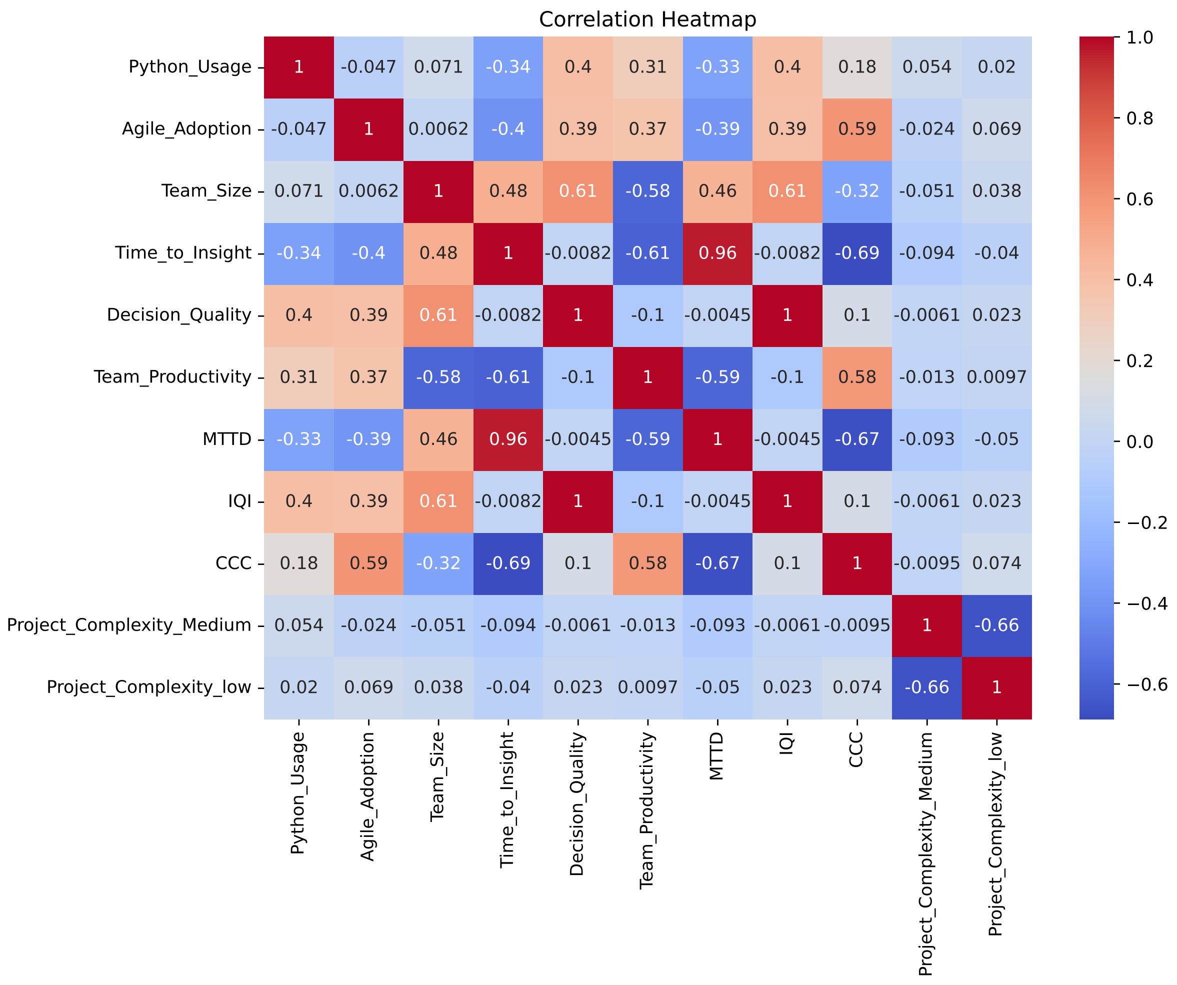
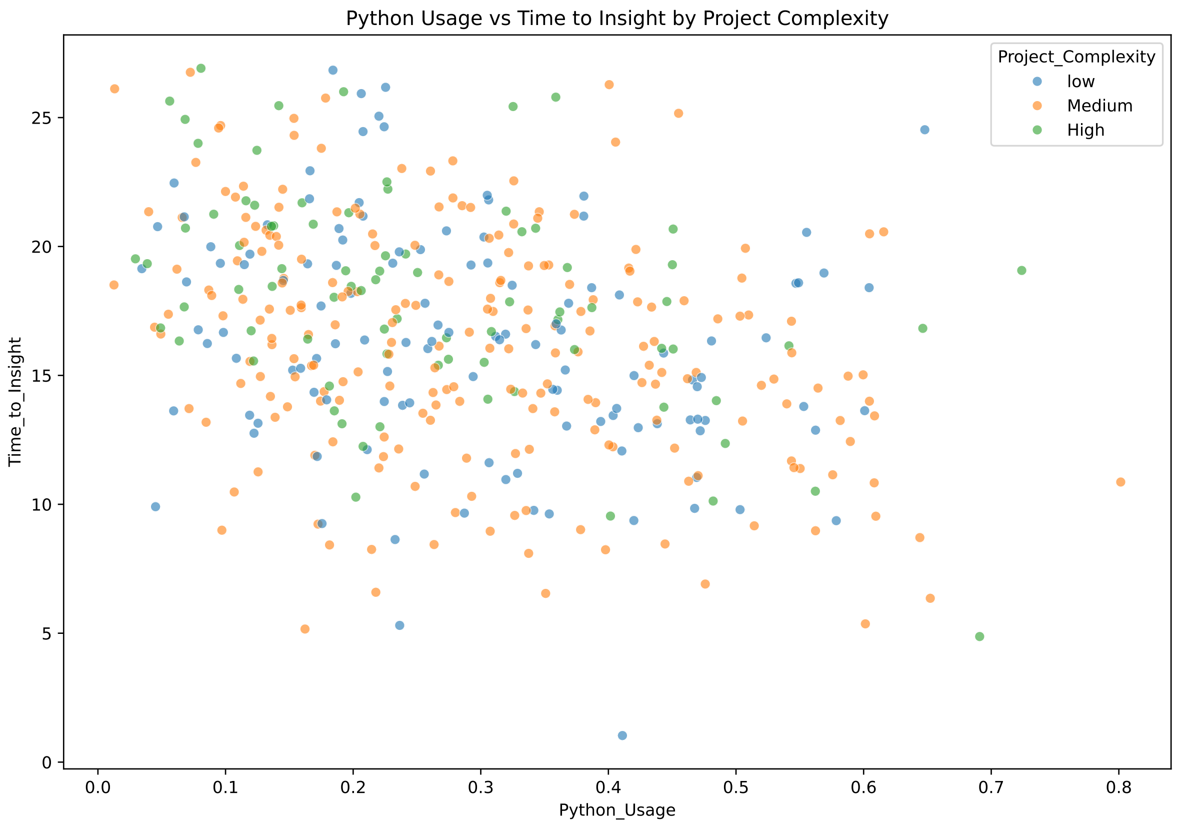
Python Project Complexity

```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Simulating a more complex dataset
np.random.seed(42)
n_samples = 500

# Independent variables
python_usage = np.random.beta(2, 5, n_samples)  # Right-skewed distribution
agile_adoption = np.random.beta(3, 3, n_samples)  # Bell-shaped distribution
team_size = np.random.randint(3, 20, n_samples)
project_complexity = np.random.choice(['low', 'Medium', 'High'], n_samples, p=[0.3, 0.5, 0.2

# Dependent variables with non-linear relationships and noise
time_to_insight = 20 - 15 * np.log(python_usage + 1) - 10 * agile_adoption + 0.5 * team_size
decision_quality = 2 + 3 * python_usage**2 + 2 * agile_adoption + 0.1 * team_size + np.rando
team_productivity = 5 + 4 * np.sqrt(python_usage) + 3 * agile_adoption - 0.2 * team_size + np

# Create DataFrame
df = pd.DataFrame({
    'Python_Usage': python_usage,
    'Agile_Adoption': agile_adoption,
    'Team_Size': team_size,
    'Project_Complexity': project_complexity,
    'Time_to_Insight': time_to_insight,
    'Decision_Quality': decision_quality,
    'Team_Productivity': team_productivity
})

# Add some missing data
df.loc[np.random.choice(df.index, 50), 'Time_to_Insight'] = np.nan
df.loc[np.random.choice(df.index, 30), 'Decision_Quality'] = np.nan

# Handling missing data (drop rows with missing values)
df = df.dropna()

# Calculate metrics - use the reduced DataFrame's length
n_samples_reduced = len(df)

# Now calculate the metrics using the reduced dataset
df['MTTD'] = df['Time_to_Insight'] * np.random.uniform(0.7, 0.9, n_samples_reduced)  # More i
df['IQI'] = (df['Decision_Quality'] - df['Decision_Quality'].min()) / (df['Decision_Quality']
df['CCC'] = (df['Team_Productivity'] * df['Agile_Adoption']) / (df['Time_to_Insight'] + 1)  #

# Encoding the categorical variable 'Project_Complexity'
df_encoded = pd.get_dummies(df, columns=['Project_Complexity'], drop_first=True)

# Correlation analysis
corr_matrix = df_encoded.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()

# Multiple Linear Regression
X = df_encoded[['Python_Usage', 'Agile_Adoption', 'Team_Size'] + [col for col in df_encoded.c
y = df_encoded['Team_Productivity']

# Scale the independent variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=

# Fit the model
model = LinearRegression()
```

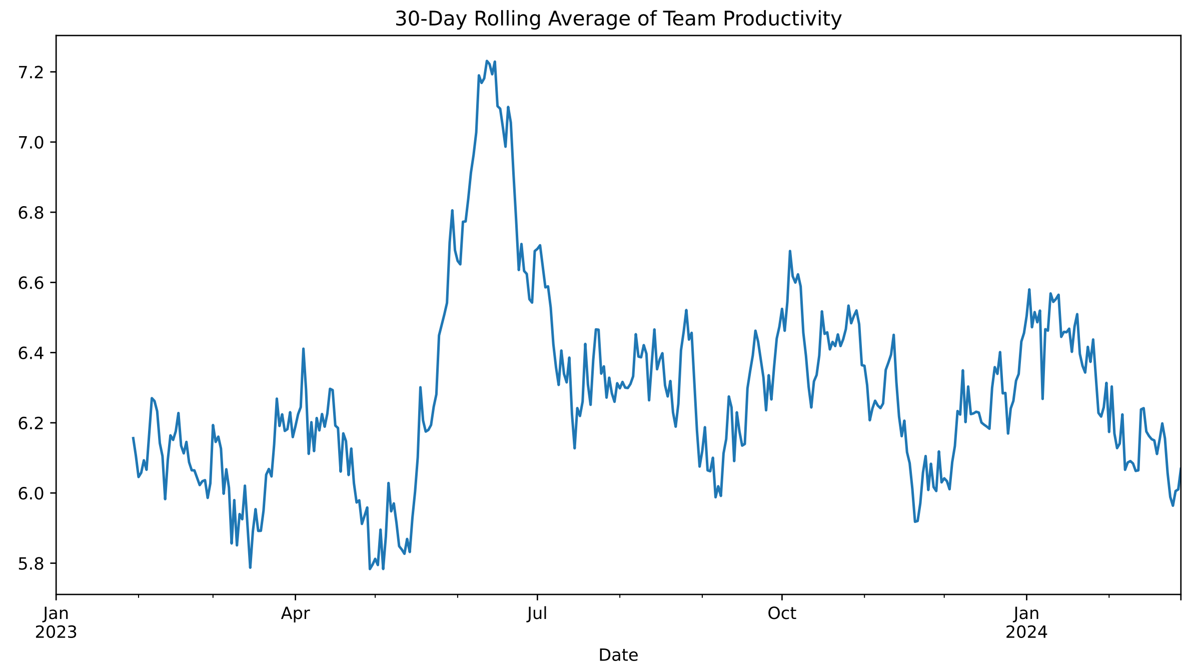
```python
model.fit(X_train, y_train)

# Print model results
print("##Multiple Linear Regression Results:")
print(f"R-squared: {model.score(X_test, y_test):.4f}")
for feature, coef in zip(X.columns, model.coef_):
    print(f"{feature}: {coef:.4f}")

# Visualize relationships with Project Complexity using the original 'Project_Complexity' col
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df, x='Python_Usage', y='Time_to_Insight', hue='Project_Complexity', alp
plt.title('Python Usage vs Time to Insight by Project Complexity')
plt.show()

# ANOVA for Project Complexity impact on Team Productivity
complexity_groups = [group for _, group in df.groupby('Project_Complexity')['Team_Productivi
f_statistic, p_value = stats.f_oneway(*complexity_groups)
print("##ANOVA Results (Project Complexity impact on Team Productivity):")
print(f"F-statistic: {f_statistic:.4f}, p-value: {p_value:.4f}")

# Metrics summary by Project Complexity
print("##Metrics Summary by Project Complexity:")
print(df.groupby('Project_Complexity')[['MTTD', 'IQI', 'CCC']].agg(['mean', 'std']))

# Time series simulation for longitudinal analysis
dates = pd.date_range(start='2023-01-01', periods=n_samples_reduced)
df['Date'] = dates
df = df.set_index('Date')

plt.figure(figsize=(12, 6))
df['Team_Productivity'].rolling(window=30).mean().plot()
plt.title('30-Day Rolling Average of Team Productivity')
plt.show()
```



```
##Multiple Linear Regression Results:
R-squared: 0.6064
Python_Usage: 0.6386
Agile_Adoption: 0.6324
Team_Size: -0.9753
Project_Complexity_Medium: -0.1463
Project_Complexity_low: -0.0872
```



```
##ANOVA Results (Project Complexity impact on Team Productivity):
F-statistic: 0.0352, p-value: 0.9654
##Metrics Summary by Project Complexity:
                         MTTD                   IQI                 CCC  \
                         mean        std       mean        std       mean
Project_Complexity
High                14.702475   3.476199   0.481064   0.180644   0.190752
Medium              13.056061   3.700421   0.487145   0.182614   0.222255
low                 13.100069   3.522965   0.494810   0.181547   0.250005


                         std
Project_Complexity
High                0.153755
Medium              0.177466
low                 0.312079
```



30-Day Rolling Average of Team Productivity
7.2
7.0
6.8
6.6
6.4
6.2
6.0
5.8
Jan
2023
Apr
Jul
Oct
Jan
2024
Date