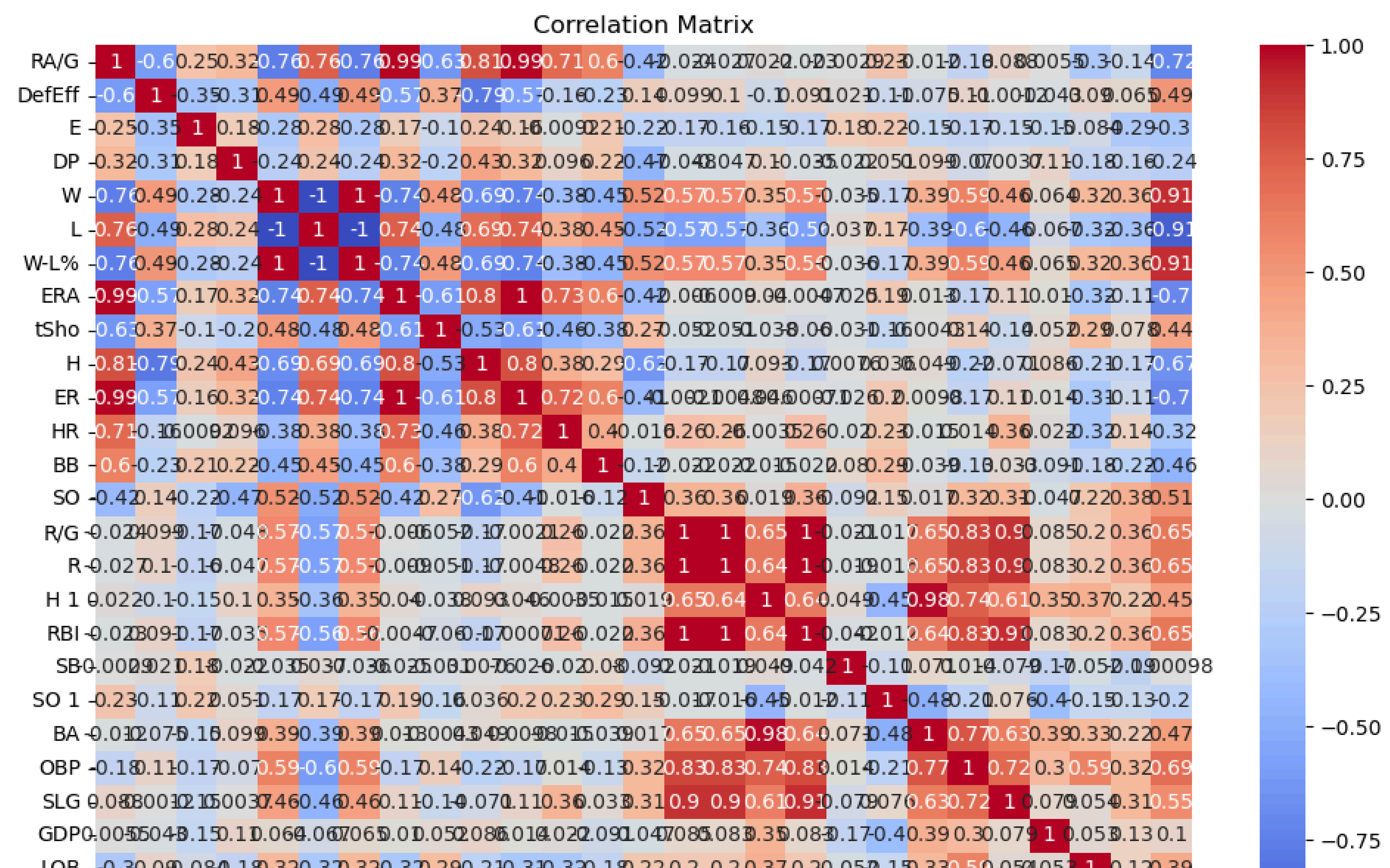
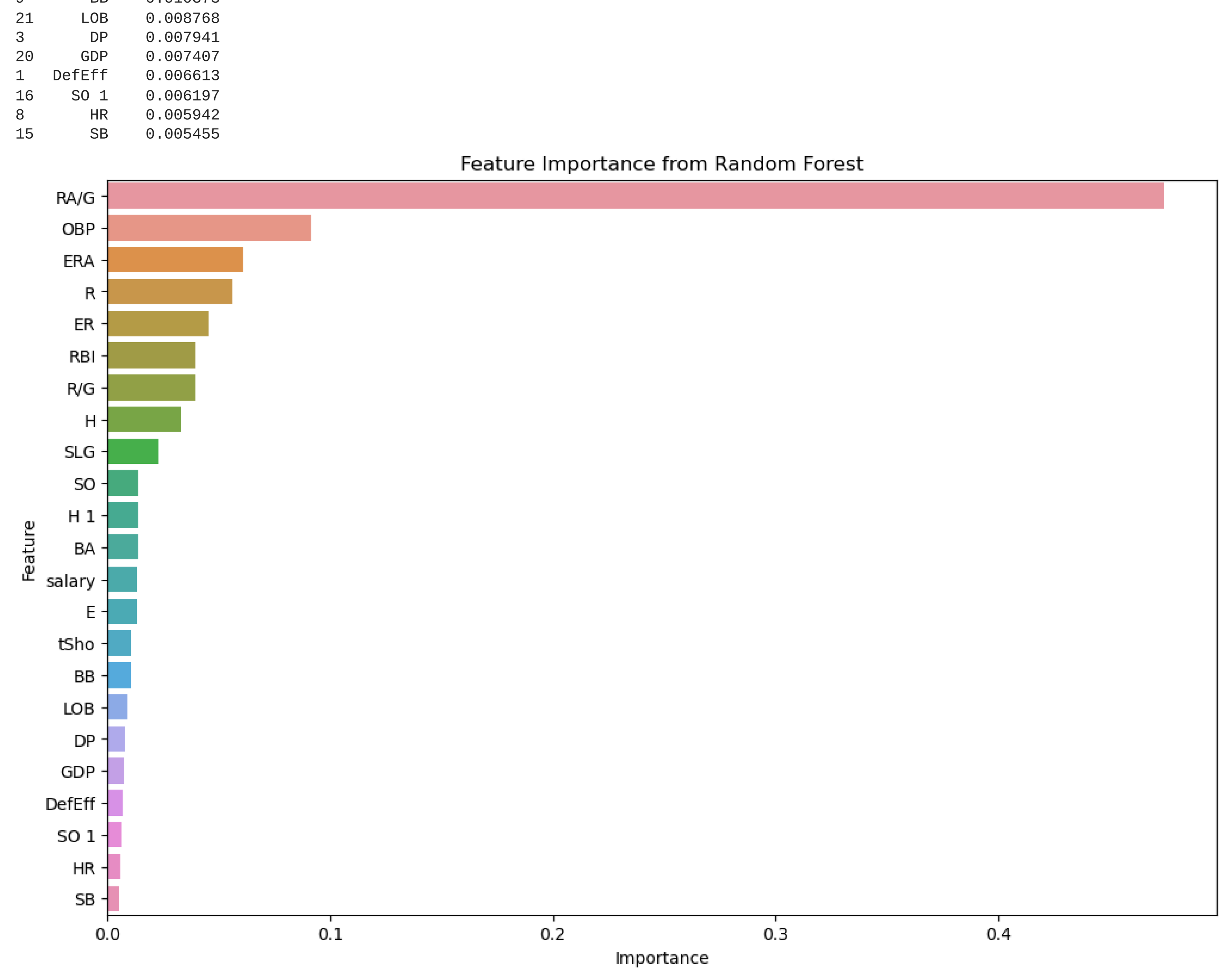
In [3]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.feature_selection import SelectKBest, f_regression

# Load the dataset
df = pd.read_csv('mlb_teams.csv')

# Display the first few rows of the dataset and column names
print(df.head())
print(df.columns)

# Check for missing values
print(df.isnull().sum())

# Inspect the column names to identify non-numeric columns
print(df.columns)

# Ensure column names are correctly identified
columns_to_drop = ['TeamName']
for col in df.columns:
    if 'season' in col.lower():
        columns_to_drop.append(col)

print(f"Columns to drop: {columns_to_drop}")

# Drop non-numeric columns if they exist (adjust as necessary based on actual column names)
df = df.drop(columns=columns_to_drop)

# If there are missing values, handle them (e.g., fill with mean, median, or drop)
df = df.fillna(df.mean())

# Verify all columns are numeric now
print(df.dtypes)

# Define the features (X) and the target variable (y)
X = df.drop(columns=['W-L%', 'W', 'L', 'WAR'])  # Adjust based on your dataset's column names
y = df['W-L%']

# 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)

# Normalize/scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Feature selection using correlation
correlation_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Feature selection using SelectKBest
selector = SelectKBest(score_func=f_regression, k='all')
selector.fit(X_train_scaled, y_train)
feature_scores = pd.DataFrame({'Feature': X.columns, 'Score': selector.scores_})
feature_scores = feature_scores.sort_values(by='Score', ascending=False)
print(feature_scores)

# Fit a linear regression model
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
y_pred_lr = lr.predict(X_test_scaled)

# Evaluate the linear regression model
print('Linear Regression RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print('Linear Regression R^2:', r2_score(y_test, y_pred_lr))

# Fit a random forest regressor model
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)

# Evaluate the random forest model
print('Random Forest RMSE:', np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print('Random Forest R^2:', r2_score(y_test, y_pred_rf))

# Feature importance from random forest
feature_importances = pd.DataFrame({'Feature': X.columns, 'Importance': rf.feature_importances_})
feature_importances = feature_importances.sort_values(by='Importance', ascending=False)
print(feature_importances)

# Plot feature importance
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importances)
plt.title('Feature Importance from Random Forest')
plt.show()
```

```
   TeamName  RA/G  DefEff    E   DP    W    L   W-L%   ERA  tSho  ...  RBI  \
0  2018 ARI  3.98   0.698   75  152   82   80  0.506  3.72     9  ...  658
1  2018 ATL  4.06   0.709   80  134   90   72  0.556  3.75    11  ...  717
2  2018 BAL  5.51   0.674  104  159   47  115  0.290  5.18     7  ...  593
3  2018 BOS  3.99   0.693   77  106  108   54  0.667  3.75    14  ...  829
4  2018 CHC  3.96   0.700  104  155   95   68  0.583  3.65    18  ...  722

    SB  SO 1     BA    OBP    SLG  GDP   LOB     salary   WAR
0   79  1460  0.235  0.310  0.397  110  1086  143324597  34.1
1   90  1290  0.257  0.324  0.417   99  1143  130649395  40.8
2   81  1412  0.239  0.298  0.391  132  1027  127633703  11.4
3  125  1253  0.268  0.339  0.453  130  1124  227398860  56.5
4   66  1388  0.258  0.333  0.410  107  1224  194259933  45.0

[5 rows x 28 columns]
Index(['TeamName', 'RA/G', 'DefEff', 'E', 'DP', 'W', 'L', 'W-L%', 'ERA',
       'tSho', 'H', 'ER', 'HR', 'BB', 'SO', 'R/G', 'R', 'H 1', 'RBI', 'SB',
       'SO 1', 'BA', 'OBP', 'SLG', 'GDP', 'LOB', 'salary', 'WAR'],
      dtype='object')
TeamName    0
RA/G        0
DefEff      0
E           0
DP          0
W           0
L           0
W-L%        0
ERA         0
tSho        0
H           0
ER          0
HR          0
BB          0
SO          0
R/G         0
R           0
H 1         0
RBI         0
SB          0
SO 1        0
BA          0
OBP         0
SLG         0
GDP         0
LOB         0
salary      0
WAR         0
dtype: int64
Index(['TeamName', 'RA/G', 'DefEff', 'E', 'DP', 'W', 'L', 'W-L%', 'ERA',
       'tSho', 'H', 'ER', 'HR', 'BB', 'SO', 'R/G', 'R', 'H 1', 'RBI', 'SB',
       'SO 1', 'BA', 'OBP', 'SLG', 'GDP', 'LOB', 'salary', 'WAR'],
      dtype='object')
Columns to drop: ['TeamName']
RA/G      float64
DefEff    float64
E           int64
DP          int64
W           int64
L           int64
W-L%      float64
ERA       float64
tSho        int64
H           int64
ER          int64
HR          int64
BB          int64
SO          int64
R/G       float64
R           int64
H 1         int64
RBI         int64
SB          int64
SO 1        int64
BA        float64
OBP       float64
SLG       float64
GDP         int64
LOB         int64
salary      int64
WAR       float64
dtype: object
```




```
   Feature       Score
0     RA/G  249.567124
4      ERA  224.427562
7       ER  223.365367
6        H  163.631605
18     OBP  100.728243
12       R   69.618082
11     R/G   68.775512
14     RBI   68.428632
5     tSho   60.264219
1   DefEff   57.965822
10      SO   57.626167
9       BB   56.162005
19     SLG   34.205305
8       HR   33.114826
17      BA   32.904535
13     H 1   25.792274
21     LOB   23.716298
22  salary   23.508915
2        E   14.901698
16    SO 1    9.574437
3       DP    6.078356
20     GDP    1.207182
15      SB    0.022458
Linear Regression RMSE: 0.02352260342713923
Linear Regression R^2: 0.8697738040454528
Random Forest RMSE: 0.026306159110422445
Random Forest R^2: 0.8371294863007793
   Feature  Importance
0     RA/G    0.474464
18     OBP    0.091246
4      ERA    0.060793
12       R    0.056258
7       ER    0.045598
14     RBI    0.039433
11     R/G    0.039411
6        H    0.033119
19     SLG    0.022841
10      SO    0.014020
13     H 1    0.013777
17      BA    0.013630
22  salary    0.013116
2        E    0.013094
5     tSho    0.010506
9       BB    0.010373
21     LOB    0.008768
3       DP    0.007941
20     GDP    0.007407
1   DefEff    0.006613
16    SO 1    0.006197
8       HR    0.005942
15      SB    0.005455
```



In [ ]: