```
Commencez à coder ou à générer avec l'IA.
```

INTRO ML - M1 ISFA 2024- 2025

Import

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from io import StringIO
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from sklearn.inspection import partial_dependence, PartialDependenceDisplay
import time

# For missing values visualization
!pip install missingno
import missingno as msno

# For outlier detection
!pip install pyod
from pyod.models.iforest import IForest

# For QQ plot data calculation
from scipy.stats import probplot

# -------------------------
# Step 0: Create and Load Dataset
# -------------------------
csv_data = """Gender,Age,Salary,Height,City
Male,25,50000,175,New York
Female,30,60000,162,Los Angeles
Male,45,,180,Chicago
Female,35,70000,158,New York
Male,50,120000,185,San Francisco
Female,29,40000,,Los Angeles
Male,33,65000,172,Chicago
Female,44,80000,165,
Male,55,130000,188,New York
Female,39,90000,163,San Francisco
Male,,45000,170,Los Angeles
Female,27,48000,155,Chicago
Male,60,150000,190,New York
Female,22,38000,,San Francisco
Male,46,85000,178,Chicago
Female,,52000,160,Los Angeles
Male,48,92000,,New York
Female,41,75000,167,San Francisco
Male,34,68000,174,Chicago
Female,31,62000,159,New York
Male,28,52000,168,Los Angeles
Female,,59000,156,Chicago
Male,52,100000,182,New York
Female,26,43000,153,San Francisco
Male,54,,186,Chicago
Female,40,85000,161,Los Angeles
Male,47,90000,179,New York
Female,33,70000,164,San Francisco
Male,57,145000,,Chicago
Female,29,42000,157,New York
Male,42,75000,176,Los Angeles
Female,34,64000,160,Chicago
Male,,49000,171,New York
Female,36,79000,,Los Angeles
Male,51,110000,183,San Francisco
Female,25,,154,Chicago
Male,61,155000,192,New York
Female,23,40000,152,San Francisco
Male,49,88000,177,Chicago
Female,38,82000,164,Los Angeles
Male,,70000,173,New York
Female,32,61000,158,San Francisco
Male,44,87000,,Chicago
Female,37,78000,161,New York
Male,29,54000,169,Los Angeles
Female,28,60000,156,Chicago
Male,56,140000,189,New York
Female,24,39000,,San Francisco
Male,58,148000,191,Chicago
Female,21,35000,151,Los Angeles
Male,35,72000,175,New York
Female,33,68000,164,San Francisco
Male,43,82000,176,Chicago
Female,29,50000,159,New York
Male,41,76000,174,Los Angeles
Female,27,62000,,Chicago
Male,59,152000,193,New York
Female,20,36000,150,San Francisco
Male,46,89000,178,Chicago
Female,42,80000,163,Los Angeles
Male,30,56000,169,New York
```

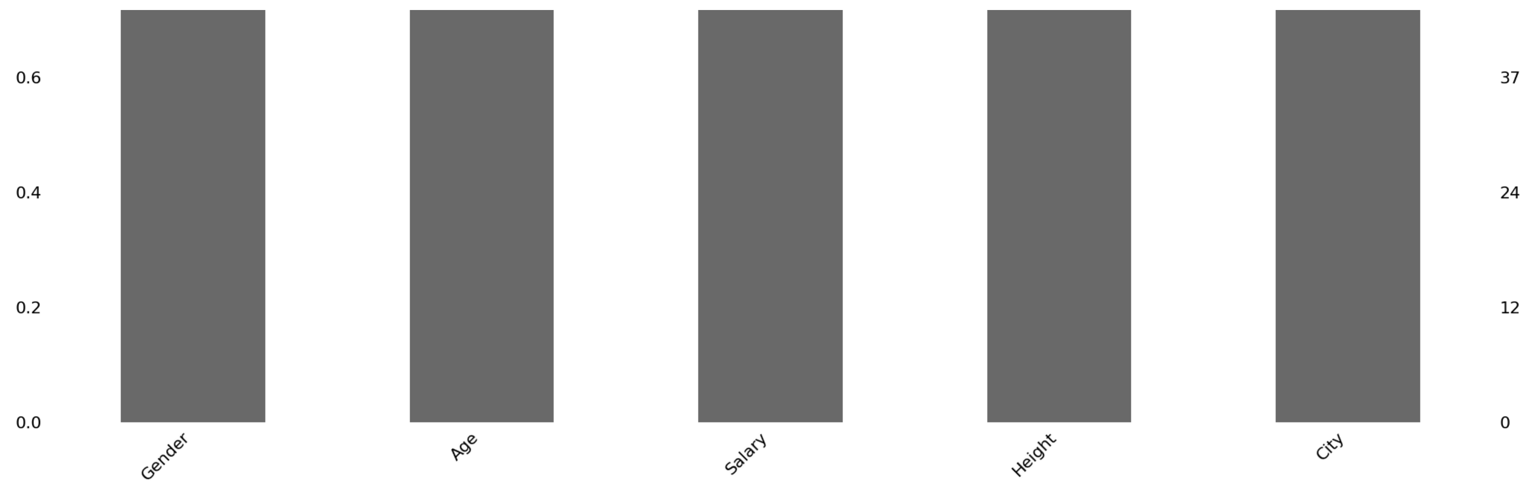
```
Female,35,69000,159,San Francisco
"""

df = pd.read_csv(StringIO(csv_data))
```

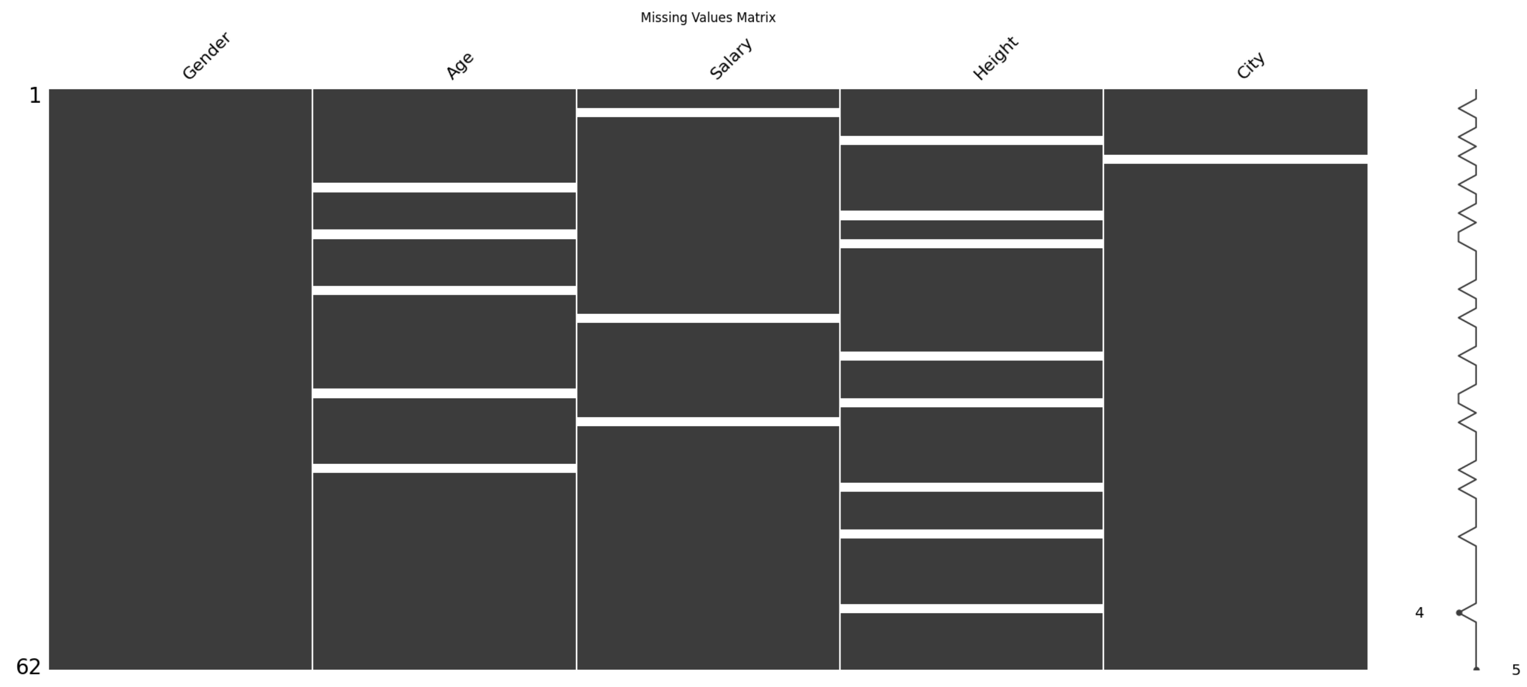
```
Requirement already satisfied: missingno in /usr/local/lib/python3.10/dist-packages (0.5.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from missingno) (3.8.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from missingno) (1.13.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (4.55.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn->missingno) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn->missingno) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn->missingno) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.17.0)
Requirement already satisfied: pyod in /usr/local/lib/python3.10/dist-packages (2.0.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from pyod) (1.4.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from pyod) (3.8.0)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.10/dist-packages (from pyod) (1.26.4)
Requirement already satisfied: numba>=0.51 in /usr/local/lib/python3.10/dist-packages (from pyod) (0.60.0)
Requirement already satisfied: scipy>=1.5.1 in /usr/local/lib/python3.10/dist-packages (from pyod) (1.13.1)
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.10/dist-packages (from pyod) (1.5.2)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51->pyod) (0.43.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.0->pyod) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (4.55.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pyod) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->pyod) (1.17.0)
```

Step 1: Basic Info and Missing Value Evaluation

Using missingno to visualize missing data

```python

print("Initial DataFrame shape:", df.shape)
print("\nColumn-wise Missing Values:\n", df.isnull().sum())

# Visualize missing data using missingno
msno.bar(df)
plt.title("Missing Values Bar Chart")
plt.show()

msno.matrix(df)
plt.title("Missing Values Matrix")
plt.show()

# Impute missing values:
num_cols = df.select_dtypes(include=[np.number]).columns
for col in num_cols:
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].mean(), inplace=True)

cat_cols = df.select_dtypes(include=['object']).columns
for col in cat_cols:
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].mode()[0], inplace=True)

print("\nDataFrame shape after missing value imputation:", df.shape)
print("Missing Values after imputation:\n", df.isnull().sum())

```





```
DataFrame shape after missing value imputation: (62, 5)
Missing Values after imputation:
 Gender    0
Age       0
Salary    0
```

Step 2: Outlier Detection Using IQR and more

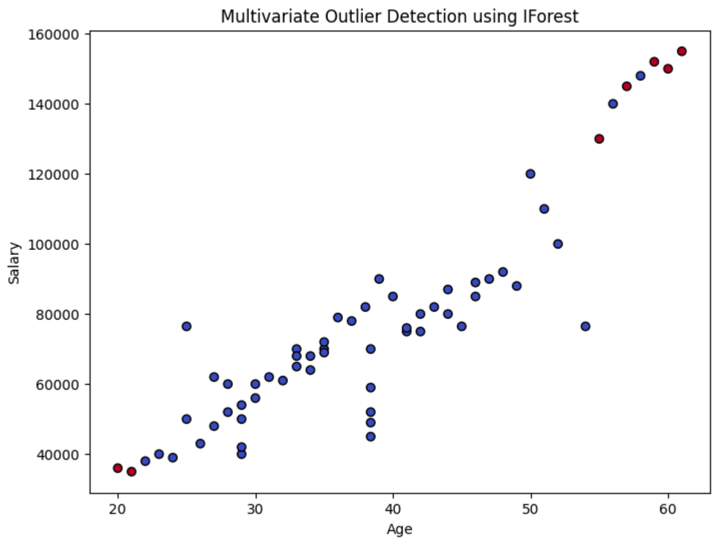
```python

Q1 = df[num_cols].quantile(0.25)
Q3 = df[num_cols].quantile(0.75)
IQR = Q3 - Q1
outlier_mask = ((df[num_cols] < (Q1 - 1.5 * IQR)) | (df[num_cols] > (Q3 + 1.5 * IQR)))
outliers_present = outlier_mask.any(axis=1)
print("\nNumber of Outliers Detected:", outliers_present.sum())

df_clean = df[~outliers_present]
print("DataFrame shape after outlier removal:", df_clean.shape)

```

```
Number of Outliers Detected: 6
DataFrame shape after outlier removal: (56, 5)
```

```python

# Using an Isolation Forest for outlier detection on numeric features
X_numeric = df[num_cols]
clf = IForest(contamination=0.1, random_state=42)
clf.fit(X_numeric)
outlier_labels = clf.labels_  # 1 for outliers, 0 for inliers

```

```python
# Count outliers
outliers_present = (outlier_labels == 1)
print("\nNumber of Outliers Detected by PYOD IForest:", outliers_present.sum())

# Visualize outliers in a 2D scatter plot (choose two numeric features)
if len(num_cols) >= 2:
    plt.figure(figsize=(8,6))
    plt.scatter(X_numeric[num_cols[0]], X_numeric[num_cols[1]], c=outlier_labels, cmap='coolwarm', edgecolor='k')
    plt.xlabel(num_cols[0])
    plt.ylabel(num_cols[1])
    plt.title("Multivariate Outlier Detection using IForest")
    plt.show()

df_clean = df[~outliers_present]
print("DataFrame shape after outlier removal:", df_clean.shape)
```

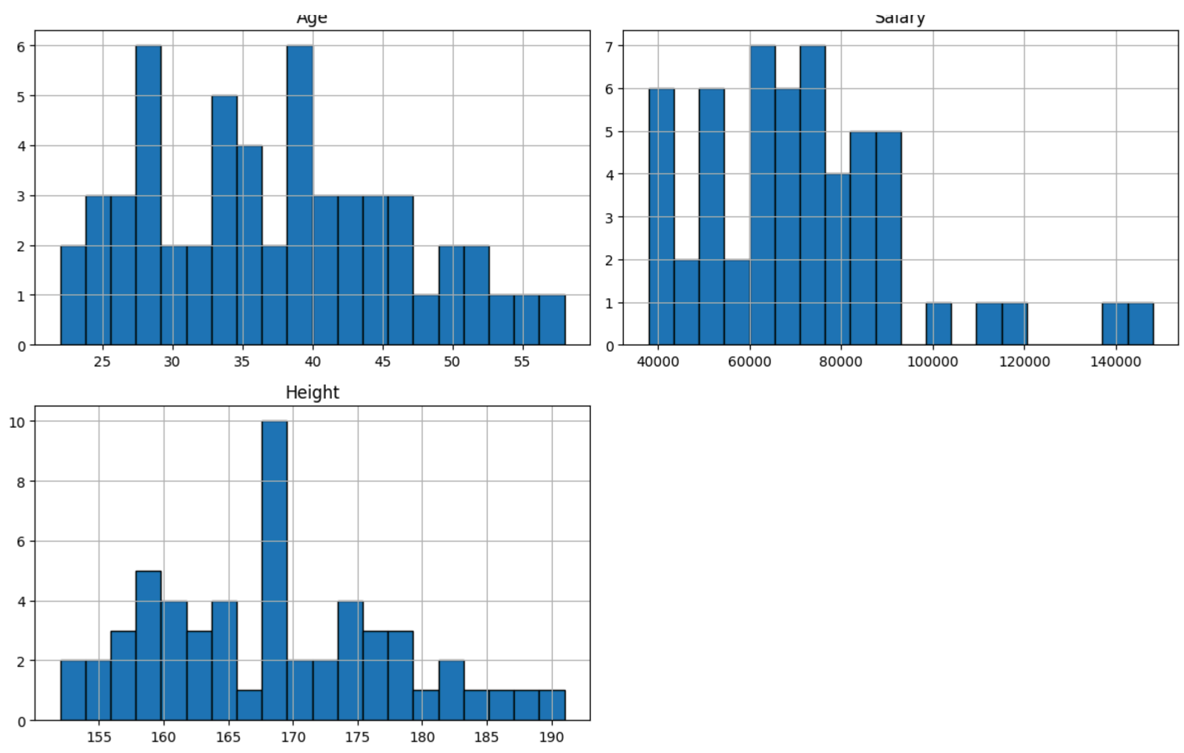
```
Number of Outliers Detected by PYOD IForest: 7
```

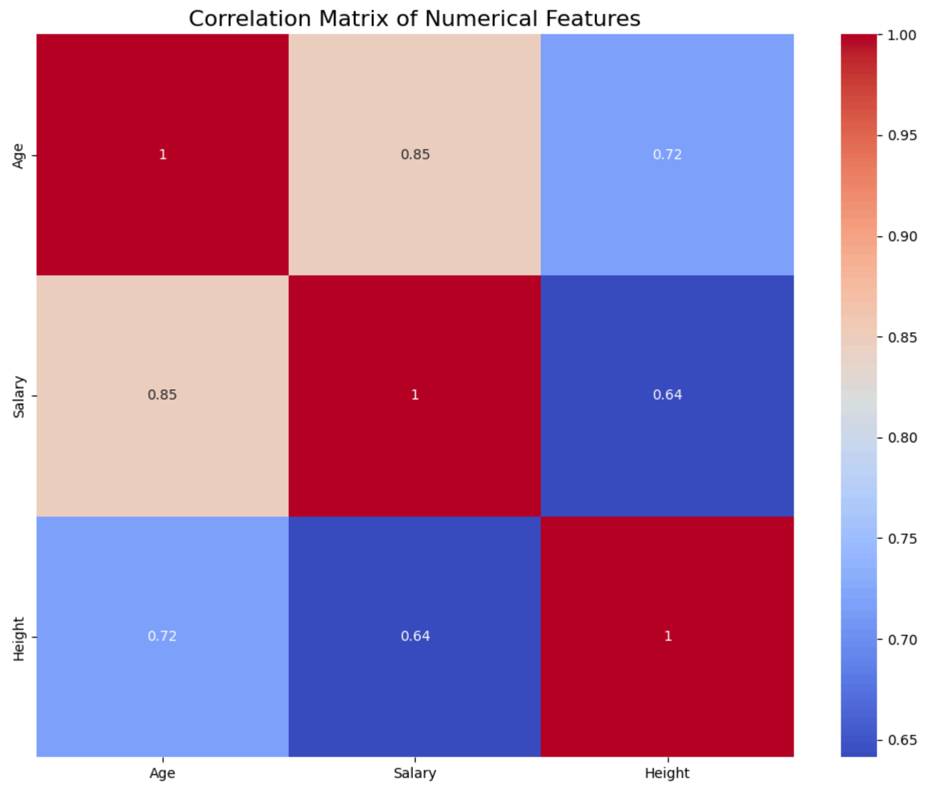


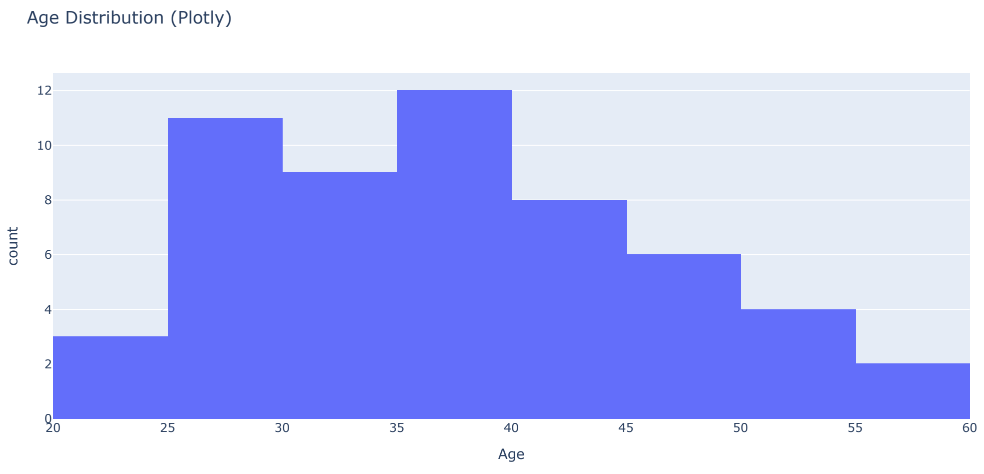
```
DataFrame shape after outlier removal: (55, 5)
```

Step 3 to 5: Visulizations

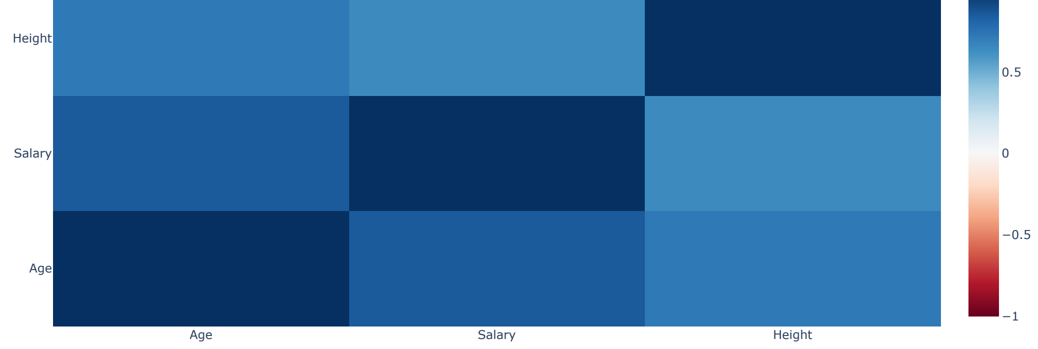
```python
# -------------------------
# Step 3: Distribution Plots using Matplotlib/Seaborn (No Change)
# -------------------------
plt.figure(figsize=(12,8))
df_clean[num_cols].hist(bins=20, edgecolor='black', figsize=(12,8))
plt.suptitle("Histograms of Numerical Features", fontsize=16)
plt.tight_layout()
plt.show()

# -------------------------
# Step 4: Correlation Analysis Using Seaborn (No Change)
# -------------------------
plt.figure(figsize=(10,8))
corr = df_clean[num_cols].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix of Numerical Features", fontsize=16)
plt.tight_layout()
plt.show()

# -------------------------
# Step 5: Interactive Visualization using Plotly (No Change)
# -------------------------
fig_age = px.histogram(df_clean, x="Age", nbins=10, title="Age Distribution (Plotly)")
fig_age.show()

fig_corr = go.Figure(data=go.Heatmap(
    z=corr.values,
    x=corr.columns,
    y=corr.columns,
    colorscale='RdBu',
    zmin=-1, zmax=1))
fig_corr.update_layout(title="Correlation Matrix (Plotly)", xaxis_nticks=36)
fig_corr.show()
```


Age
Salary
Height


Correlation Matrix of Numerical Features
Age
Salary
Height
1
0.85
0.72
0.85
1
0.64
0.72
0.64
1


Age Distribution (Plotly)
count
Age

Correlation Matrix (Plotly)



Step 6: Prepare Data for Supervised Machine Learning

```

# For demonstration, let's predict Salary from the other features.
# We'll consider "Salary" as the target variable and the rest as features.
X = df_clean.drop("Salary", axis=1)
y = df_clean["Salary"]

numeric_features = X.select_dtypes(include=np.number).columns.tolist()
categorical_features = X.select_dtypes(include=['object']).columns.tolist()

preprocessor = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), numeric_features),
        ("cat", OneHotEncoder(drop='first'), categorical_features)
    ]
)

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

```

Step 7: Extend Modeling - Compare Multiple Models using RMSE and QQ plots

Models: LinearRegression, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor

```


models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(random_state=42, n_estimators=500),
    "Gradient Boosting": GradientBoostingRegressor(random_state=42, n_estimators=500)
}


results = {}
inference_times = {}

for model_name, model_obj in models.items():
    pipe = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('regressor', model_obj)
    ])
    pipe.fit(X_train, y_train)

    # Inference time measurement
    start_time = time.time()
    y_pred = pipe.predict(X_test)
    end_time = time.time()
    inference_time = end_time - start_time
    inference_times[model_name] = inference_time

    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    results[model_name] = (pipe, rmse, y_pred)

# Print RMSE results
print("\nModel Performance (RMSE):")
for model_name, (pipe, rmse, y_pred) in results.items():
    print(f"{model_name}: RMSE = {rmse:.2f}")

```

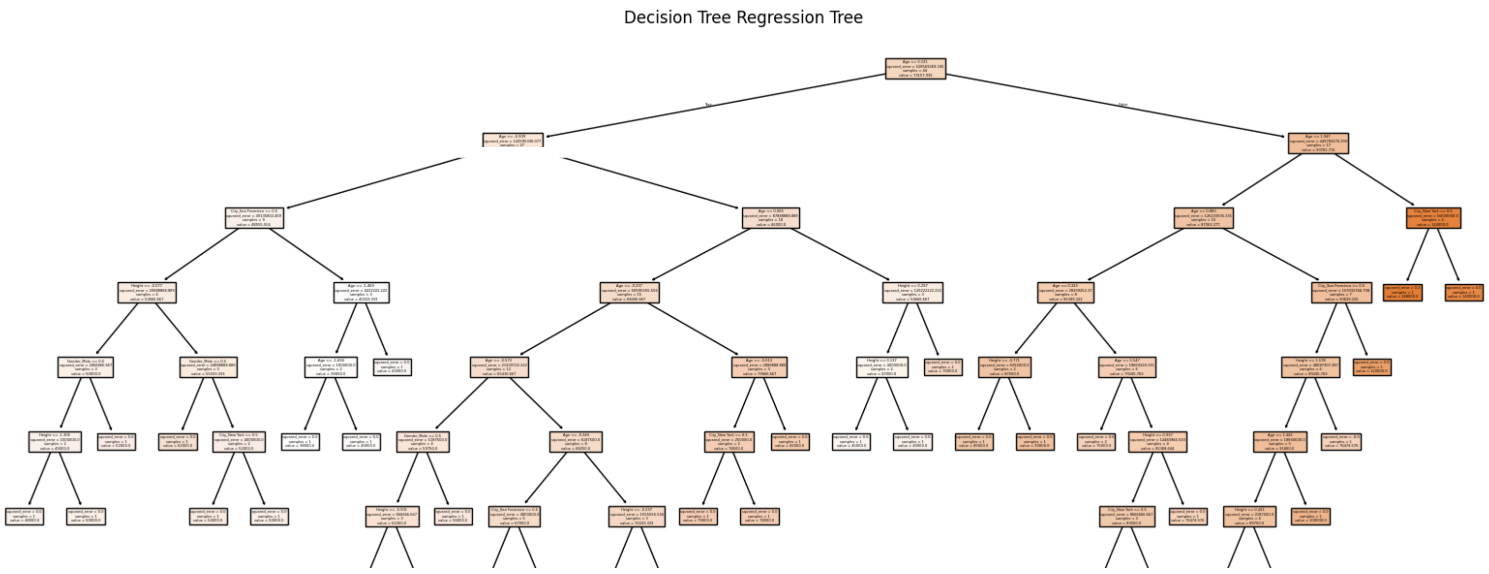
```
Model Performance (RMSE):
Linear Regression: RMSE = 16652.39
Decision Tree: RMSE = 13013.29
Random Forest: RMSE = 11602.97
Gradient Boosting: RMSE = 13201.14
```

```

# Select best model based on RMSE
best_model_name = min(results, key=lambda x: results[x][1])
best_pipe, best_rmse, best_pred = results[best_model_name]
print(f"\nBest Model: {best_model_name} with RMSE = {best_rmse:.2f}")

```

```
Best Model: Random Forest with RMSE = 11602.97
```

```

# 7a: Tree representation of the regression tree
# Note: Only works for the Decision Tree model
tree_model = results["Decision Tree"][0].named_steps['regressor']
plt.figure(figsize=(20,10))
plot_tree(tree_model, feature_names=numeric_features+list(results["Decision Tree"][0].named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(ca
plt.title("Decision Tree Regression Tree")
```

```python
plt.show()

```



```python

# 7b: QQ plots using plotly
def qqplot_plotly(residuals, model_name):
    # Calculate theoretical quantiles and sample quantiles
    qq = probplot(residuals, dist="norm")
    theoretical_q = qq[0][0]
    sample_q = qq[0][1]
    slope, intercept = qq[1][0], qq[1][1]

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=theoretical_q, y=sample_q, mode='markers', name='Data'))
    # line of best fit
    line_y = slope*theoretical_q + intercept
    fig.add_trace(go.Scatter(x=theoretical_q, y=line_y, mode='lines', name='Best Fit Line'))
    fig.update_layout(title=f"QQ Plot of Residuals - {model_name}",
                      xaxis_title='Theoretical Quantiles',
                      yaxis_title='Sample Quantiles')
    fig.show()

for model_name, (pipe, rmse, y_pred) in results.items():
    residuals = y_test - y_pred
    qqplot_plotly(residuals, model_name)

```