```
# Import necessary libraries
from pyflink.table import EnvironmentSettings, TableEnvironment
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Initialize the Flink batch environment
env_settings = EnvironmentSettings.new_instance().in_batch_mode().build()
table_env = TableEnvironment.create(env_settings)

# Test by creating a simple Flink table
test_data = [(1, 'Hello'), (2, 'Flink'), (3, 'Test')]
test_table = table_env.from_elements(test_data, ['id', 'text'])
# Print the table schema and data
test_table.execute().print()

# Specify the S3 file path
s3_file_path = ' s3://cloud-project-bucket/movies_metadata.csv '

# Load the dataset directly from S3
movies_data = pd.read_csv(s3_file_path, storage_options={'anon': False},
low_memory=False)
```

```
# Keep only relevant columns
movies_data_cleaned = movies_data[['id', 'budget', 'vote_average']]

# Drop rows with null values in 'budget' or 'vote_average'
movies_data_cleaned = movies_data_cleaned.dropna(subset=['budget', 'vote_average'])

# Convert 'budget' column to numeric, and remove invalid or zero budgets
movies_data_cleaned['budget'] = pd.to_numeric(movies_data_cleaned['budget'],
errors='coerce')
movies_data_cleaned = movies_data_cleaned[movies_data_cleaned['budget'] > 0]

# Drop duplicate rows
movies_data_cleaned = movies_data_cleaned.drop_duplicates()

# Independent variable (X) - budget
X = movies_data_cleaned[['budget']]
# Dependent variable (y) - vote_average
y = movies_data_cleaned['vote_average']

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

```
# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R2 Score: {r2}")

# Plotting the results
# Scatter plot with regression line (manual plotting)
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test, y_pred, color='red', label='Regression Line')
plt.xlabel('Budget')
plt.ylabel('Vote Average')
plt.title('Budget vs Vote Average with Regression Line')
plt.legend()
plt.show()

# Scatter plot with regression line (using Seaborn)
plt.figure(figsize=(10, 6))
sns.regplot(x='budget', y='vote_average', data=movies_data_cleaned,
scatter_kws={'color': 'blue'}, line_kws={'color': 'red'})
plt.title('Budget vs Vote Average with Regression Line')
plt.xlabel('Budget')
```

```
plt.ylabel('Vote Average')

plt.show()
```