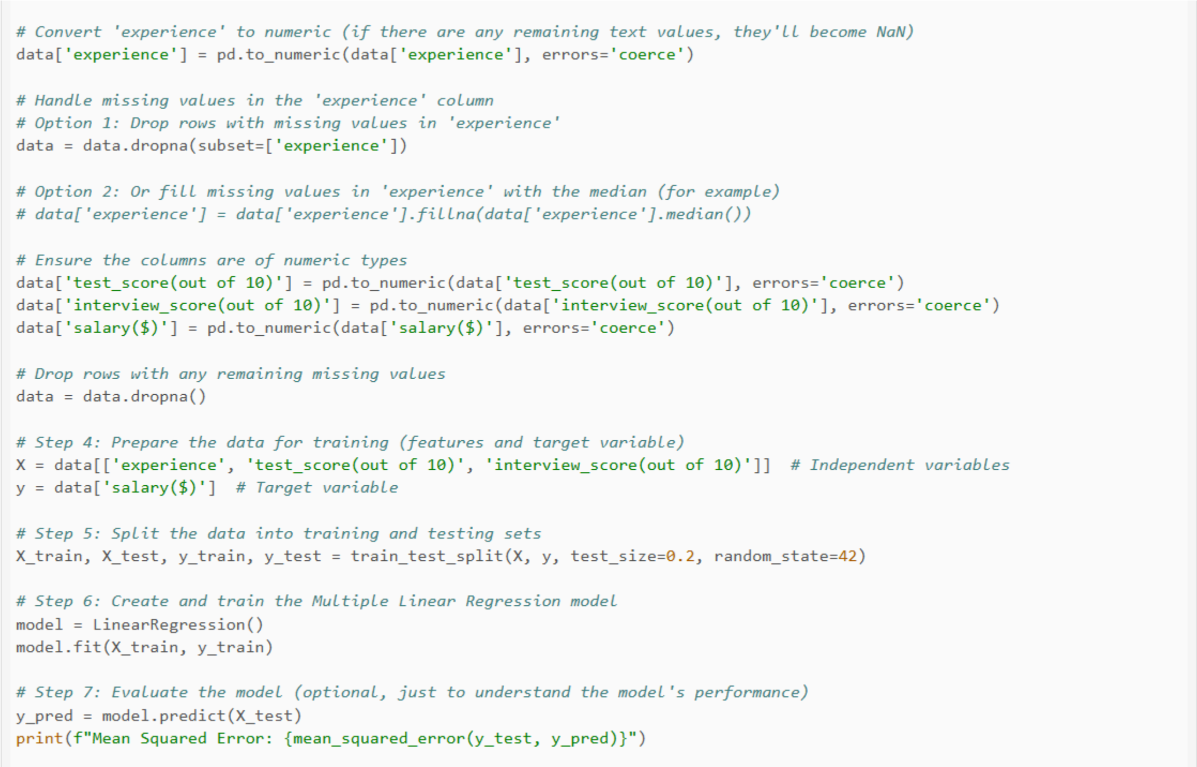
Years of Experience

```
Mean Squared Error: 27180506.800821673
Predicted salary for an employee with 12 years of experience: $140,337.54
```

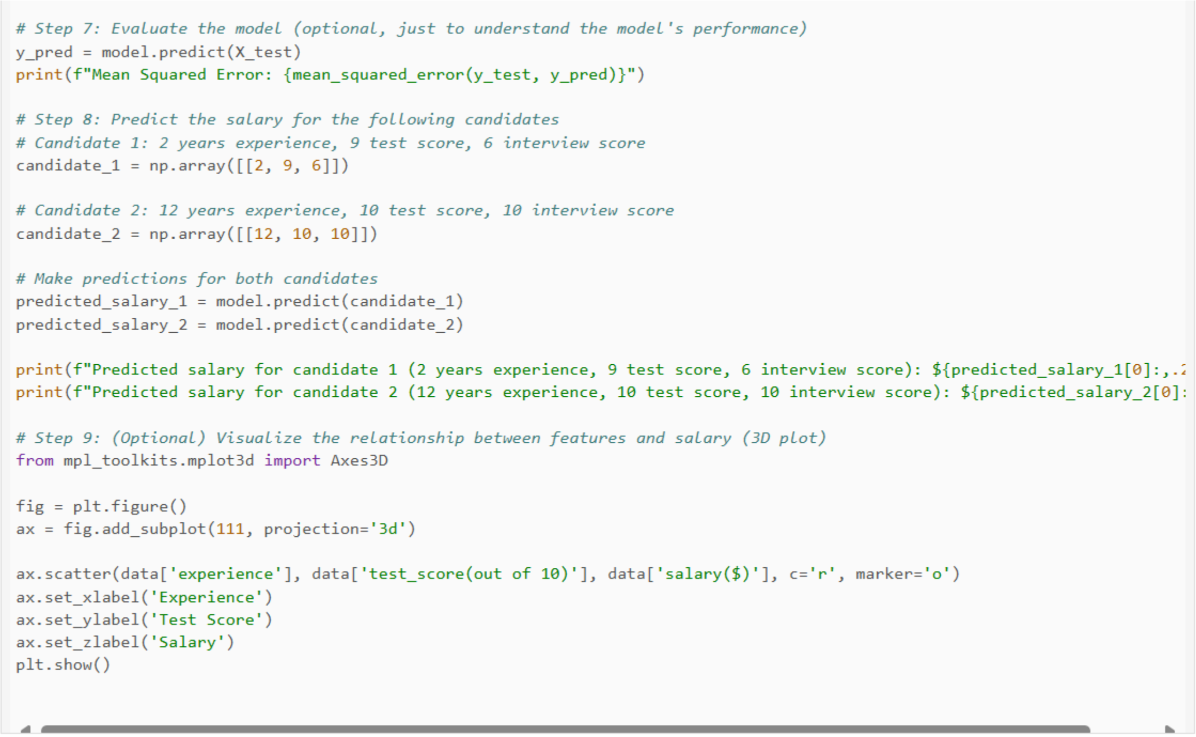


In [8]:

```
#hiring
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Step 2: Load the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
data = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(data.head())

# Step 3: Data preprocessing

# Handle non-numeric values in the 'experience' column by mapping text to numeric
experience_map = {
    'five': 5,
    'two': 2,
    'seven': 7
}

# Replace textual experience values with their numeric counterparts
data['experience'] = data['experience'].replace(experience_map)

# Convert 'experience' to numeric (if there are any remaining text values, they'll become NaN)
data['experience'] = pd.to_numeric(data['experience'], errors='coerce')
```



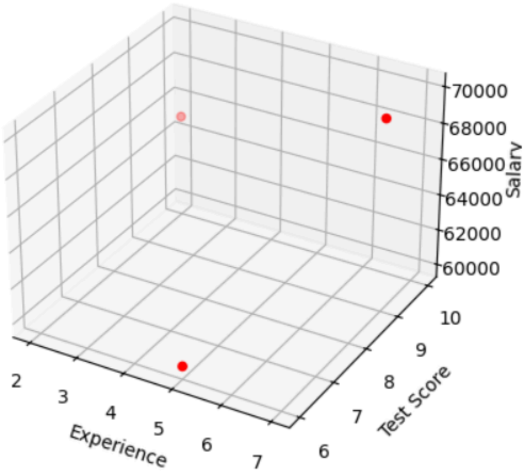
```python
# Convert 'experience' to numeric (if there are any remaining text values, they'll become NaN)
data['experience'] = pd.to_numeric(data['experience'], errors='coerce')

# Handle missing values in the 'experience' column
# Option 1: Drop rows with missing values in 'experience'
data = data.dropna(subset=['experience'])

# Option 2: Or fill missing values in 'experience' with the median (for example)
# data['experience'] = data['experience'].fillna(data['experience'].median())

# Ensure the columns are of numeric types
data['test_score(out of 10)'] = pd.to_numeric(data['test_score(out of 10)'], errors='coerce')
data['interview_score(out of 10)'] = pd.to_numeric(data['interview_score(out of 10)'], errors='coerce')
data['salary($)'] = pd.to_numeric(data['salary($)'], errors='coerce')

# Drop rows with any remaining missing values
data = data.dropna()

# Step 4: Prepare the data for training (features and target variable)
X = data[['experience', 'test_score(out of 10)', 'interview_score(out of 10)']]  # Independent variables
y = data['salary($)']  # Target variable

# Step 5: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 6: Create and train the Multiple Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 7: Evaluate the model (optional, just to understand the model's performance)
y_pred = model.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
```



```python
# Step 7: Evaluate the model (optional, just to understand the model's performance)
y_pred = model.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")

# Step 8: Predict the salary for the following candidates
# Candidate 1: 2 years experience, 9 test score, 6 interview score
candidate_1 = np.array([[2, 9, 6]])

# Candidate 2: 12 years experience, 10 test score, 10 interview score
candidate_2 = np.array([[12, 10, 10]])

# Make predictions for both candidates
predicted_salary_1 = model.predict(candidate_1)
predicted_salary_2 = model.predict(candidate_2)

print(f"Predicted salary for candidate 1 (2 years experience, 9 test score, 6 interview score): ${predicted_salary_1[0]:,.2
print(f"Predicted salary for candidate 2 (12 years experience, 10 test score, 10 interview score): ${predicted_salary_2[0]:

# Step 9: (Optional) Visualize the relationship between features and salary (3D plot)
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(data['experience'], data['test_score(out of 10)'], data['salary($)'], c='r', marker='o')
ax.set_xlabel('Experience')
ax.set_ylabel('Test Score')
ax.set_zlabel('Salary')
plt.show()
```

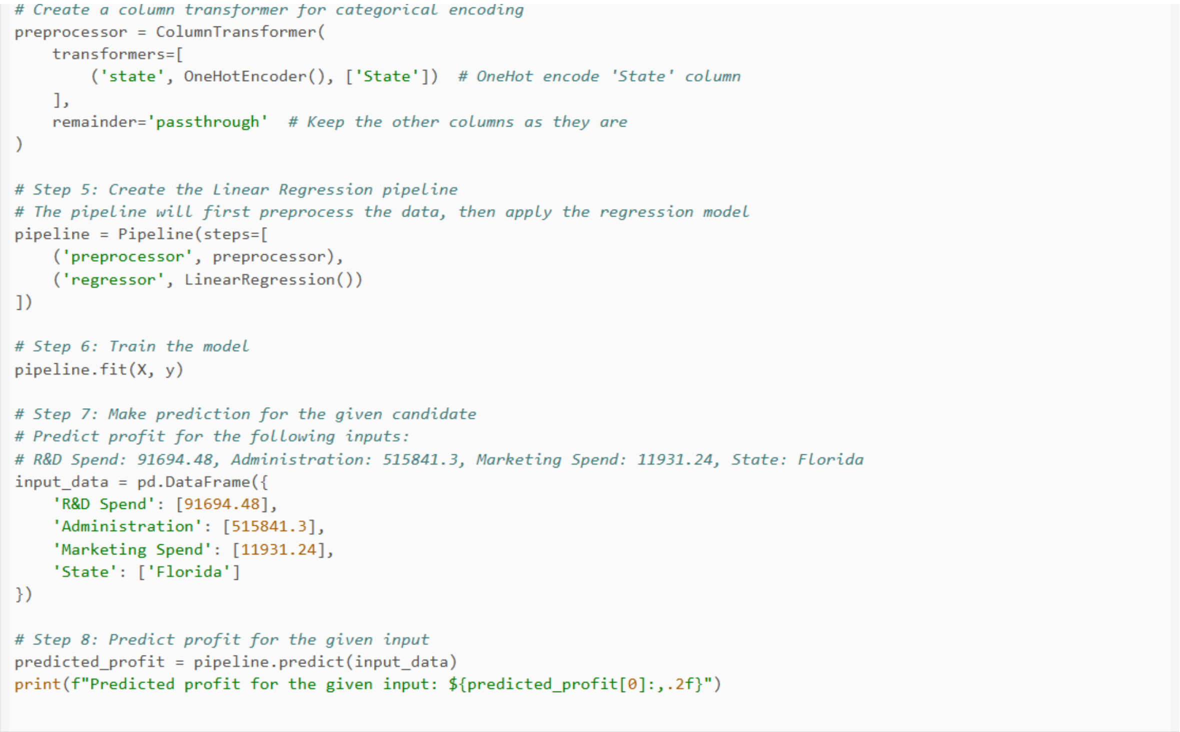
```
0        NaN                  8.0                              9       50000
1        NaN                  8.0                              6       45000
2       five                  6.0                              7       60000
3        two                 10.0                             10       65000
4      seven                  9.0                              6       70000
Mean Squared Error: 75524376.41723369
Predicted salary for candidate 1 (2 years experience, 9 test score, 6 interview score): $67,023.81
Predicted salary for candidate 2 (12 years experience, 10 test score, 10 interview score): $70,952.38
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
```

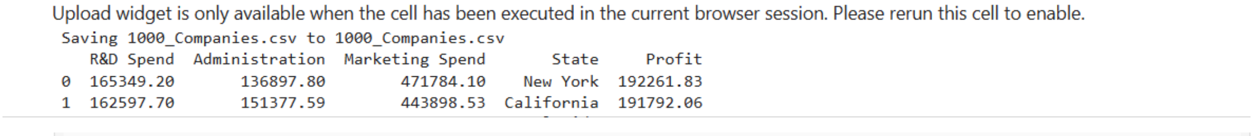


In [9]:

```
#1000 companies
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error

# Step 2: Load the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
data = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(data.head())

# Step 3: Data Preprocessing
# Check for missing values
print(data.info())  # Check the data info for missing values and types
print(data.head())  # Preview the first few rows

# Separate features and target variable
X = data[['R&D Spend', 'Administration', 'Marketing Spend', 'State']]  # Features
y = data['Profit']  # Target variable

# Step 4: Handle categorical data for 'State' column using One-Hot Encoding
# The 'State' column is categorical and needs to be converted to numeric values
# Using OneHotEncoder to convert 'State' into numeric format

# Create a column transformer for categorical encoding
preprocessor = ColumnTransformer(
    transformers=[
```

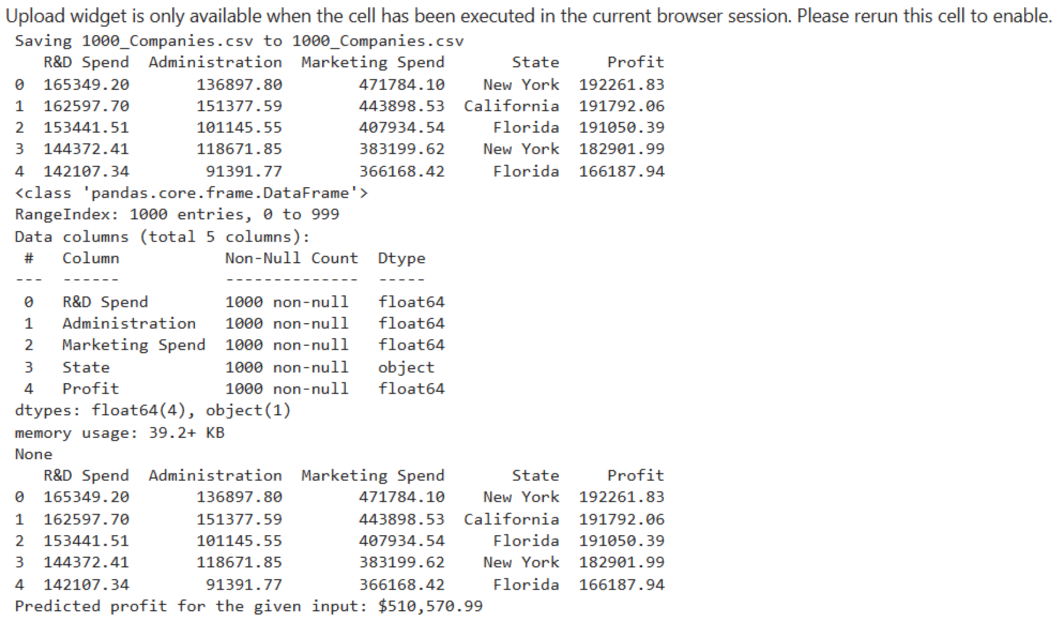


```python
# Create a column transformer for categorical encoding
preprocessor = ColumnTransformer(
    transformers=[
        ('state', OneHotEncoder(), ['State'])  # OneHot encode 'State' column
    ],
    remainder='passthrough'  # Keep the other columns as they are
)

# Step 5: Create the Linear Regression pipeline
# The pipeline will first preprocess the data, then apply the regression model
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Step 6: Train the model
pipeline.fit(X, y)

# Step 7: Make prediction for the given candidate
# Predict profit for the following inputs:
# R&D Spend: 91694.48, Administration: 515841.3, Marketing Spend: 11931.24, State: Florida
input_data = pd.DataFrame({
    'R&D Spend': [91694.48],
    'Administration': [515841.3],
    'Marketing Spend': [11931.24],
    'State': ['Florida']
})

# Step 8: Predict profit for the given input
predicted_profit = pipeline.predict(input_data)
print(f"Predicted profit for the given input: ${predicted_profit[0]:,.2f}")
```



Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 1000_Companies.csv to 1000_Companies.csv
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
```



Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 1000_Companies.csv to 1000_Companies.csv
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        1000 non-null   float64
 1   Administration   1000 non-null   float64
 2   Marketing Spend  1000 non-null   float64
 3   State            1000 non-null   object 
 4   Profit           1000 non-null   float64
dtypes: float64(4), object(1)
memory usage: 39.2+ KB
None
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
Predicted profit for the given input: $510,570.99
```

In [12]:

```
#Linear-Regression-Housing_Area_Price.ipynb

import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt

# Step 2: Load the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(df.head())

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')

new_df = df.drop('price',axis='columns')
new_df

price = df.price
price

# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(new_df,price)

"""(1) Predict price of a home with area = 3300 sqr ft"""

reg.predict([[3300]])

reg.coef
```

```
reg.predict([[3300]])

reg.coef_

reg.intercept_

"""Y = m * X + b (m is coefficient and b is intercept)"""

3300*135.78767123 + 180616.43835616432

"""(1) Predict price of a home with area = 5000 sqr ft"""

reg.predict([[5000]])
```

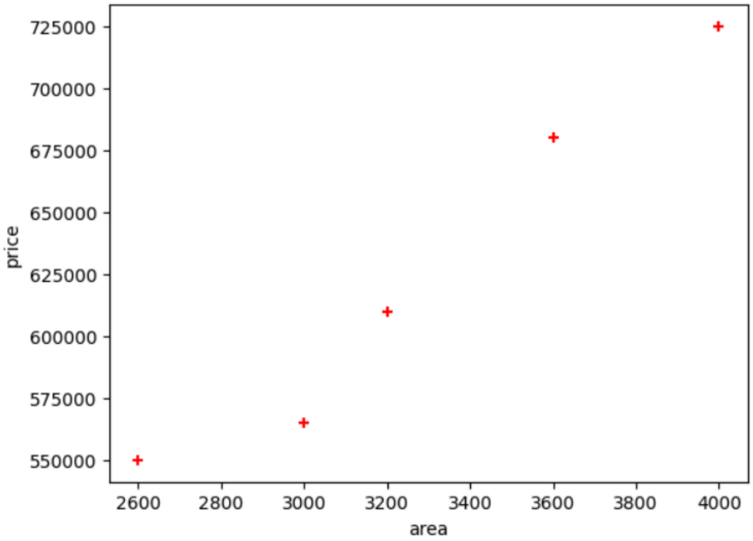
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving housing_area_price.csv to housing_area_price (1).csv
   area   price
0  2600  550000
1  3000  565000
2  3200  610000
3  3600  680000
4  4000  725000
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
warnings.warn(
```

Out[12]:

```
array([859554.79452055])
```



In [13]:

```
#Multiple_LR_HomePrice.ipynb

import pandas as pd
import numpy as np
from sklearn import linear_model

# Step 2: Load the dataset
from google.colab import files
# Upload the CSV file
uploaded = files.upload()
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(next(iter(uploaded)))  # Load the first uploaded file
print(df.head())

"""Data Preprocessing: Fill NA values with median value of a column"""

df.bedrooms.median()

df.bedrooms = df.bedrooms.fillna(df.bedrooms.median())
df

reg = linear_model.LinearRegression()
reg.fit(df.drop('price',axis='columns'),df.price)

reg.coef_

reg.intercept_

"""Find price of home with 3000 sqr ft area, 3 bedrooms, 40 year old"""

reg.predict([[3000, 3, 40]])

112.06244194*3000 + 23388.88007794*3 + -3231.71790863*40 + 221323.00186540384
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving homeprices_Multiple_LR.csv to homeprices_Multiple_LR.csv
   area  bedrooms  age   price
0  2600       3.0   20  550000
1  3000       4.0   15  565000
2  3200       NaN   18  610000
3  3600       3.0   30  595000
4  4000       5.0    8  760000
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, b
ut LinearRegression was fitted with feature names
  warnings.warn(
```

Out[13]:

```
498408.25157402386
```