```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 1000_Companies.csv

- **1000_Companies.csv**(text/csv) - 52203 bytes, last modified: 3/17/2025 - 100% done

```
Saving 1000_Companies.csv to 1000_Companies.csv
```

```
df = pd.read_csv('1000_Companies.csv')
```

```
df
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.800	471784.1000	New York	192261.83000
1	162597.70	151377.590	443898.5300	California	191792.06000
2	153441.51	101145.550	407934.5400	Florida	191050.39000
3	144372.41	118671.850	383199.6200	New York	182901.99000
4	142107.34	91391.770	366168.4200	Florida	166187.94000
...	...	...	...	...	...
995	54135.00	118451.999	173232.6695	California	95279.96251
996	134970.00	130390.080	329204.0228	California	164336.60550
997	100275.47	241926.310	227142.8200	California	413956.48000
998	128456.23	321652.140	281692.3200	California	333962.19000
999	161181.72	270939.860	295442.1700	New York	476485.43000

1000 rows × 5 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
missing_values = df.isnull().sum()

# Display the number of missing values per column
print(f"Missing values in each column:\n{missing_values}")
```

```
Missing values in each column:
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

```
X = df[['R&D Spend', 'Administration', 'Marketing Spend', 'State']]
y = df['Profit']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ('state', OneHotEncoder(), ['State']),
        ('num', 'passthrough', ['R&D Spend', 'Administration', 'Marketing Spend'])
    ])
```

```
df
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.800	471784.1000	New York	192261.83000
1	162597.70	151377.590	443898.5300	California	191792.06000
2	153441.51	101145.550	407934.5400	Florida	191050.39000
3	144372.41	118671.850	383199.6200	New York	182901.99000
4	142107.34	91391.770	366168.4200	Florida	166187.94000
...	...	...	...	...	...
995	54135.00	118451.999	173232.6695	California	95279.96251
996	134970.00	130390.080	329204.0228	California	164336.60550
997	100275.47	241926.310	227142.8200	California	413956.48000
998	128456.23	321652.140	281692.3200	California	333962.19000
999	161181.72	270939.860	295442.1700	New York	476485.43000

1000 rows × 5 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())  # Linear regression model
])
```

```
pipeline.fit(X_train, y_train)
```

```
Pipeline
preprocessor: ColumnTransformer
state
OneHotEncoder
num
passthrough
LinearRegression
```

```
from sklearn.preprocessing import StandardScaler

# Create a pipeline with scaling and regression (if needed)
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('scaler', StandardScaler()),  # Apply scaling to numerical features
    ('regressor', LinearRegression())
])

# Fit the model
pipeline.fit(X_train, y_train)
```

```
Pipeline
preprocessor: ColumnTransformer
state
OneHotEncoder
num
passthrough
StandardScaler
LinearRegression
```

```
new_data = pd.DataFrame({
    'R&D Spend': [91694.48],
```

```
    'Administration': [515841.3],
    'Marketing Spend': [11931.24],
    'State': ['Florida']
})
```

+ Code + Text

```
predicted_profit = pipeline.predict(new_data)
print(f"Predicted Profit: {predicted_profit[0]}")
```

```
Predicted Profit: 554066.3031289799
```