In [ ]:
```
#Avinya D
#240970006
#MCA-B
#Batch-B1
```

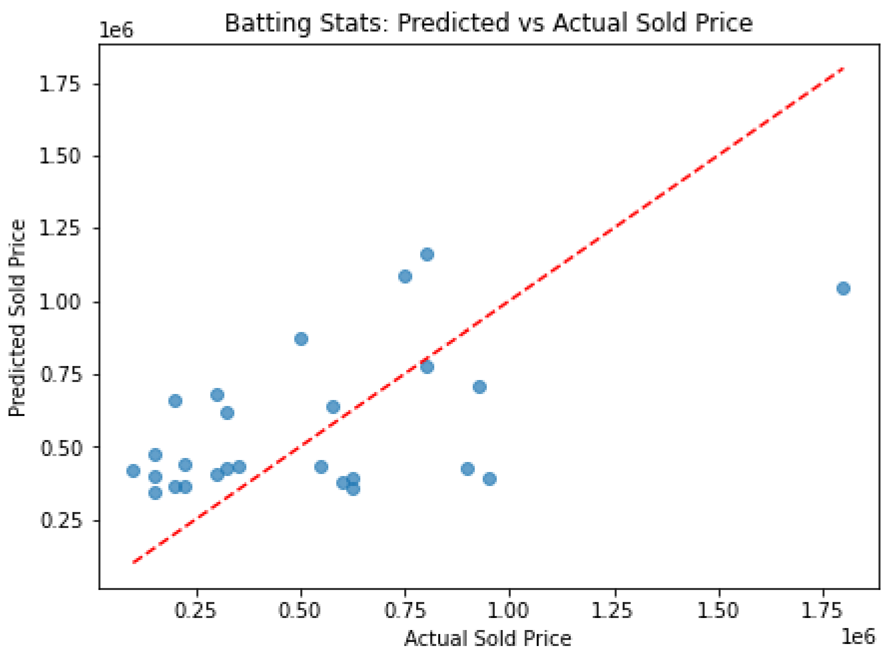
In [ ]:
```
#Exercise 1
#Using the given IPL 2013 dataset, apply Linear Regression techniques to predi
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
df=pd.read_csv("IPL DATA2013.csv")
```

In [ ]:
```
#1. Price Prediction from Batting Stats
#• Build a linear regression model to predict SOLD PRICE based on batting stat
#RUNS-S, HS (High Score), SR-B (Strike Rate), SIXERS.
#• Evaluate model accuracy using R² score and RMSE.
#• Visualize the predicted vs actual prices with a scatter plot.
```

In [5]:
```
X = df[['RUNS-S', 'HS', 'SR-B', 'SIXERS']]
y = df['SOLD PRICE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Batting Model R2: {r2:.3f}")
print(f"Batting Model RMSE: {rmse:.2f}")

# Plot Predicted vs Actual
plt.figure(figsize=(7,5))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.xlabel("Actual Sold Price")
plt.ylabel("Predicted Sold Price")
plt.title("Batting Stats: Predicted vs Actual Sold Price")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.show()
```

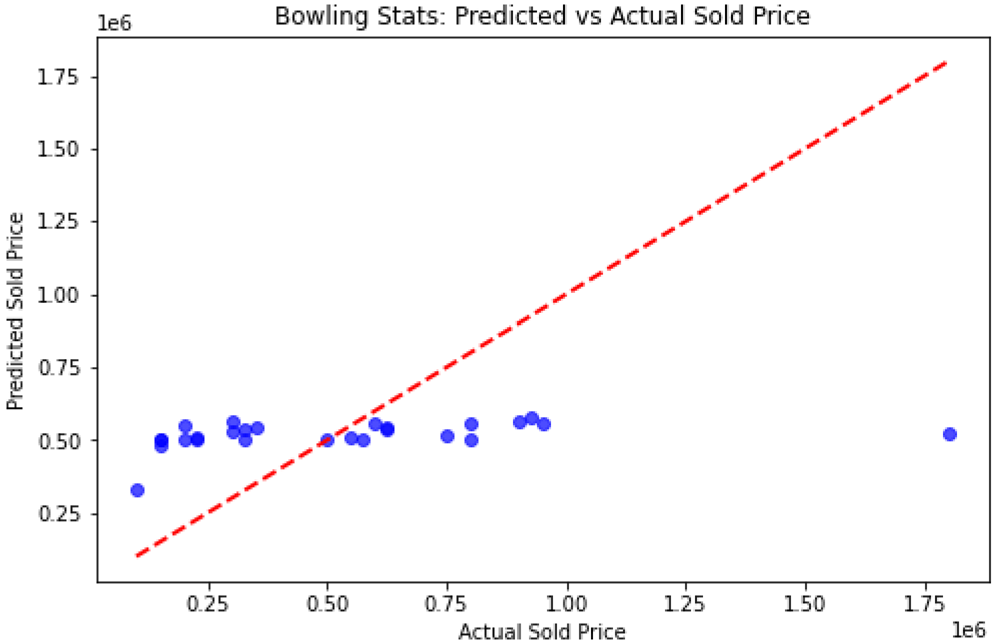
```
Batting Model R2: 0.266
Batting Model RMSE: 316135.83
```

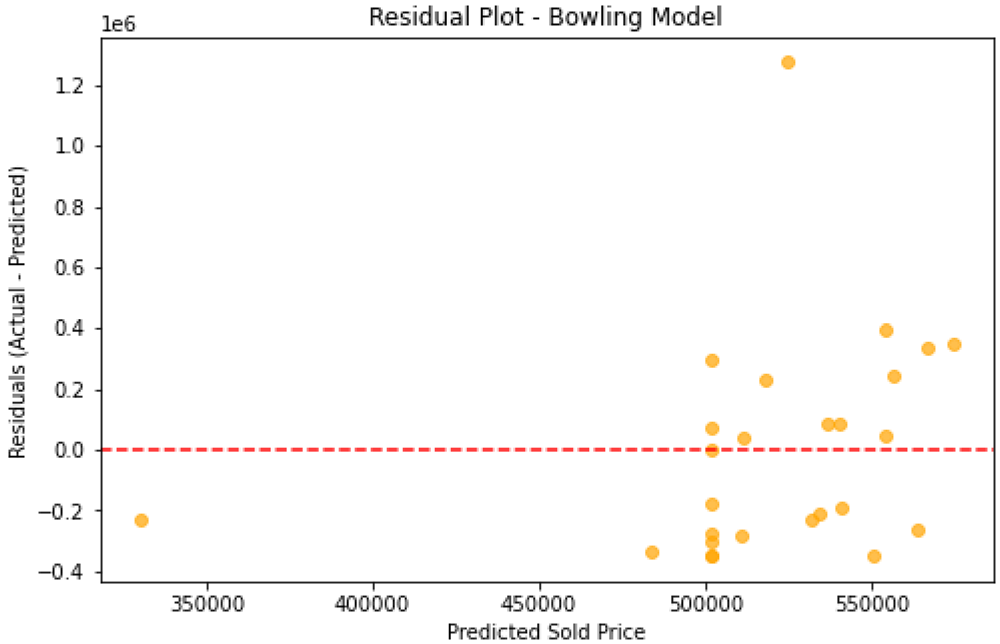


In [6]:

```python
#2. Price Prediction from Bowling Stats
#• Train a linear regression model to predict SOLD PRICE using bowling feature
#AVE-BL, ECON, SR-BL.
#• Compare predicted vs actual prices and check which bowling feature contribu
#coefficients).
#• Plot residuals to check model errors.

X = df[['WKTS', 'AVE-BL', 'ECON', 'SR-BL']]
y = df['SOLD PRICE']

# Split data into train and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

lr_bowling = LinearRegression()
lr_bowling.fit(X_train, y_train)

y_pred = lr_bowling.predict(X_test)

# Evaluate model
r2 = r2_score(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f"Bowling Model R²: {r2:.3f}")
print(f"Bowling Model RMSE: {rmse:.2f}")

coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr_bowling.coef_
}).sort_values(by='Coefficient', key=abs, ascending=False)

print("\nFeature importance (coefficients):")
print(coef_df)

plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, alpha=0.7, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw
plt.xlabel('Actual Sold Price')
plt.ylabel('Predicted Sold Price')
plt.title('Bowling Stats: Predicted vs Actual Sold Price')
plt.show()

residuals = y_test - y_pred
plt.figure(figsize=(8,5))
plt.scatter(y_pred, residuals, alpha=0.7, color='orange')
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Sold Price')
plt.ylabel('Residuals (Actual - Predicted)')
plt.title('Residual Plot - Bowling Model')
plt.show()
```

```
Bowling Model R²: 0.079
Bowling Model RMSE: 354193.58

Feature importance (coefficients):
  Feature  Coefficient
2    ECON -8185.005700
1  AVE-BL  4172.406266
3   SR-BL -2505.340876
0    WKTS  1050.434506
```

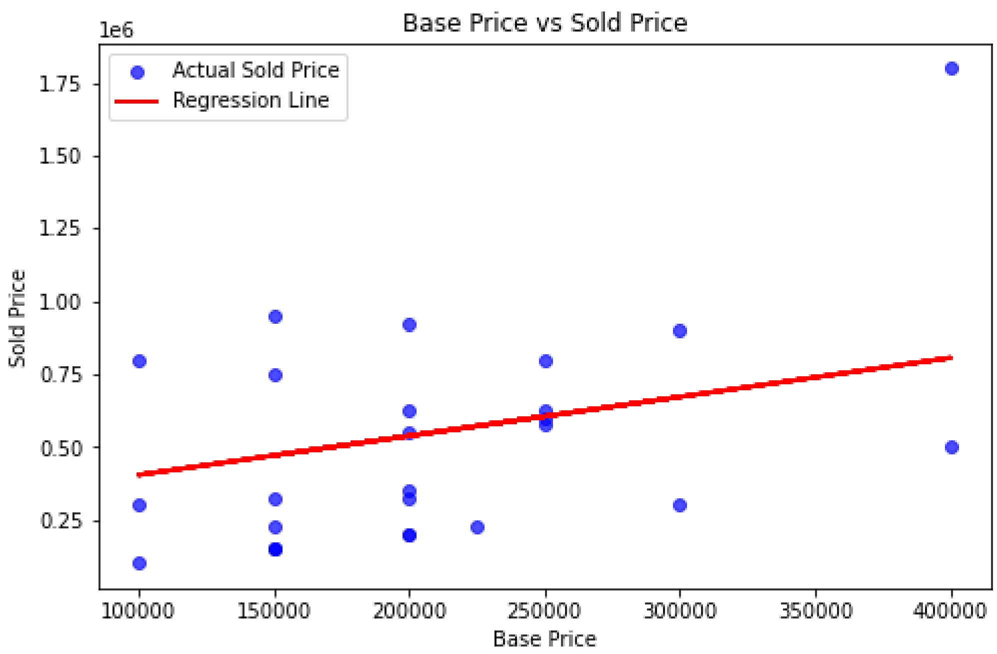

Bowling Stats: Predicted vs Actual Sold Price
1e6
Predicted Sold Price
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Actual Sold Price
1e6


Residual Plot - Bowling Model
1e6
Residuals (Actual - Predicted)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
350000
400000
450000
500000
550000
Predicted Sold Price

In [7]:

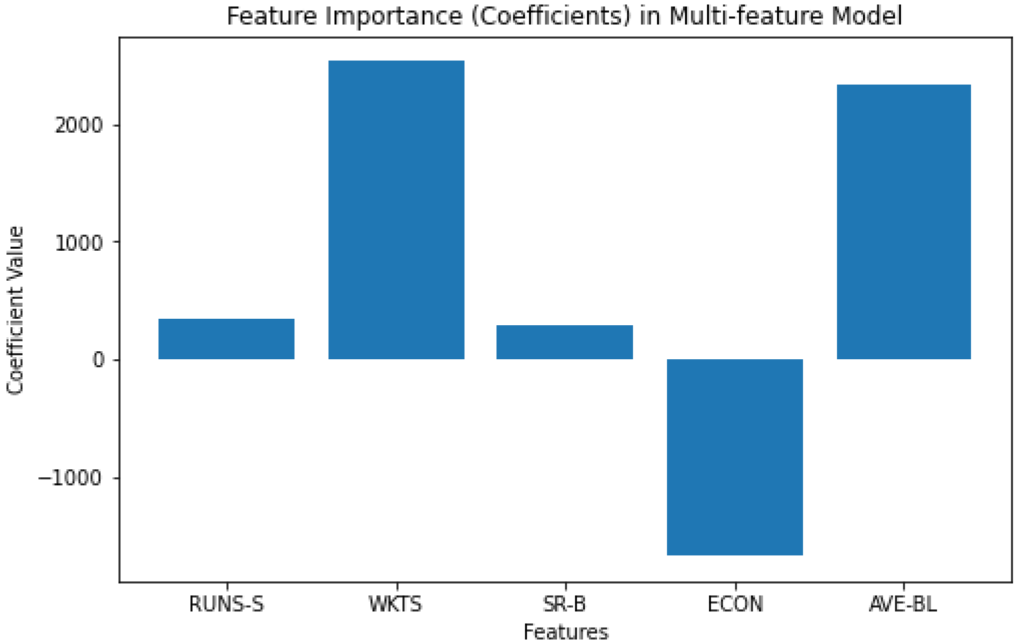
```python
#3. Base Price vs Sold Price Relationship
#• Fit a regression model with BASE PRICE as input and SOLD PRICE as output.
#• Check accuracy using Mean Absolute Error (MAE).
#• Visualize using a regression line plot.

X = df[['BASE PRICE']]
y = df['SOLD PRICE']

# Split dataset (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

lr_base = LinearRegression()
lr_base.fit(X_train, y_train)

y_pred = lr_base.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")

plt.figure(figsize=(8,5))
plt.scatter(X_test, y_test, color='blue', alpha=0.7, label='Actual Sold Price'
plt.plot(X_test.values.flatten(), y_pred, color='red', linewidth=2, label='Reg
plt.xlabel('Base Price')
plt.ylabel('Sold Price')
plt.title('Base Price vs Sold Price')
plt.legend()
plt.show()
```

```
Mean Absolute Error (MAE): 268302.61
```



In [8]:
```python
#4. Multi-feature Model for Auction Price
#• Build a multiple linear regression model using both batting and bowling fea
#RUNS-S, WKTS, SR-B, ECON, AVE-BL, etc.) to predict SOLD PRICE.
#• Compare performance against single-feature models.
#• Show feature importance (coefficients bar chart).

features = ['RUNS-S', 'WKTS', 'SR-B', 'ECON', 'AVE-BL']
X = df[features]
y = df['SOLD PRICE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

multi_lr = LinearRegression()
multi_lr.fit(X_train, y_train)

y_pred_multi = multi_lr.predict(X_test)

r2_multi = r2_score(y_test, y_pred_multi)
rmse_multi = np.sqrt(mean_squared_error(y_test, y_pred_multi))

print(f"Multi-feature Model R²: {r2_multi:.4f}")
print(f"Multi-feature Model RMSE: {rmse_multi:.2f}")

# Single feature model (RUNS-S)
X_train_single = X_train[['RUNS-S']]
X_test_single = X_test[['RUNS-S']]

single_lr = LinearRegression()
single_lr.fit(X_train_single, y_train)

y_pred_single = single_lr.predict(X_test_single)

r2_single = r2_score(y_test, y_pred_single)
rmse_single = np.sqrt(mean_squared_error(y_test, y_pred_single))

print(f"Single feature (RUNS-S) Model R²: {r2_single:.4f}")
print(f"Single feature (RUNS-S) Model RMSE: {rmse_single:.2f}")

coefficients = multi_lr.coef_
plt.figure(figsize=(8, 5))
plt.bar(features, coefficients)
plt.xlabel('Features')
plt.ylabel('Coefficient Value')
plt.title('Feature Importance (Coefficients) in Multi-feature Model')
plt.show()
```

```
Multi-feature Model R²: 0.3517
Multi-feature Model RMSE: 297153.61
Single feature (RUNS-S) Model R²: 0.2615
Single feature (RUNS-S) Model RMSE: 317157.11
```

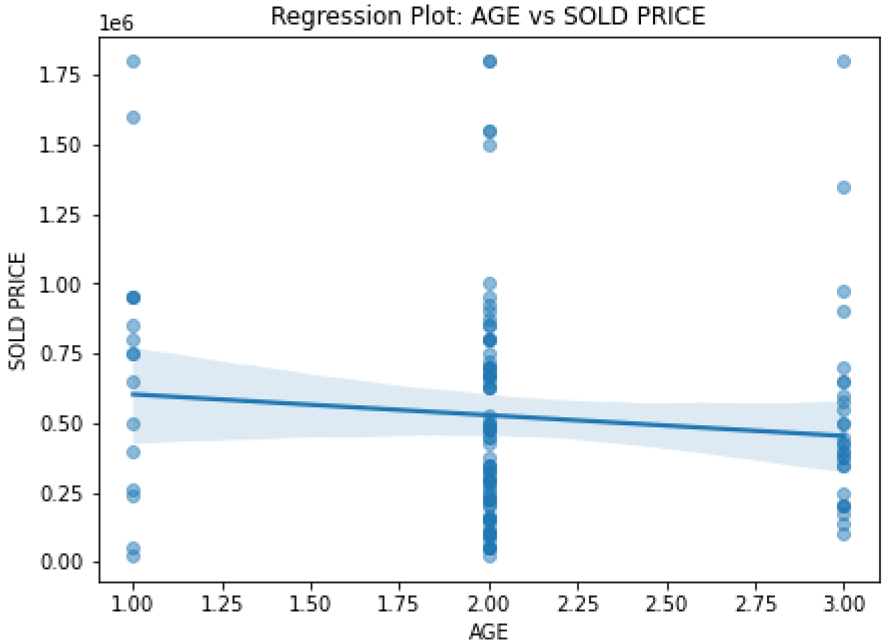


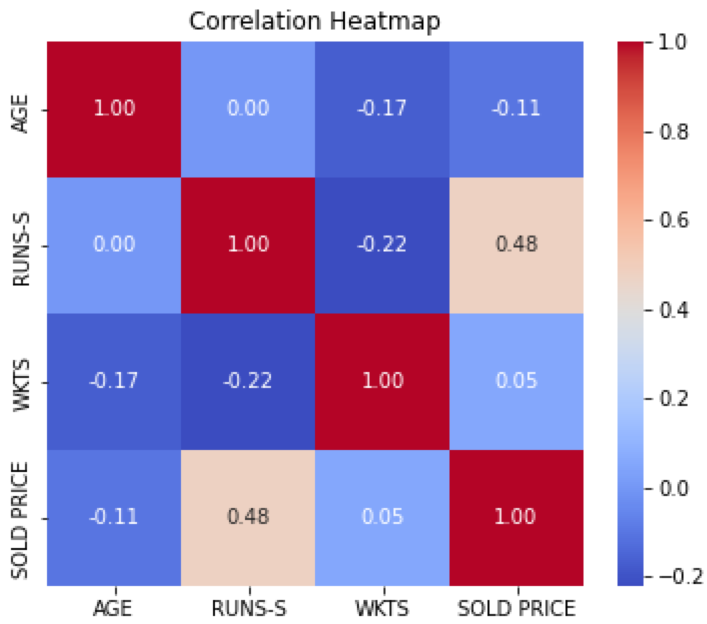
Feature Importance (Coefficients) in Multi-feature Model
Coefficient Value
2000
1000
0
−1000
RUNS-S
WKTS
SR-B
ECON
AVE-BL
Features

In [9]:

```
#5. Age Impact on Price
#• Train a regression model to see if AGE and performance stats (RUNS, WKTS) e
#variations in SOLD PRICE.
#• Evaluate correlation between AGE and price using regression plots.
#• Visualize with heatmap of correlations.

features = ['AGE', 'RUNS-S', 'WKTS']
X = df[features]
y = df['SOLD PRICE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

age_model = LinearRegression()
age_model.fit(X_train, y_train)

y_pred = age_model.predict(X_test)

print(f"R² score: {r2_score(y_test, y_pred):.4f}")

plt.figure(figsize=(7,5))
sns.regplot(x='AGE', y='SOLD PRICE', data=df, scatter_kws={'alpha':0.5})
plt.title('Regression Plot: AGE vs SOLD PRICE')
plt.show()

corr_features = ['AGE', 'RUNS-S', 'WKTS', 'SOLD PRICE']
corr_matrix = df[corr_features].corr()

plt.figure(figsize=(6,5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
R² score: 0.3076
```

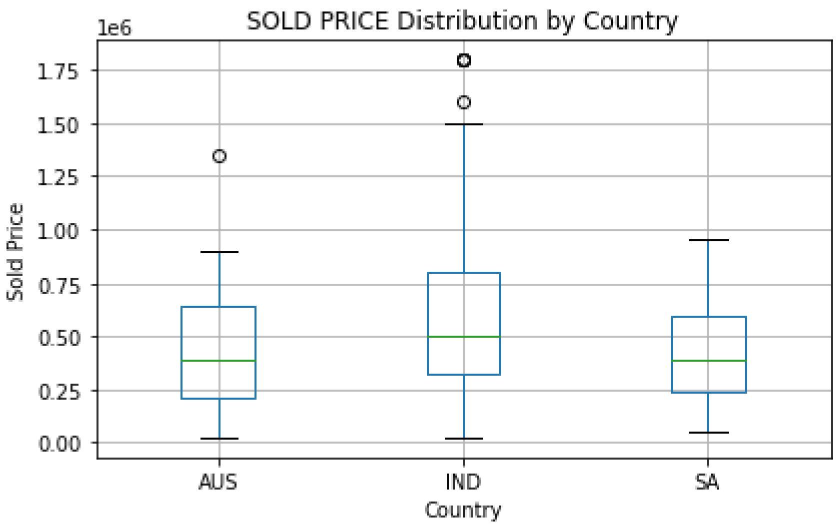



Correlation Heatmap
AGE
RUNS-S
WKTS
SOLD PRICE
1.00
0.00
-0.17
-0.11
0.00
1.00
-0.22
0.48
-0.17
-0.22
1.00
0.05
-0.11
0.48
0.05
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

In [10]:

```python
#6. Country-wise Price Prediction
#• Build regression models for different COUNTRY groups (e.g., IND vs AUS vs S
#• Compare model accuracies for each country.
#• Use boxplots to visualize SOLD PRICE distribution by country.

# Drop rows with missing values
df = df.dropna(subset=["COUNTRY", "SOLD PRICE", "RUNS-S", "WKTS"])

# Choose countries
selected_countries = ["IND", "AUS", "SA"]

# Boxplot of SOLD PRICE by COUNTRY
plt.figure(figsize=(8, 5))
df_selected = df[df["COUNTRY"].isin(selected_countries)]
df_selected.boxplot(column="SOLD PRICE", by="COUNTRY")
plt.title("SOLD PRICE Distribution by Country")
plt.suptitle("")
plt.xlabel("Country")
plt.ylabel("Sold Price")
plt.grid(True)
plt.tight_layout()
plt.show()

# Dictionary to store R² scores
r2_scores = {}

# Run regression for each country
for country in selected_countries:
    country_df = df[df["COUNTRY"] == country]
    X = country_df[["RUNS-S", "WKTS"]]
    y = country_df["SOLD PRICE"]

    # Train/test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r

    # Fit model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict and score
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    r2_scores[country] = round(r2, 3)

# Show model performance
print("R² Scores by Country:")
for country, score in r2_scores.items():
    print(f"{country}: {score}")
```

```
<Figure size 576x360 with 0 Axes>
```

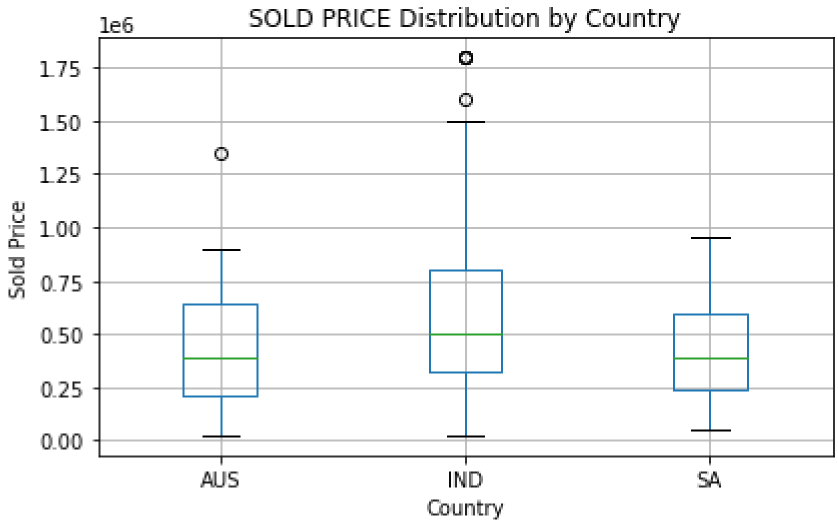



```
R² Scores by Country:
IND: 0.148
AUS: 0.632
SA: 0.001
```

In [11]:

```python
#6. Country-wise Price Prediction
#• Build regression models for different COUNTRY groups (e.g., IND vs AUS vs S
#• Compare model accuracies for each country.
#• Use boxplots to visualize SOLD PRICE distribution by country.

# Filter valid entries
df = df.dropna(subset=["COUNTRY", "SOLD PRICE", "RUNS-S", "WKTS"])

# Select specific countries
countries = ['IND', 'AUS', 'SA']
r2_scores = {}

# Boxplot: SOLD PRICE by COUNTRY
plt.figure(figsize=(8, 5))
df[df['COUNTRY'].isin(countries)].boxplot(column='SOLD PRICE', by='COUNTRY')
plt.title('SOLD PRICE Distribution by Country')
plt.suptitle('')
plt.xlabel('Country')
plt.ylabel('Sold Price')
plt.grid(True)
plt.tight_layout()
plt.show()

# Regression for each country
for country in countries:
    subset = df[df['COUNTRY'] == country]
    X = subset[['RUNS-S', 'WKTS']]
    y = subset['SOLD PRICE']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    r2 = r2_score(y_test, y_pred)
    r2_scores[country] = round(r2, 3)

# Print R² scores
print("Country-wise R² scores for SOLD PRICE prediction:")
for country, score in r2_scores.items():
    print(f"{country}: {score}")
```

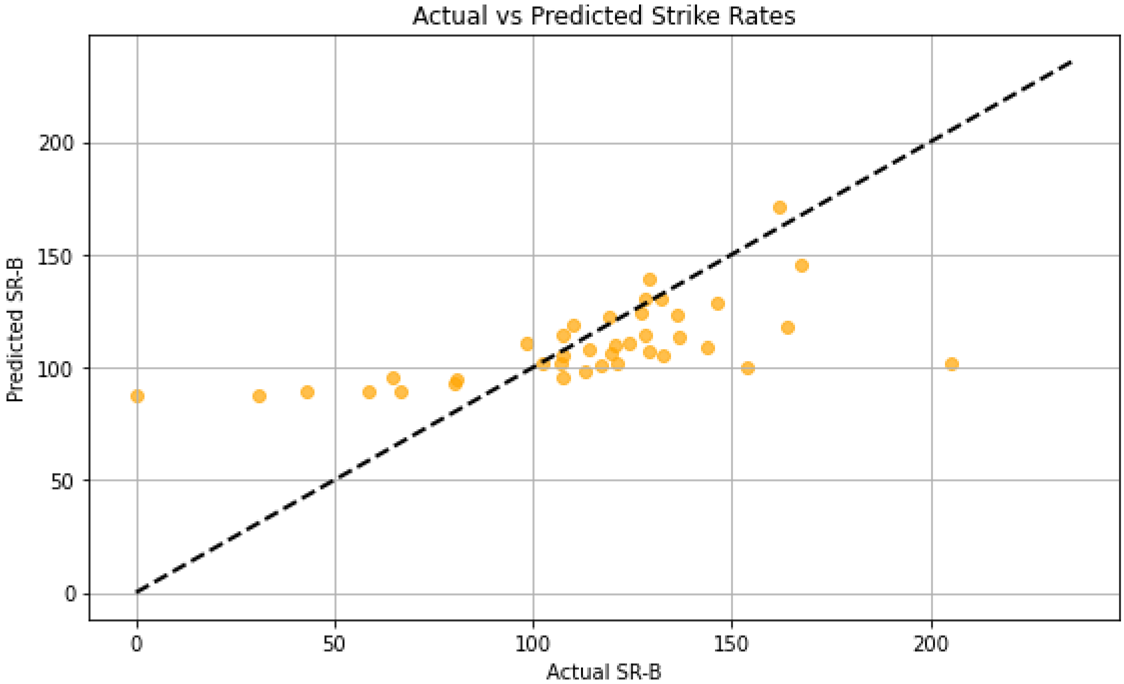
```
<Figure size 576x360 with 0 Axes>
```



```
Country-wise R² scores for SOLD PRICE prediction:
IND: 0.148
AUS: 0.632
SA: 0.001
```

In [12]:
```python
#7. Prediction of Strike Rate (Batting)
#• Use player stats (RUNS-S, HS, SIXERS, AGE) to predict SR-B (Strike Rate) wi
#regression.
#• Evaluate prediction accuracy.
#• Plot actual vs predicted strike rates.

from sklearn.metrics import mean_squared_error

# Filter rows with necessary batting stats
batting_df = df.dropna(subset=["RUNS-S", "HS", "SIXERS", "AGE", "SR-B"])

# Define features and target
X = batting_df[["RUNS-S", "HS", "SIXERS", "AGE"]]
y = batting_df["SR-B"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

# Train model
sr_model = LinearRegression()
sr_model.fit(X_train, y_train)
y_pred = sr_model.predict(X_test)

# Evaluate
r2 = r2_score(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print(f"\nStrike Rate Prediction — R²: {r2:.3f}, RMSE: {rmse:.2f}")

# Plot actual vs predicted SR-B
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.7, color='orange')
plt.xlabel("Actual SR-B")
plt.ylabel("Predicted SR-B")
plt.title("Actual vs Predicted Strike Rates")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Strike Rate Prediction — R²: 0.352, RMSE: 31.06
```



Actual vs Predicted Strike Rates
Predicted SR-B
Actual SR-B
0
50
100
150
200