In [64]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from math import floor
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.regressionplots import influence_plot
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import SGDRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score,accuracy_score
%matplotlib inline

# matplotlib defaults
plt.style.use("seaborn-whitegrid")
plt.rc("figure", autolayout=True)
plt.rc(
    "axes",
    labelweight="bold",
    labelsize="large",
    titleweight="bold",
    titlesize=14,
    titlepad=10,
)

import warnings
warnings.filterwarnings('ignore')
```

Descriptive and Exploratory data analysis

In [65]:

```
df = pd.read_csv('ipl.csv', index_col=0)
df.head()
```

Out[65]:

SI.NO.	PLAYER NAME	AGE	COUNTRY	TEAM	PLAYING ROLE	T-RUNS	T-WKTS	ODI-RUNS-S	ODI-SR-B	ODI-WKTS	
1	Abdulla, YA	2	SA	KXIP	Allrounder	0	0	0	0.00	0	.
2	Abdur Razzak	2	BAN	RCB	Bowler	214	18	657	71.41	185	.
3	Agarkar, AB	2	IND	KKR	Bowler	571	58	1269	80.62	288	.
4	Ashwin, R	1	IND	CSK	Bowler	284	31	241	84.56	51	.
5	Badrinath, S	2	IND	CSK	Batsman	63	0	79	45.93	0	.

5 rows × 25 columns

In [66]:

```
print("shape of the dataset: ", df.shape)
print("""--------------------------------------
----------------------------------------""")
print(df.info())
```

```
shape of the dataset:  (130, 25)
--------------------------------------
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 130 entries, 1 to 130
Data columns (total 25 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   PLAYER NAME    130 non-null    object
 1   AGE            130 non-null    int64
 2   COUNTRY        130 non-null    object
 3   TEAM           130 non-null    object
 4   PLAYING ROLE   130 non-null    object
 5   T-RUNS         130 non-null    int64
 6   T-WKTS         130 non-null    int64
 7   ODI-RUNS-S     130 non-null    int64
 8   ODI-SR-B       130 non-null    float64
 9   ODI-WKTS       130 non-null    int64
 10  ODI-SR-BL      130 non-null    float64
 11  CAPTAINCY EXP  130 non-null    int64
 12  RUNS-S         130 non-null    int64
 13  HS             130 non-null    int64
 14  AVE            130 non-null    float64
 15  SR-B           130 non-null    float64
 16  SIXERS         130 non-null    int64
 17  RUNS-C         130 non-null    int64
 18  WKTS           130 non-null    int64
 19  AVE-BL         130 non-null    float64
 20  ECON           130 non-null    float64
 21  SR-BL          130 non-null    float64
 22  AUCTION YEAR   130 non-null    int64
 23  BASE PRICE     130 non-null    int64
 24  SOLD PRICE     130 non-null    int64
dtypes: float64(7), int64(14), object(4)
memory usage: 26.4+ KB
None
```

In [67]:

```
# Dividing categorical and numerical columns
print("Columns names are as below: \n",df.columns)
print("-------------------------------------------------------------------------------")
cat_col = [col for col in df.columns if df[col].dtype == 'object']
num_col = [col for col in df.columns if df[col].dtype in ['float64','int64']]
print("categorical columns: ", cat_col)
print("-------------------------------------------------------------------------------")
print("numerical columns: ", num_col)
```

```
Columns names are as below:
 Index(['PLAYER NAME', 'AGE', 'COUNTRY', 'TEAM', 'PLAYING ROLE', 'T-RUNS',
       'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B', 'ODI-WKTS', 'ODI-SR-BL',
       'CAPTAINCY EXP', 'RUNS-S', 'HS', 'AVE', 'SR-B', 'SIXERS', 'RUNS-C',
       'WKTS', 'AVE-BL', 'ECON', 'SR-BL', 'AUCTION YEAR', 'BASE PRICE',
       'SOLD PRICE'],
      dtype='object')
-------------------------------------------------------------------------------
categorical columns:  ['PLAYER NAME', 'COUNTRY', 'TEAM', 'PLAYING ROLE']
-------------------------------------------------------------------------------
numerical columns:  ['AGE', 'T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B',
'ODI-WKTS', 'ODI-SR-BL', 'CAPTAINCY EXP', 'RUNS-S', 'HS', 'AVE', 'SR-B',
'SIXERS', 'RUNS-C', 'WKTS', 'AVE-BL', 'ECON', 'SR-BL', 'AUCTION YEAR', 'BA
SE PRICE', 'SOLD PRICE']
```

In [68]:

```
df.describe().T.sort_values(by='std', ascending=False)
```

Out[68]:

	count	mean	std	min	25%	50%	
SOLD PRICE	130.0	521223.076923	406807.351419	20000.0	225000.0000	437500.000	700000
BASE PRICE	130.0	192230.769231	153097.300897	20000.0	100000.0000	200000.000	225000
ODI-RUNS-S	130.0	2508.738462	3582.205625	0.0	73.2500	835.000	3523
T-RUNS	130.0	2166.715385	3305.646757	0.0	25.5000	542.500	3002
RUNS-S	130.0	514.246154	615.226335	0.0	39.0000	172.000	925
RUNS-C	130.0	475.523077	558.314049	0.0	0.0000	297.000	689
T-WKTS	130.0	66.530769	142.676855	0.0	0.0000	7.000	47
ODI-WKTS	130.0	76.076923	111.205070	0.0	0.0000	18.500	106
HS	130.0	47.430769	36.403624	0.0	16.0000	35.500	73
SR-B	130.0	111.053462	35.928907	0.0	98.2375	118.510	129
ODI-SR-BL	130.0	34.033846	26.751749	0.0	0.0000	36.600	45
ODI-SR-B	130.0	71.164385	25.898440	0.0	65.6500	78.225	86
SIXERS	130.0	17.692308	23.828146	0.0	1.0000	6.000	29
WKTS	130.0	17.169231	21.816763	0.0	0.0000	8.500	23
AVE-BL	130.0	23.110231	20.802057	0.0	0.0000	24.785	35
SR-BL	130.0	17.382615	15.273422	0.0	0.0000	19.935	26
AVE	130.0	18.719308	11.094224	0.0	9.8250	18.635	27
ECON	130.0	6.204462	4.941531	0.0	0.0000	7.380	8
AUCTION YEAR	130.0	2009.092308	1.377821	2008.0	2008.0000	2008.000	2011
AGE	130.0	2.092308	0.576627	1.0	2.0000	2.000	2
CAPTAINCY EXP	130.0	0.315385	0.466466	0.0	0.0000	0.000	1

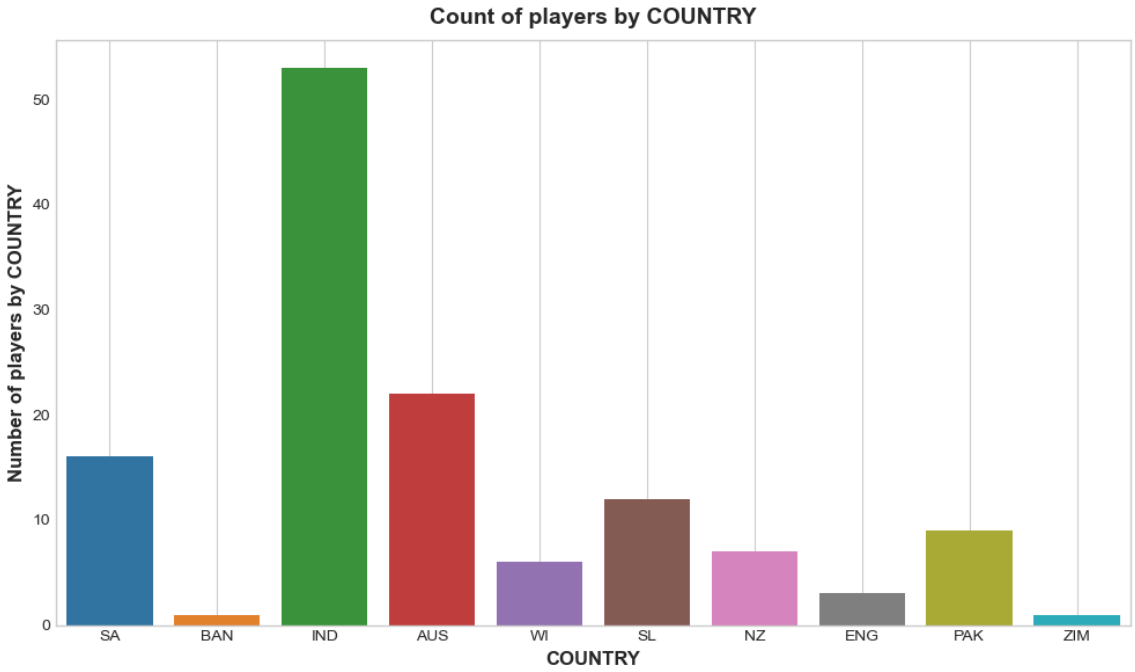
In [69]:

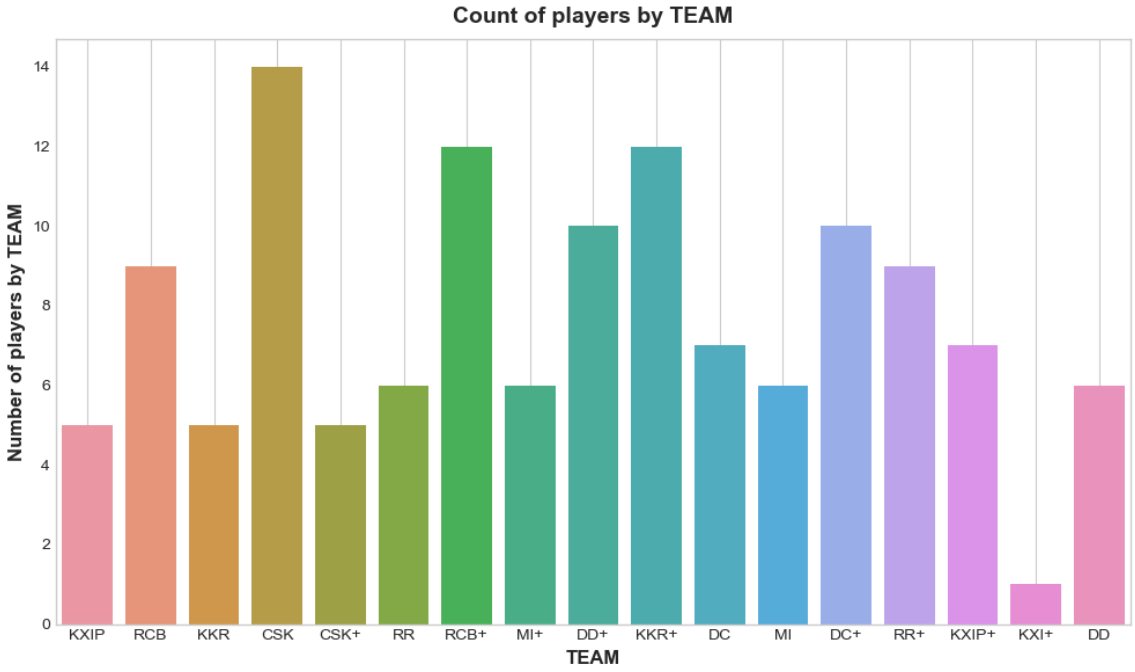
```
# Count the values in each categorical variable
for col in cat_col:
    print("total unique values in {} column: ".format(col), df[col].nunique())
print("----------------------------------------------------")
# count the values in each numerical variable:
for col in num_col:
    print("total unique values in {} column: ".format(col), df[col].nunique())
```

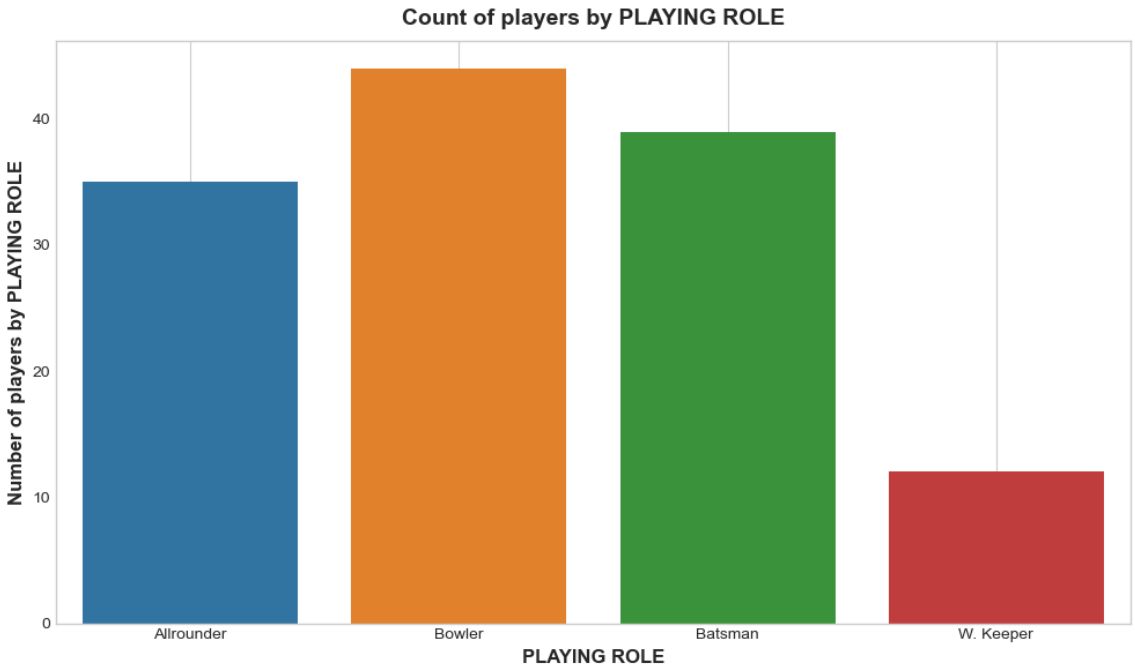
```
total unique values in PLAYER NAME column:  130
total unique values in COUNTRY column:  10
total unique values in TEAM column:  17
total unique values in PLAYING ROLE column:  4
----------------------------------------------------
total unique values in AGE column:  3
total unique values in T-RUNS column:  103
total unique values in T-WKTS column:  60
total unique values in ODI-RUNS-S column:  117
total unique values in ODI-SR-B column:  118
total unique values in ODI-WKTS column:  74
total unique values in ODI-SR-BL column:  82
total unique values in CAPTAINCY EXP column:  2
total unique values in RUNS-S column:  115
total unique values in HS column:  73
total unique values in AVE column:  113
total unique values in SR-B column:  125
total unique values in SIXERS column:  48
total unique values in RUNS-C column:  92
total unique values in WKTS column:  51
total unique values in AVE-BL column:  88
total unique values in ECON column:  83
total unique values in SR-BL column:  82
total unique values in AUCTION YEAR column:  4
total unique values in BASE PRICE column:  17
total unique values in SOLD PRICE column:  53
```

In [70]:

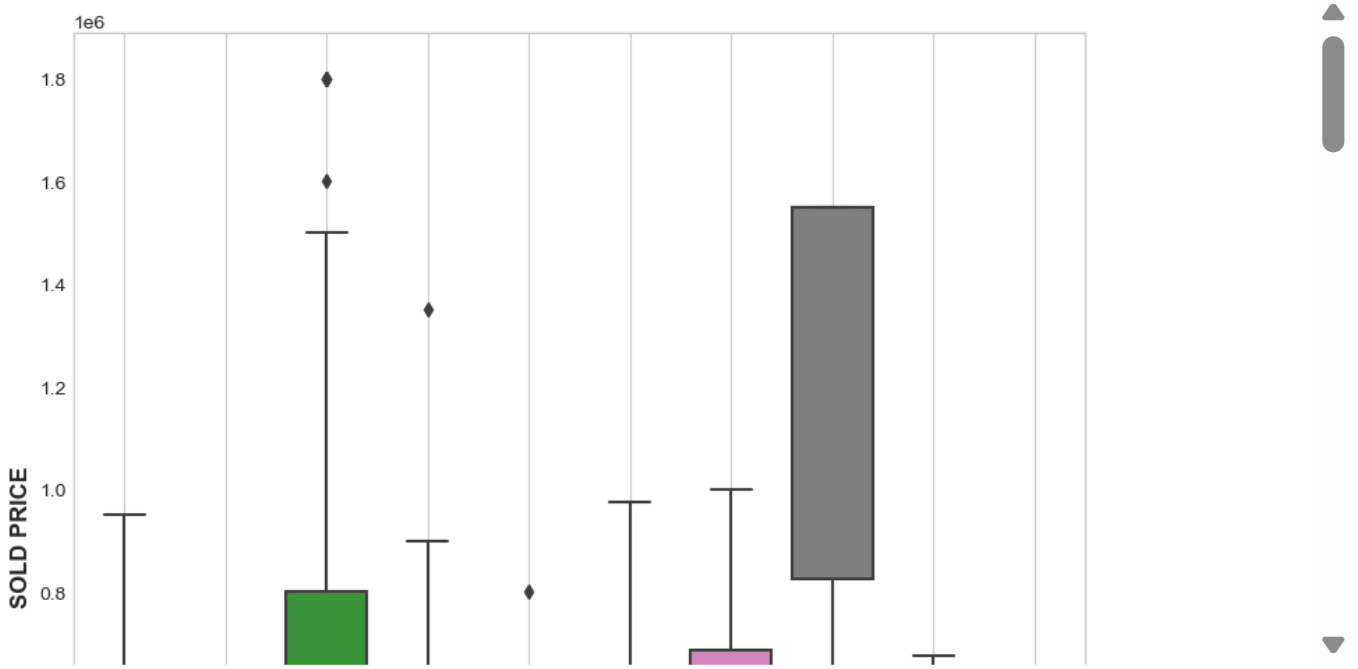
```
# count of players by country, team and playing role
cols = ['COUNTRY','TEAM','PLAYING ROLE']

def cat_plt_func(cols_list):
    for idx, col in enumerate(cols):
        plt.figure(idx, figsize=(10,6))
        sns.countplot(x=col, data=df)
        plt.title("Count of players by {}".format(col))
        plt.ylabel("Number of players by {}".format(col))
        plt.grid()
        plt.show()


cat_plt_func(cols)
```





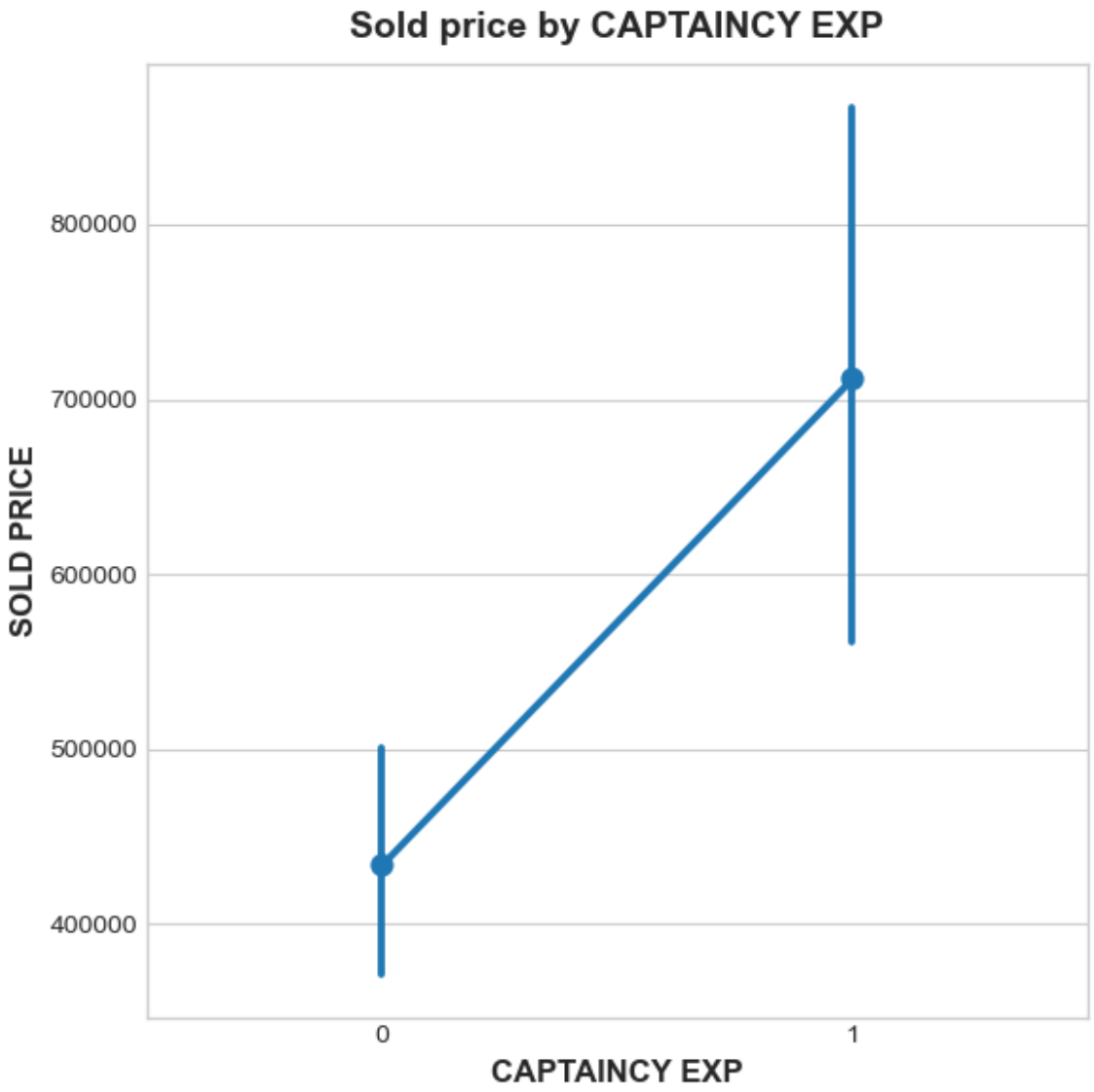



```
cols = ['COUNTRY','TEAM','PLAYING ROLE']
for idx, col in enumerate(cols):
    plt.figure(figsize=(8,8))
    sns.boxplot(x=col, y='SOLD PRICE', data=df)
    plt.yticks(list(range(200000,2000000,200000)))
    plt.grid()
    plt.show()
```



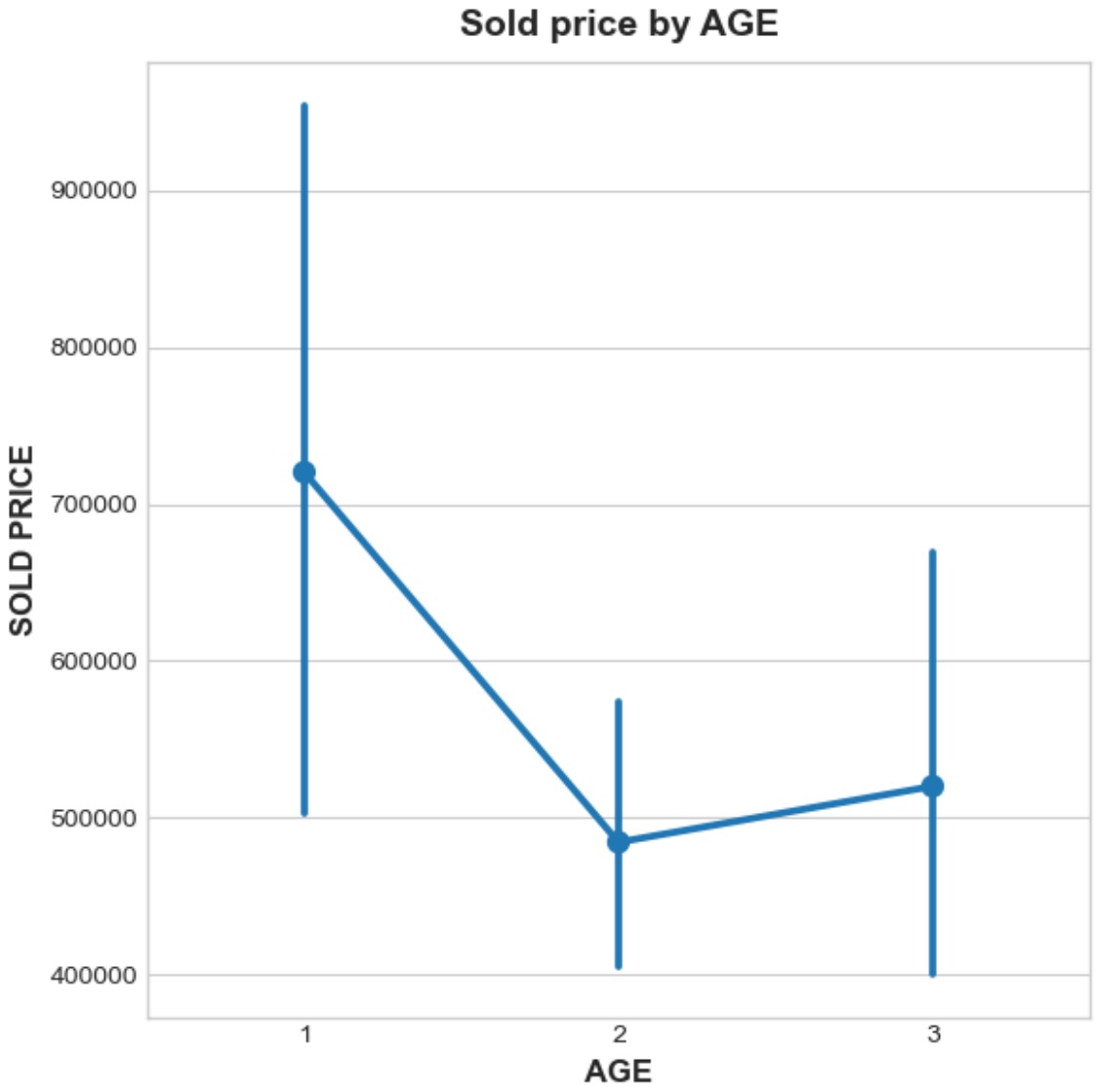
In [72]:

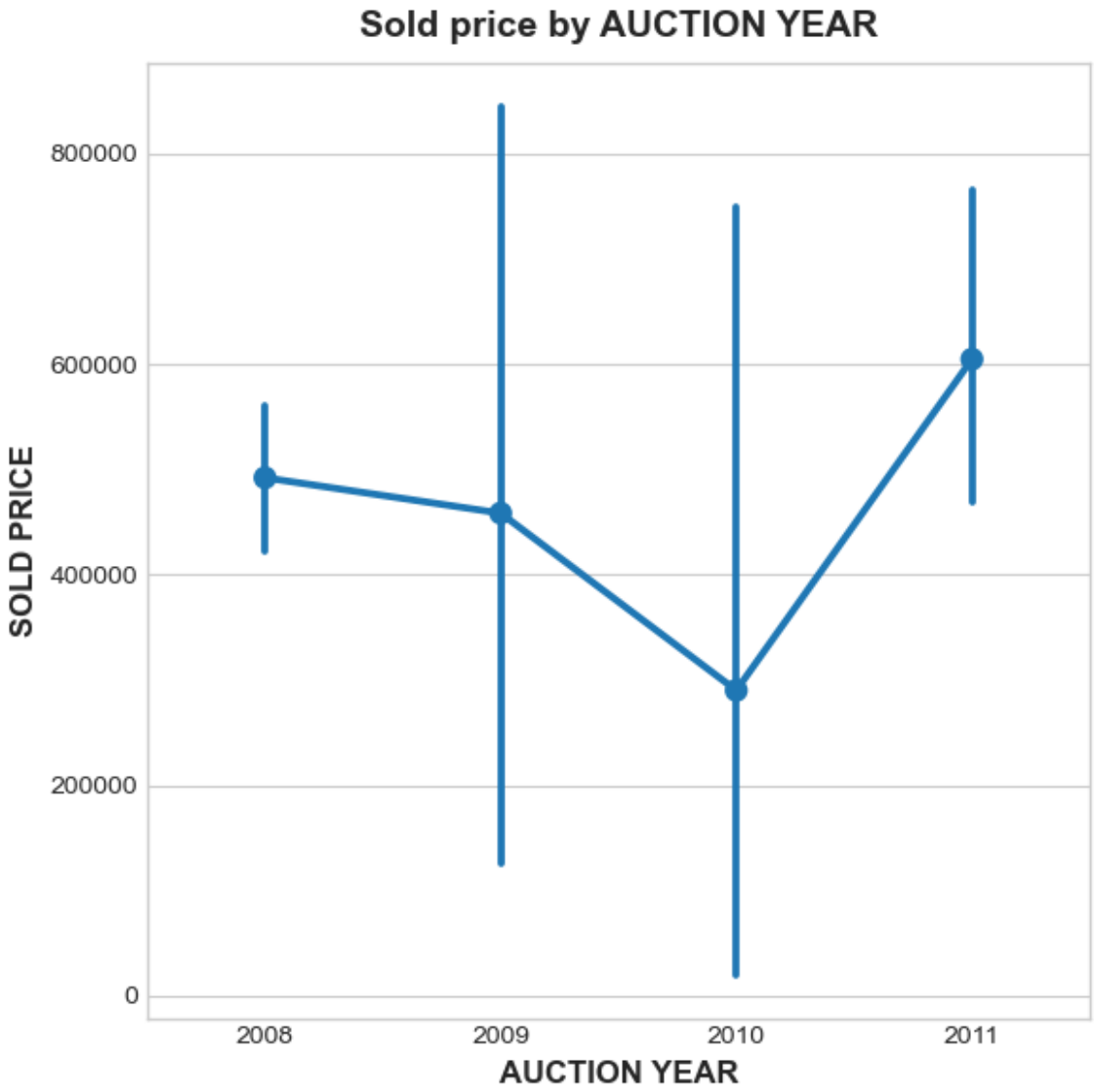
```
df['CAPTAINCY EXP'].value_counts()

def cat_func(col):
    plt.figure(figsize=(6,6))
    sns.pointplot(x=col, y='SOLD PRICE', data=df)
    plt.title("Sold price by {}".format(col))
    plt.show()

cat_func('CAPTAINCY EXP')
```



In [73]:

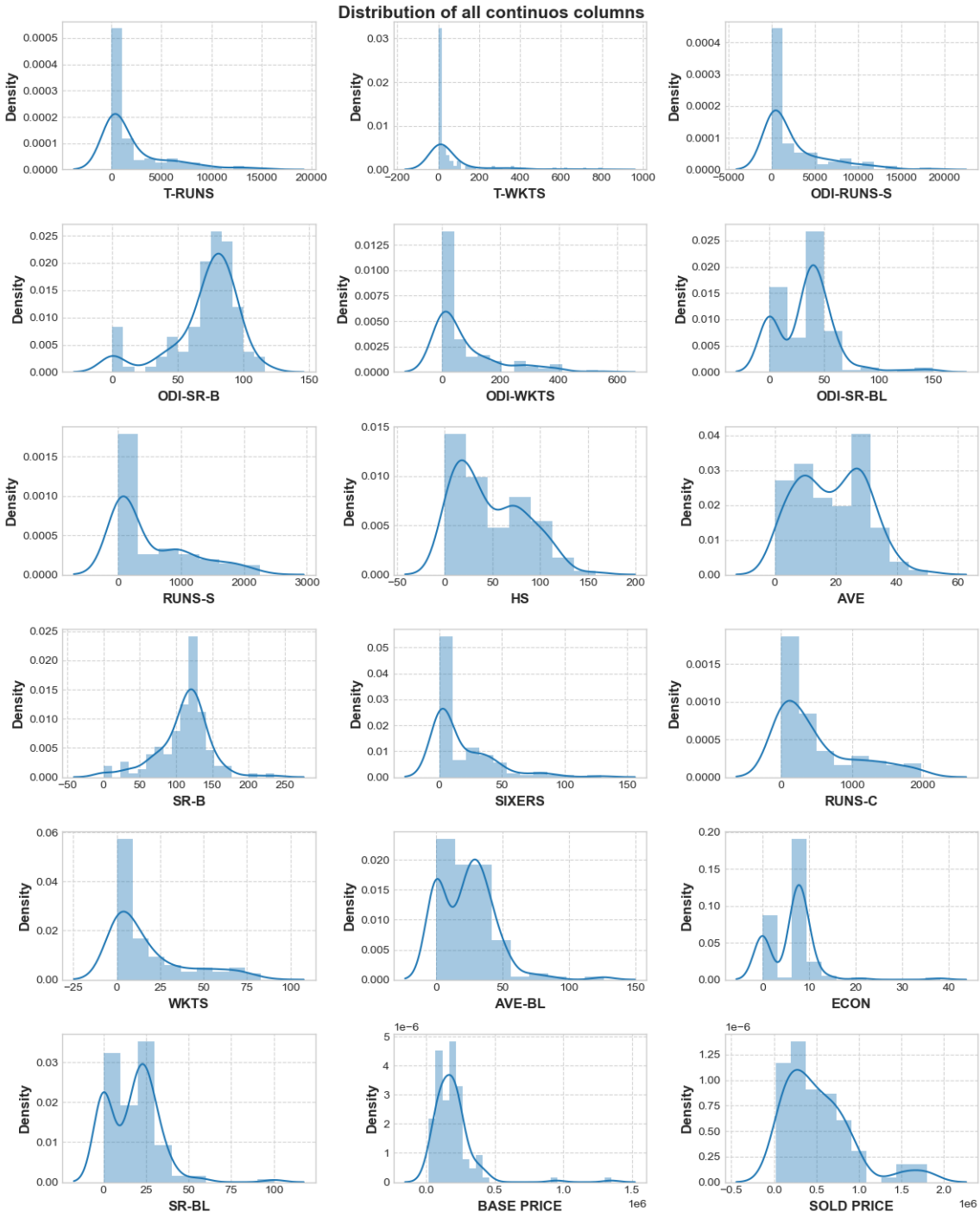
```
discreat_fea=['AGE','AUCTION YEAR']

for col in discreat_fea:
    cat_func(col)
```




Sold price by AUCTION YEAR
800000
600000
400000
200000
0
SOLD PRICE
2008
2009
2010
2011
AUCTION YEAR

In [74]:

```
new_col = [col for col in num_col if df[col].nunique() > 4]
len(new_col)

fig, ax = plt.subplots(6,3, figsize=(12,15))
for idx, col in enumerate(new_col):
    sns.distplot(x=df[col],
                 ax=ax[floor(idx/3),
                       idx%3])
    ax[floor(idx/3), idx%3].grid(visible=True,
                                 color='lightgrey',
                                 linestyle="--" )
    ax[floor(idx/3), idx%3].set_xlabel(col)

plt.suptitle("Distribution of all continuos columns",
             fontsize=15,
             fontweight='bold')
fig.tight_layout()
plt.show()
```


Distribution of all continuos columns
Density
T-RUNS
T-WKTS
ODI-RUNS-S
ODI-SR-B
ODI-WKTS
ODI-SR-BL
RUNS-S
HS
AVE
SR-B
SIXERS
RUNS-C
WKTS
AVE-BL
ECON
SR-BL
BASE PRICE
1e6
SOLD PRICE
1e6

In [75]:

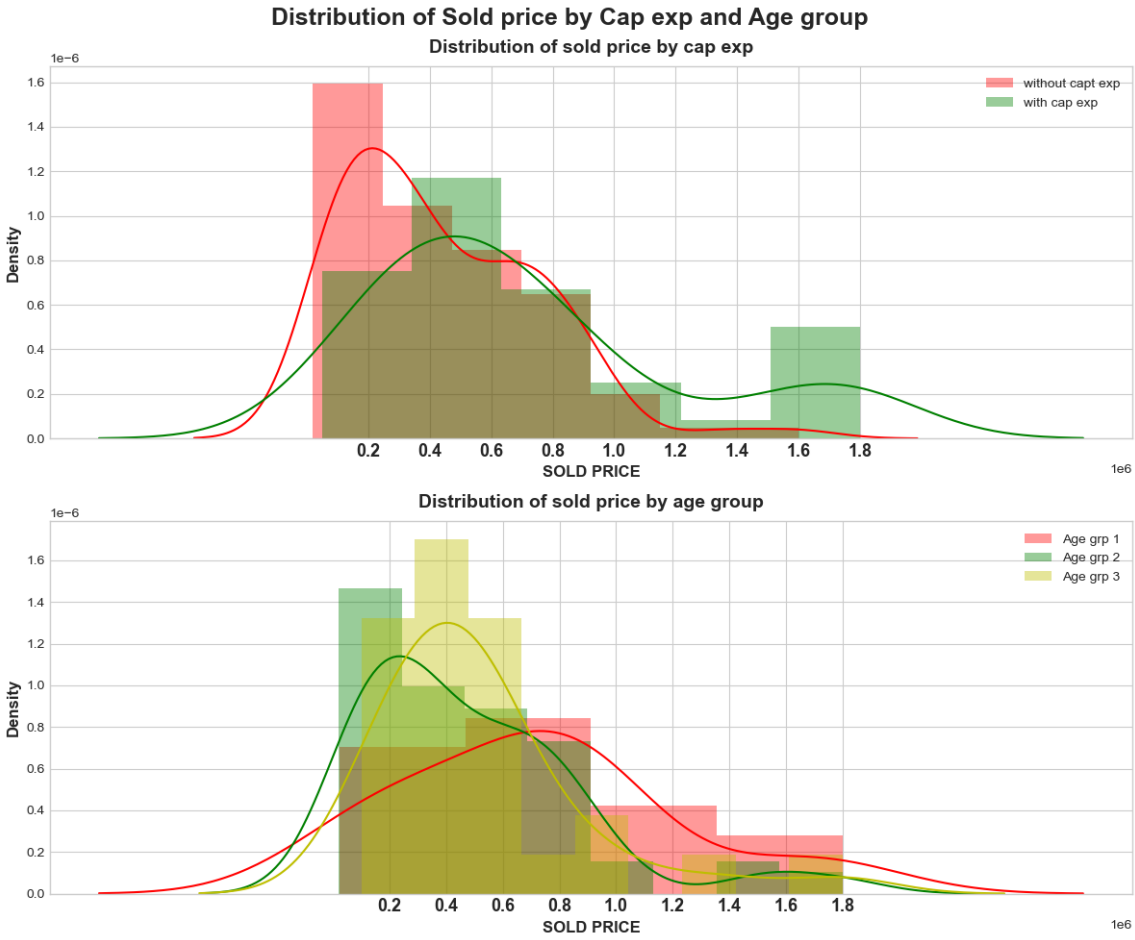
```
# cross tabulation
pd.crosstab(index=df['AGE'],
            columns=df['PLAYING ROLE'],
            values=df['SOLD PRICE'],
            aggfunc='count')
```

Out[75]:

PLAYING ROLE	Allrounder	Batsman	Bowler	W. Keeper
AGE				
1	4.0	5.0	7.0	NaN
2	25.0	21.0	29.0	11.0
3	6.0	13.0	8.0	1.0

In [76]:

```
x = df[df['CAPTAINCY EXP'] == 0]['SOLD PRICE']

plt.figure(figsize=(12,10))

plt.subplot(2,1,1)
sns.distplot(x=df[df['CAPTAINCY EXP'] == 0]['SOLD PRICE'], label='without capt exp', col
sns.distplot(x=df[df['CAPTAINCY EXP'] == 1]['SOLD PRICE'], label='with cap exp', color='
plt.xlabel("SOLD PRICE")
plt.xticks(list(range(200000,2000000,200000)),fontsize=13, fontweight='bold')
plt.title("Distribution of sold price by cap exp")
plt.legend()

plt.subplot(2,1,2)
sns.distplot(x=df[df['AGE'] == 1]['SOLD PRICE'], label='Age grp 1', color='r')
sns.distplot(x=df[df['AGE'] == 2]['SOLD PRICE'], label='Age grp 2', color='g')
sns.distplot(x=df[df['AGE'] == 3]['SOLD PRICE'], label='Age grp 3', color='y')
plt.xlabel("SOLD PRICE")
plt.xticks(list(range(200000,2000000,200000)),fontsize=13, fontweight='bold')
plt.title("Distribution of sold price by age group")
plt.legend()

plt.suptitle("Distribution of Sold price by Cap exp and Age group", fontsize=18, fontwei
fig.tight_layout()
plt.show()
```



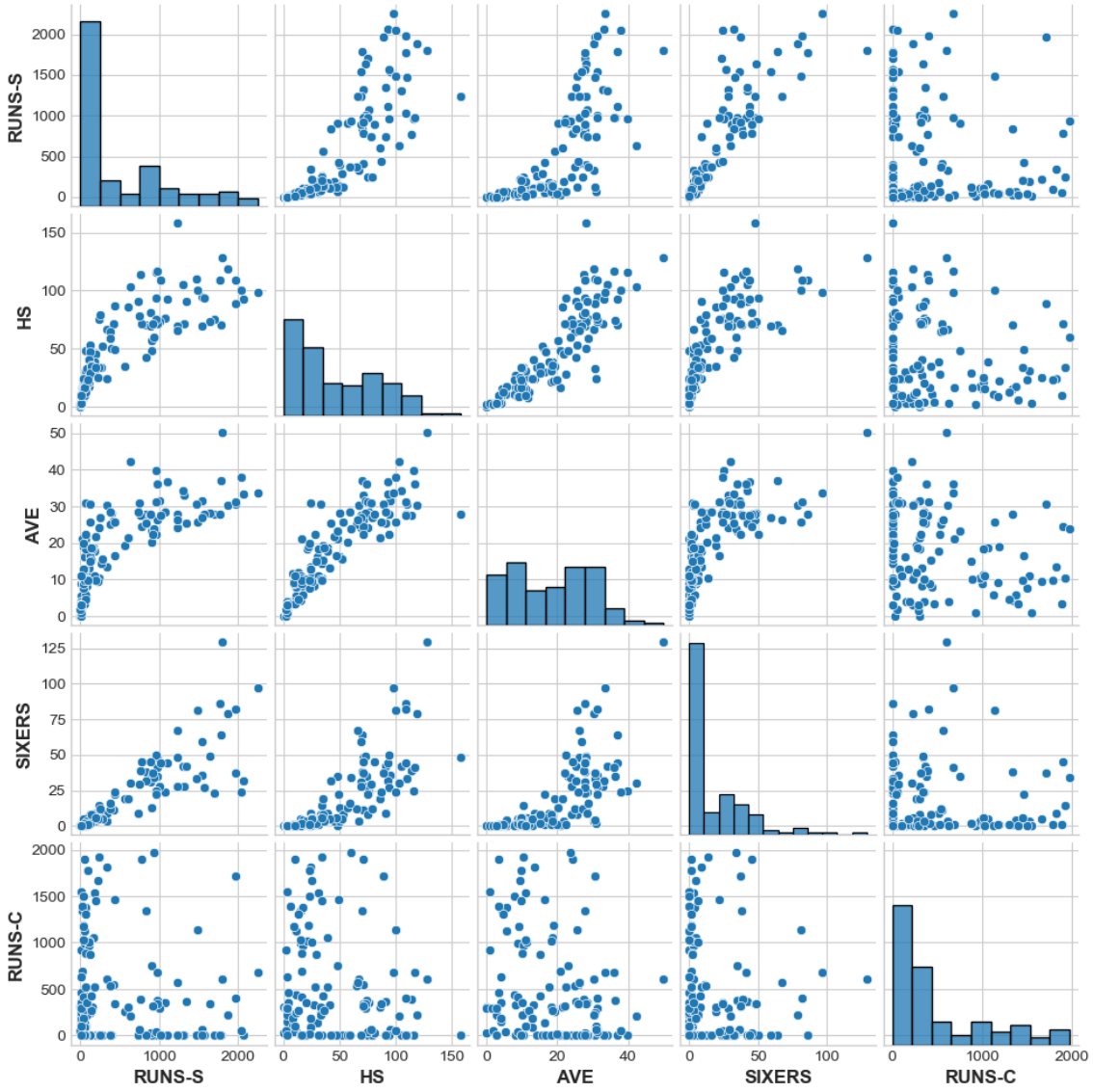
In [77]:

```
fig, axs = plt.subplots(6,3, figsize=(15,19),constrained_layout=True)
for idx, col in enumerate(new_col):
    axs[floor(idx/3), idx%3].scatter(np.array(df[col]),
                                     np.array(df['SOLD PRICE']),
                                     marker='o')
    axs[floor(idx/3), idx%3].set_xlabel(col)
    axs[floor(idx/3), idx%3].set_ylabel('Sold price')
plt.show()
```

In [78]:

```
# pairplot among most correlating columns
data = df.loc[:,['RUNS-S','HS','AVE','SIXERS','RUNS-C']]
plt.figure(figsize=(6,6))
sns.pairplot(data,
             height=2,
             kind='scatter')
plt.show()
```

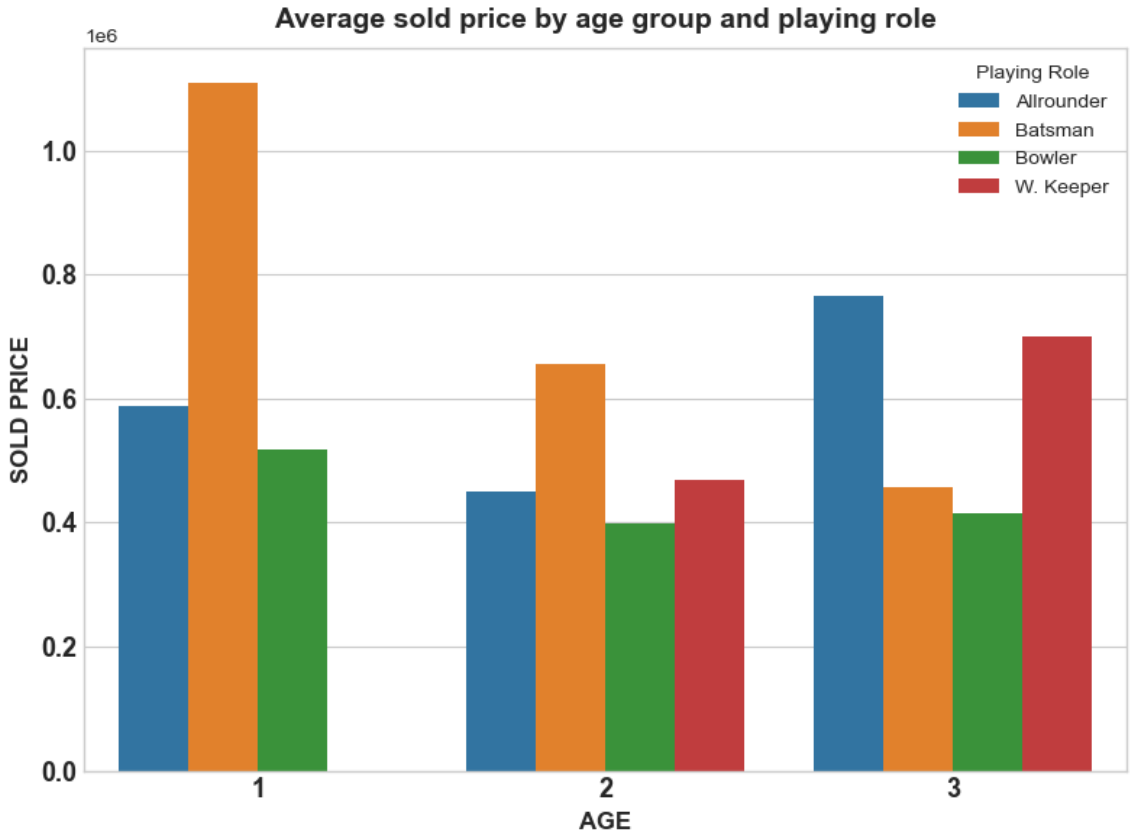
```
<Figure size 600x600 with 0 Axes>
```



In [79]:

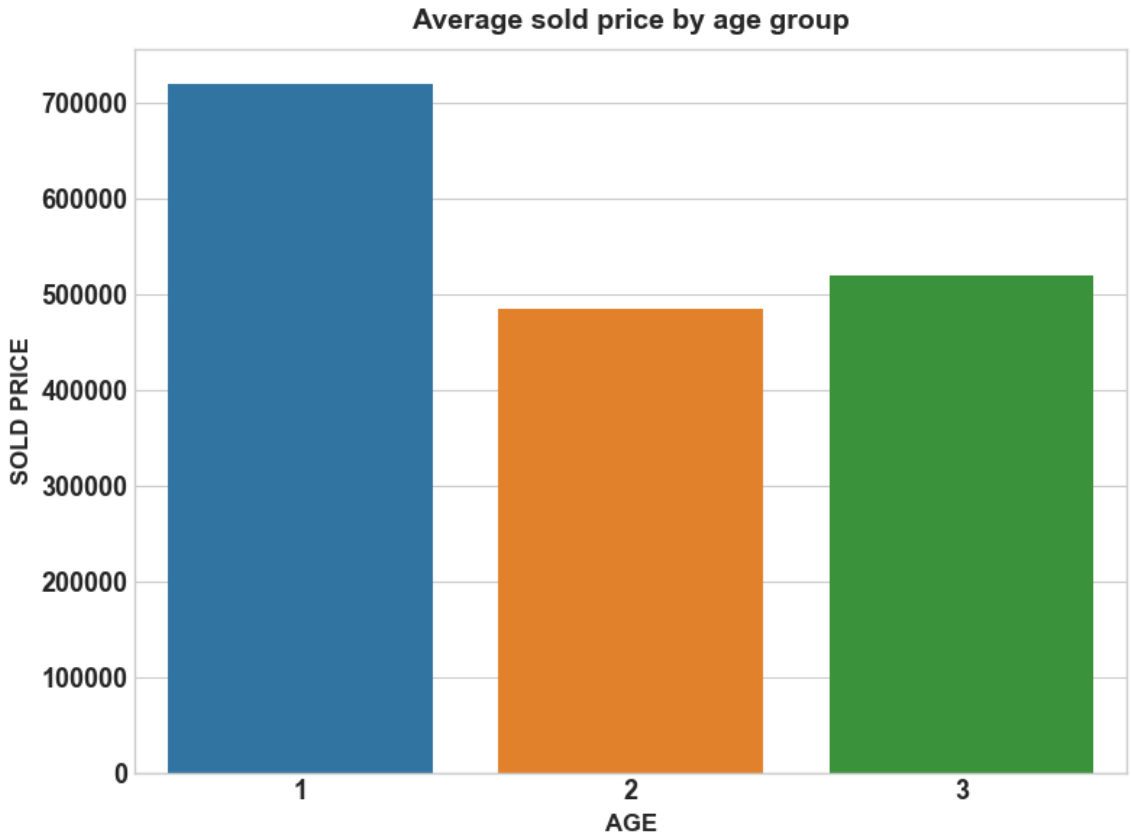
```
ave_prdf = pd.DataFrame(df.groupby(['AGE','PLAYING ROLE'])['SOLD PRICE'].mean()).reset_i
print(ave_prdf)
plt.figure(figsize=(8,6))
sns.barplot(x='AGE', y='SOLD PRICE', hue='PLAYING ROLE', data=ave_prdf)
plt.title("Average sold price by age group and playing role")
plt.xticks(fontsize=13, fontweight='bold')
plt.yticks(fontsize=13, fontweight='bold')
plt.legend(title='Playing Role')
plt.show()
```

```
    AGE PLAYING ROLE    SOLD PRICE
0     1   Allrounder  5.875000e+05
1     1      Batsman  1.110000e+06
2     1       Bowler  5.177143e+05
3     2   Allrounder  4.494000e+05
4     2      Batsman  6.547619e+05
5     2       Bowler  3.979310e+05
6     2    W. Keeper  4.677273e+05
7     3   Allrounder  7.666667e+05
8     3      Batsman  4.576923e+05
9     3       Bowler  4.143750e+05
10    3    W. Keeper  7.000000e+05
```



In [80]:

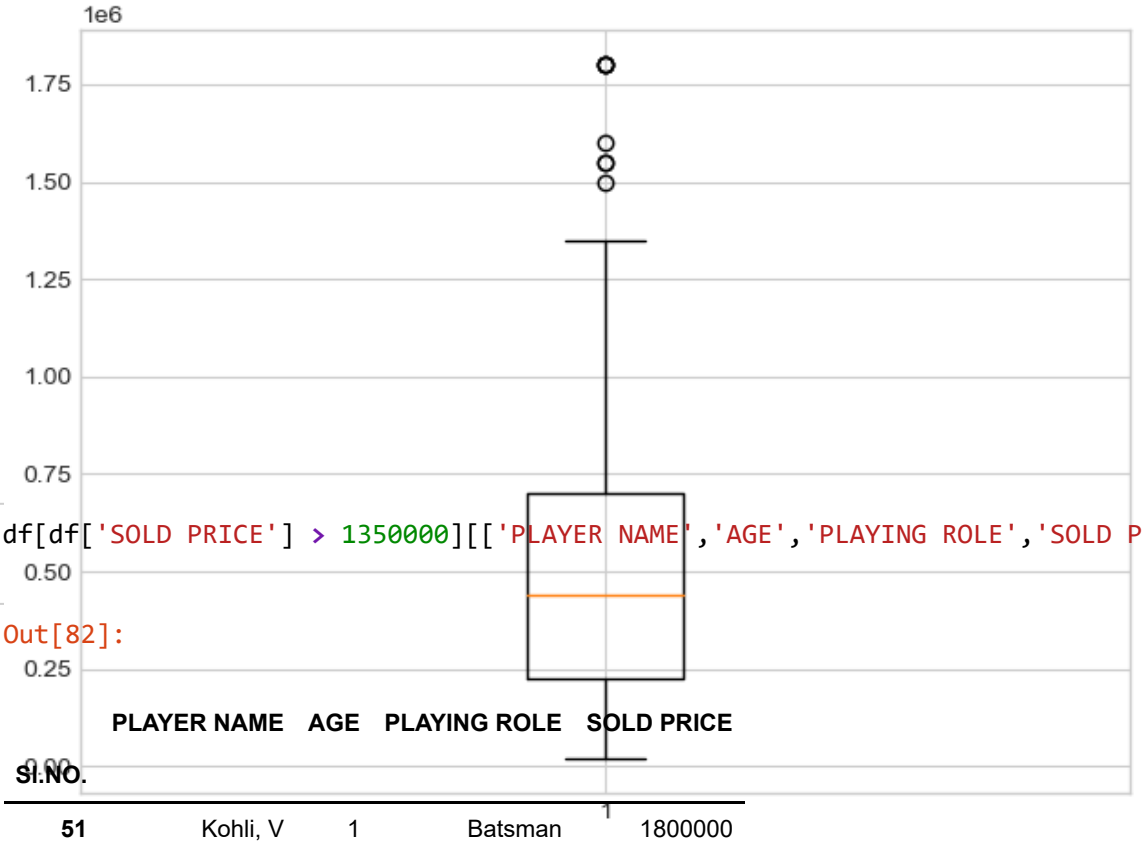
```
av_age = (df.groupby('AGE')['SOLD PRICE'].mean().reset_index()).sort_values(by='SOLD PRI
                                                                              ascending=Fa
plt.figure(figsize=(8,6))
sns.barplot(x='AGE', y='SOLD PRICE', data=av_age)
plt.title("Average sold price by age group")
plt.xticks(fontsize=13, fontweight='bold')
plt.yticks(fontsize=13, fontweight='bold')
plt.show()
plt.show()
```



In [81]:

```
box = plt.boxplot(df['SOLD PRICE'])
caps = [item.get_ydata()[0] for item in box['caps']]
quantiles =[item.get_ydata()[0] for item in box['whiskers']]
median = [item.get_ydata()[0] for item in box['medians']]
outliers = [item.get_ydata() for item in box['fliers']]


print("Distribution sold price has followong property {}".format(box))
print("------------------------------------------------------------")
print("------------------------------------------------------------")
print("Minimum sold price is {} and maximum sold price is {}".format(caps[0],caps[1]))
print("------------------------------------------------------------")
print("------------------------------------------------------------")
print('25th quantils is {} and 75th qunatile is {}'.format(quantiles[0],quantiles[1]))
print("------------------------------------------------------------")
print("------------------------------------------------------------")
print('Median of the sold price is {}'.format(median[0]))
print("------------------------------------------------------------")
print("------------------------------------------------------------")
print("Outliears in sold price are {}".format(outliers))
print("------------------------------------------------------------")
print("------------------------------------------------------------")
```

```
Distribution sold price has followong property {'whiskers': [<matplotlib.l
ines.Line2D object at 0x0000020FD5398D30>, <matplotlib.lines.Line2D object
at 0x0000020FD5176040>], 'caps': [<matplotlib.lines.Line2D object at 0x000
0020FD5176310>, <matplotlib.lines.Line2D object at 0x0000020FD51765E0>],
'boxes': [<matplotlib.lines.Line2D object at 0x0000020FD5398A60>], 'median
s': [<matplotlib.lines.Line2D object at 0x0000020FD51768B0>], 'fliers': [<
matplotlib.lines.Line2D object at 0x0000020FD5176B80>], 'means': []}
------------------------------------------------------------
------------------------------------------------------------
Minimum sold price is 20000 and maximum sold price is 1350000
------------------------------------------------------------
------------------------------------------------------------
25th quantils is 225000.0 and 75th qunatile is 700000.0
------------------------------------------------------------
------------------------------------------------------------
Median of the sold price is 437500.0
------------------------------------------------------------
------------------------------------------------------------
Outliears in sold price are [array([1500000, 1550000, 1800000, 1550000, 18
00000, 1800000, 1600000,
       1800000], dtype=int64)]
------------------------------------------------------------
------------------------------------------------------------
```




```
df[df['SOLD PRICE'] > 1350000][['PLAYER NAME','AGE','PLAYING ROLE','SOLD PRICE']].sort_v
```

Out[82]:

Sl.NO.	PLAYER NAME	AGE	PLAYING ROLE	SOLD PRICE
51	Kohli, V	1	Batsman	1800000
94	Sehwag, V	2	Batsman	1800000
112	Tendulkar, SR	3	Batsman	1800000
128	Yuvraj Singh	2	Batsman	1800000
114	Tiwary, SS	1	Batsman	1600000
24	Flintoff, A	2	Allrounder	1550000
84	Pietersen, KP	2	Batsman	1550000
16	Dhoni, MS	2	W. Keeper	1500000

In [83]:

```
lis = [row[0] for row in df[df['SOLD PRICE']> 1350000].itertuples(index=True, name=None)
lis
```

Out[83]:

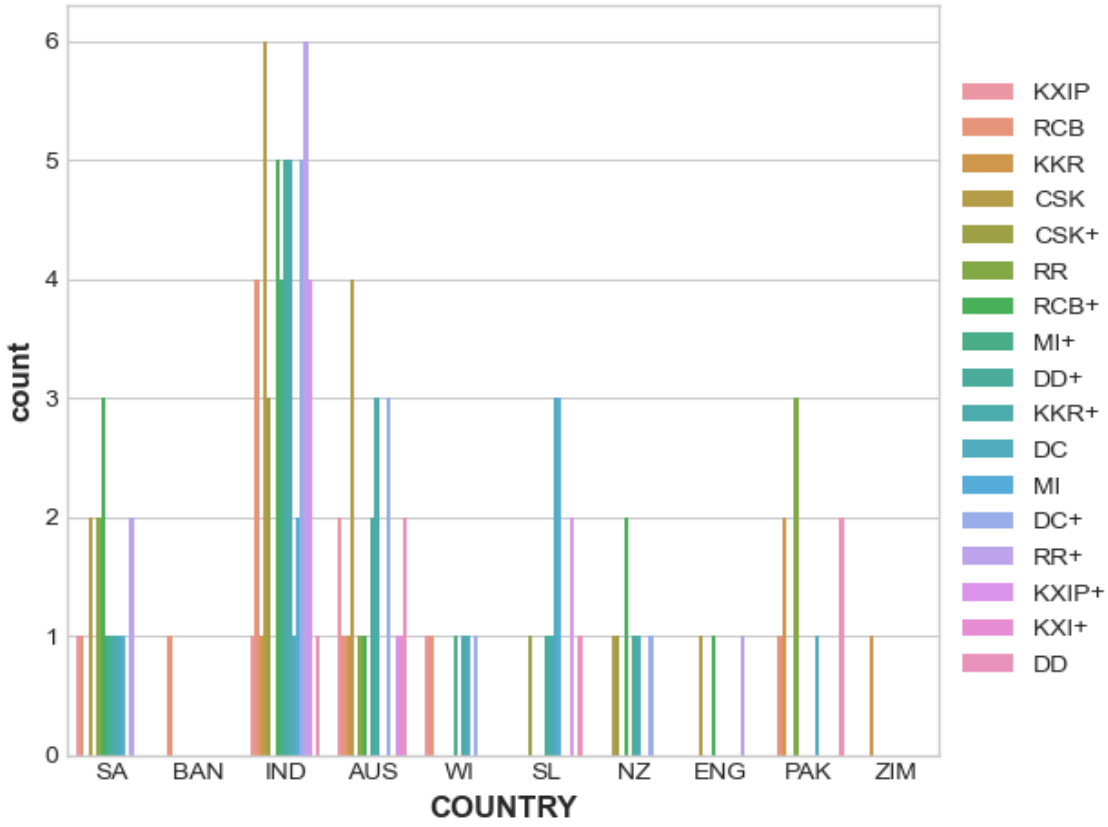
```
[16, 24, 51, 84, 94, 112, 114, 128]
```

In [84]:

```
sns.countplot('COUNTRY',hue='TEAM',data=df)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[84]:

```
<matplotlib.legend.Legend at 0x20fd536a6a0>
```

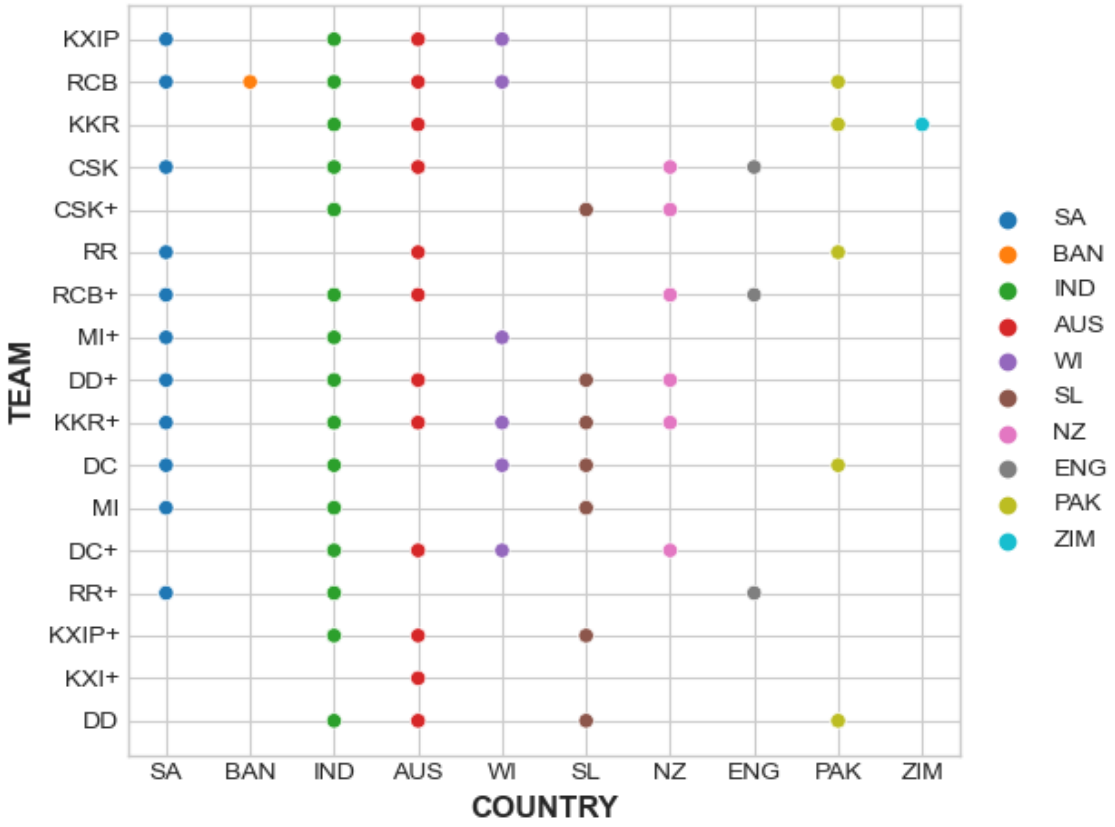


In [85]:

```
sns.scatterplot(df['COUNTRY'],df['TEAM'],hue=df['COUNTRY'])
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[85]:

```
<matplotlib.legend.Legend at 0x20fd51960d0>
```

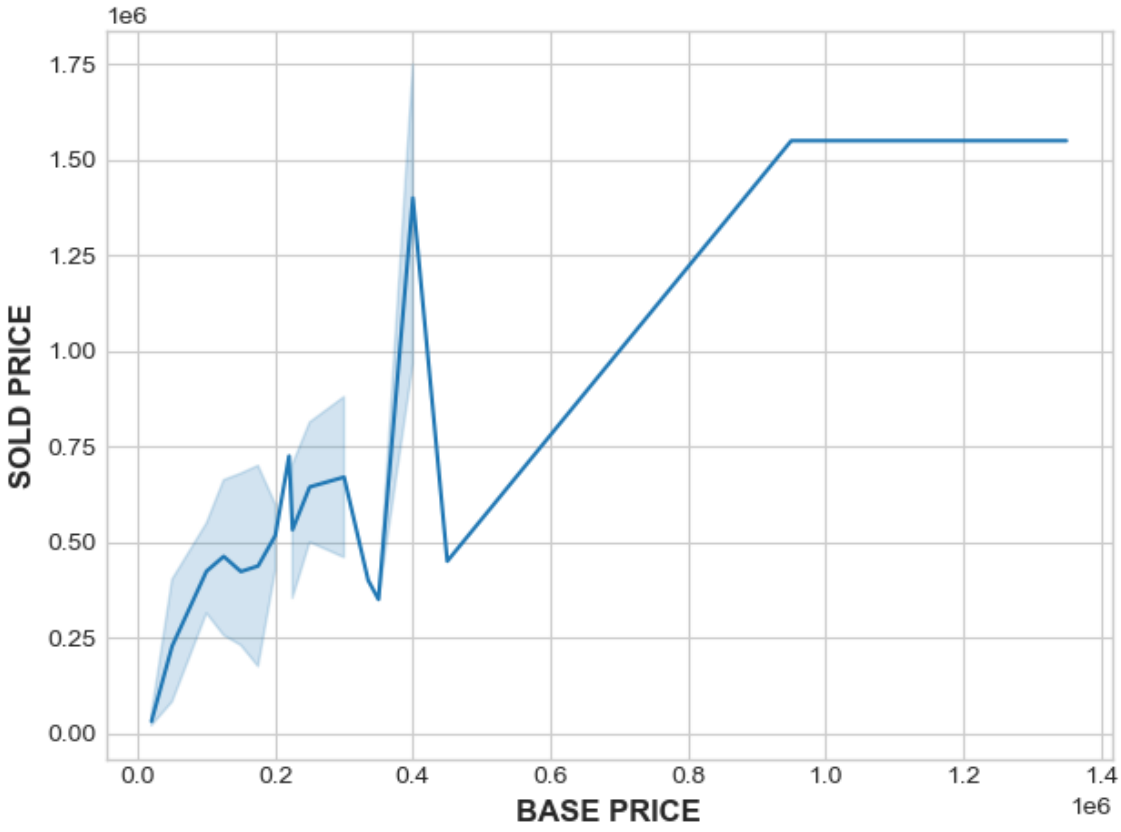


In [86]:

```
sns.lineplot(df['BASE PRICE'],df['SOLD PRICE'])
```

Out[86]:

```
<AxesSubplot:xlabel='BASE PRICE', ylabel='SOLD PRICE'>
```



In [87]:

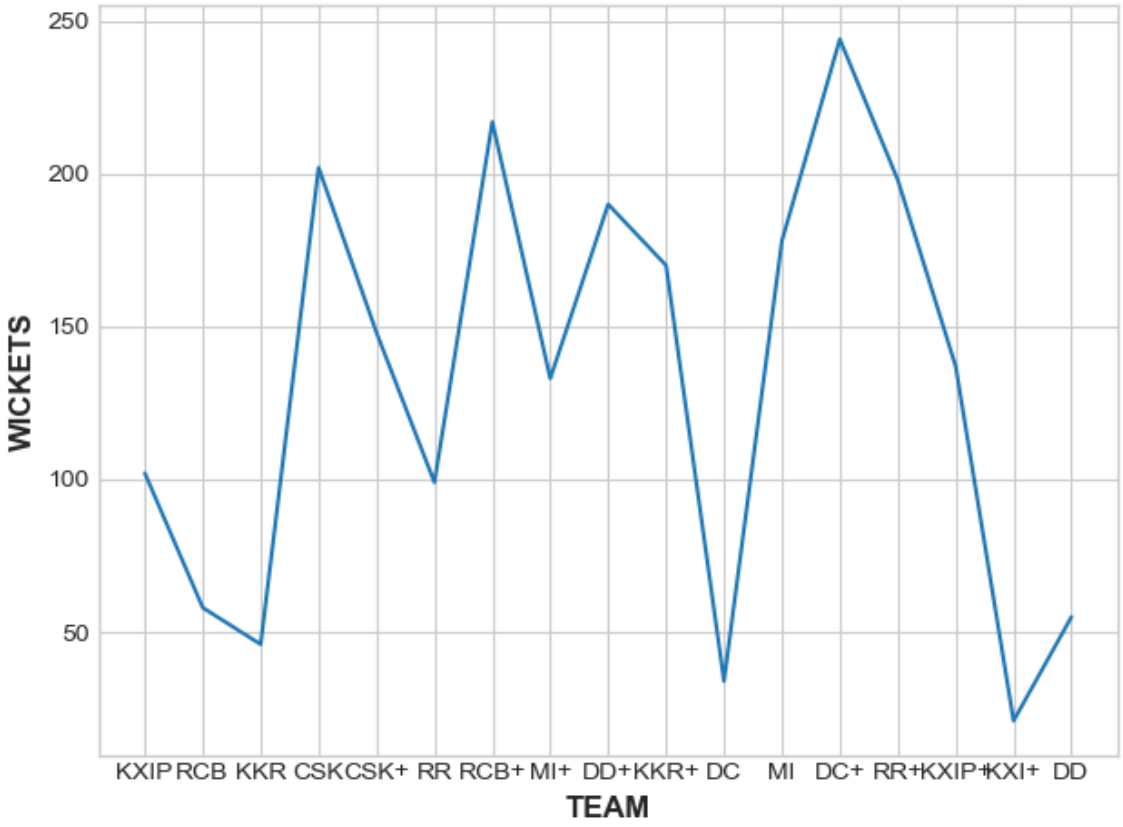
```
x=[]
y=list(pd.unique(df['TEAM']))
for i in y:
    c=df[df['TEAM']==i]['WKTS']
    x.append(sum(list(c)))
```

In [88]:

```
sns.lineplot(y,x)
plt.xlabel('TEAM')
plt.ylabel('WICKETS')
```

Out[88]:

```
Text(0, 0.5, 'WICKETS')
```

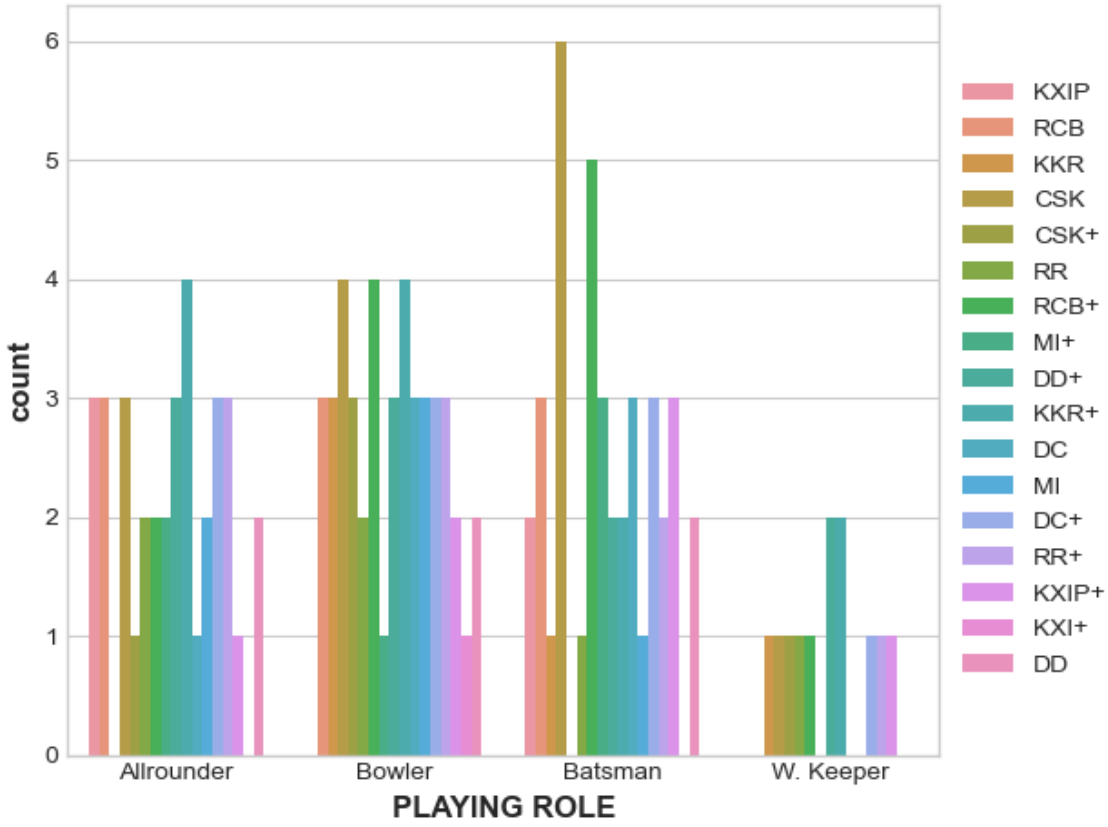


In [89]:

```
sns.countplot('PLAYING ROLE',hue='TEAM',data=df)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[89]:

```
<matplotlib.legend.Legend at 0x20fd53a9b20>
```

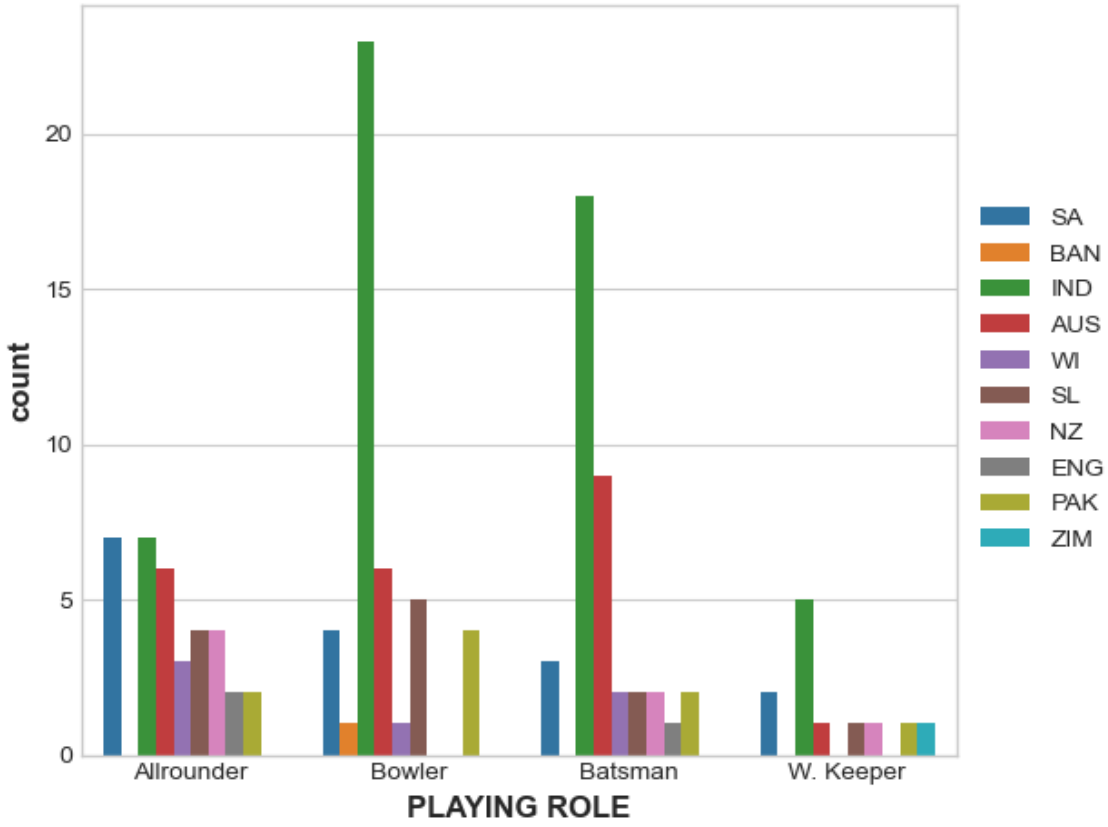


In [90]:

```
sns.countplot('PLAYING ROLE',hue='COUNTRY',data=df)
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[90]:

```
<matplotlib.legend.Legend at 0x20fd4b80c40>
```



In [91]:

```
c=list(df.columns[8:12])
c.append('TEAM')
```

In [92]:

```
sns.pairplot(df[c],diag_kind='kde',hue='TEAM')
```

Out[92]:

```
<seaborn.axisgrid.PairGrid at 0x20fd4611f40>
```

In [93]:

```
c.remove('TEAM')
c.append('COUNTRY')
```

In [94]:

```
sns.pairplot(df[c],diag_kind='kde',hue='COUNTRY')
```

Out[94]:

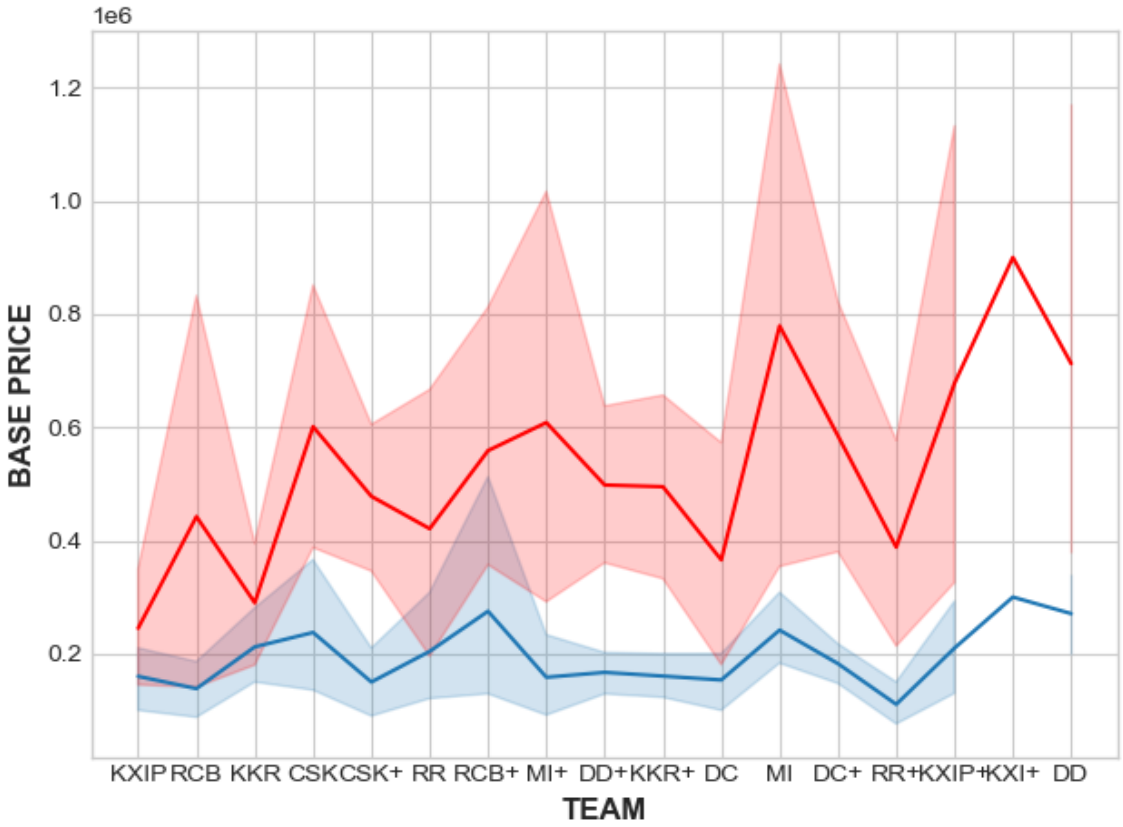
```
<seaborn.axisgrid.PairGrid at 0x20fcd0b7cd0>
```

In [95]:

```
sns.lineplot(df['TEAM'],df['BASE PRICE'])
sns.lineplot(df['TEAM'],df['SOLD PRICE'],color='r')
```

Out[95]:

```
<AxesSubplot:xlabel='TEAM', ylabel='BASE PRICE'>
```

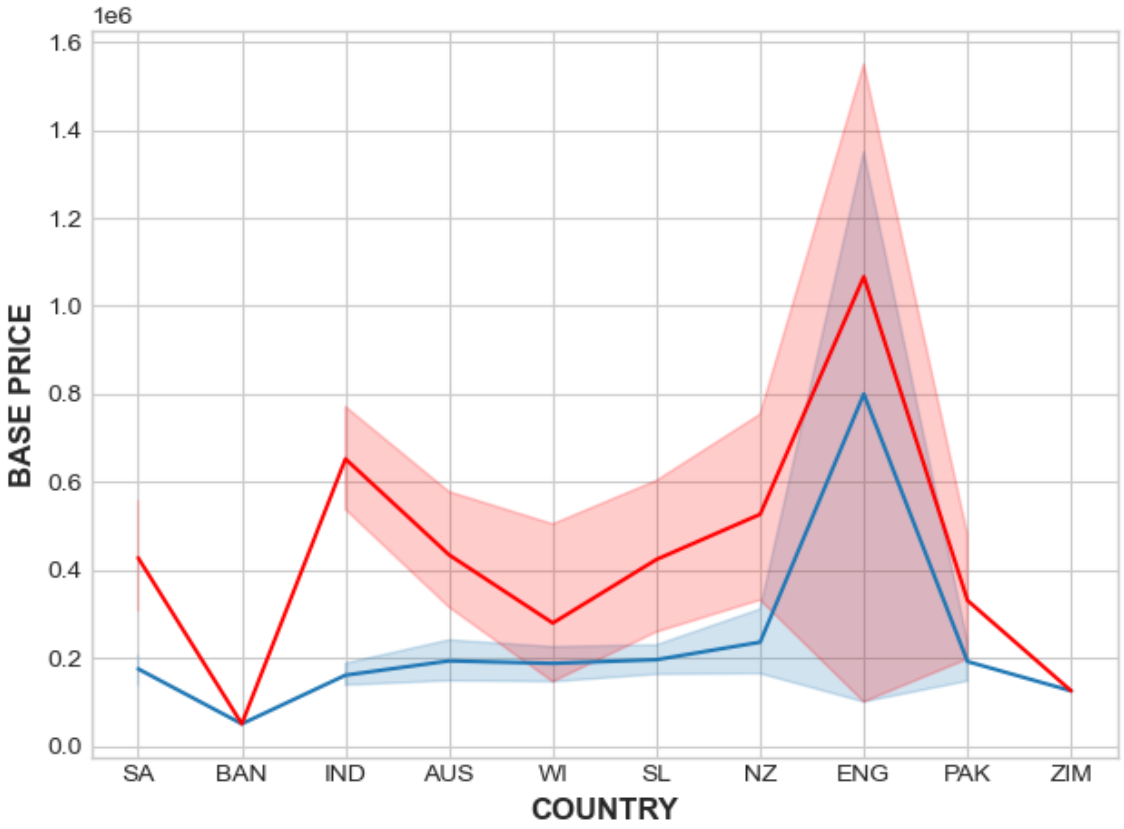


In [96]:

```
sns.lineplot(df['COUNTRY'],df['BASE PRICE'])
sns.lineplot(df['COUNTRY'],df['SOLD PRICE'],color='r')
```

Out[96]:

```
<AxesSubplot:xlabel='COUNTRY', ylabel='BASE PRICE'>
```



In [97]:

```
pre_df = df.copy()
pre_df.drop(['PLAYER NAME','AUCTION YEAR'], inplace=True, axis=1)
pre_df['PREMIUM'] = pre_df['SOLD PRICE'] - pre_df['BASE PRICE']
```

In [98]:

```
cap_df = pre_df.groupby('CAPTAINCY EXP')['SOLD PRICE'].mean().reset_index()
combined_df = pd.merge(pre_df, cap_df, how='left', on='CAPTAINCY EXP')
combined_df.rename(columns={'SOLD PRICE_x':'SOLD PRICE',
                    'SOLD PRICE_y':'AVE PRICE BY CAP'}, inplace=True)
coun_sp = pre_df.groupby('COUNTRY')['SOLD PRICE'].median().reset_index()
final_df = pd.merge(combined_df, coun_sp, how='left', on='COUNTRY')
final_df.rename(columns={'SOLD PRICE_x':'SOLD PRICE',
                    'SOLD PRICE_y':'AVE PRICE BY COUNTRY'}, inplace=True)
encoded_df = pd.get_dummies(final_df,
                            columns=['COUNTRY', 'TEAM', 'PLAYING ROLE'],
                            drop_first=True)
lis = [row[0] for row in df[df['SOLD PRICE']> 1350000].itertuples(index=True, name=None)
after_df = encoded_df.drop(lis, axis=0)

print("Final columns for model training: \n", encoded_df.columns)
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Shape of the final dataset: ", encoded_df.shape)
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Data types of columns:\n", encoded_df.dtypes)
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Shape of after_df: ", after_df.shape)
```

```
Final columns for model training:
 Index(['AGE', 'T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B', 'ODI-WKTS',
       'ODI-SR-BL', 'CAPTAINCY EXP', 'RUNS-S', 'HS', 'AVE', 'SR-B', 'SIXER
S',
       'RUNS-C', 'WKTS', 'AVE-BL', 'ECON', 'SR-BL', 'BASE PRICE', 'SOLD PR
ICE',
       'PREMIUM', 'AVE PRICE BY CAP', 'AVE PRICE BY COUNTRY', 'COUNTRY_BA
N',
       'COUNTRY_ENG', 'COUNTRY_IND', 'COUNTRY_NZ', 'COUNTRY_PAK', 'COUNTRY
_SA',
       'COUNTRY_SL', 'COUNTRY_WI', 'COUNTRY_ZIM', 'TEAM_CSK+', 'TEAM_DC',
       'TEAM_DC+', 'TEAM_DD', 'TEAM_DD+', 'TEAM_KKR', 'TEAM_KKR+', 'TEAM_K
XI+',
       'TEAM_KXIP', 'TEAM_KXIP+', 'TEAM_MI', 'TEAM_MI+', 'TEAM_RCB',
       'TEAM_RCB+', 'TEAM_RR', 'TEAM_RR+', 'PLAYING ROLE_Batsman',
       'PLAYING ROLE_Bowler', 'PLAYING ROLE_W. Keeper'],
      dtype='object')
------------------------------------------------------------
------------------------------------------------------------
Shape of the final dataset:  (130, 51)
------------------------------------------------------------
------------------------------------------------------------
Data types of columns:
 AGE                          int64
T-RUNS                        int64
T-WKTS                        int64
ODI-RUNS-S                    int64
ODI-SR-B                    float64
ODI-WKTS                      int64
ODI-SR-BL                   float64
CAPTAINCY EXP                 int64
RUNS-S                        int64
HS                            int64
AVE                         float64
SR-B                        float64
SIXERS                        int64
RUNS-C                        int64
WKTS                          int64
AVE-BL                      float64
ECON                        float64
SR-BL                       float64
BASE PRICE                    int64
SOLD PRICE                    int64
PREMIUM                       int64
AVE PRICE BY CAP            float64
AVE PRICE BY COUNTRY        float64
COUNTRY_BAN                   uint8
COUNTRY_ENG                   uint8
COUNTRY_IND                   uint8
COUNTRY_NZ                    uint8
COUNTRY_PAK                   uint8
COUNTRY_SA                    uint8
COUNTRY_SL                    uint8
COUNTRY_WI                    uint8
COUNTRY_ZIM                   uint8
TEAM_CSK+                     uint8
TEAM_DC                       uint8
TEAM_DC+                      uint8
TEAM_DD                       uint8
TEAM_DD+                      uint8
TEAM_KKR                      uint8
```

```
TEAM_KKR+                   uint8
TEAM_KXI+                   uint8
TEAM_KXIP                   uint8
TEAM_KXIP+                  uint8
TEAM_MI                     uint8
TEAM_MI+                    uint8
TEAM_RCB                    uint8
TEAM_RCB+                   uint8
TEAM_RR                     uint8
TEAM_RR+                    uint8
PLAYING ROLE_Batsman        uint8
PLAYING ROLE_Bowler         uint8
PLAYING ROLE_W. Keeper      uint8
dtype: object
------------------------------------------------------------
------------------------------------------------------------
Shape of after_df:  (122, 51)
```

In [99]:

```
train_test_df = after_df.copy()

X = sm.add_constant(train_test_df.drop('SOLD PRICE', axis=1))
y = train_test_df['SOLD PRICE']

X.shape, y.shape

# devide train and test data set

train_X, val_X, train_y, val_y  = train_test_split(X, y,
                                                   train_size=0.8,
                                                   test_size=0.2,
                                                   random_state=42)
```

In [100]:

```
# let's build baseline model

sm_model = sm.OLS(train_y,train_X).fit()
sm_model.summary2()
```

Out[100]:

Model:	OLS	Adj. R-squared:	1.000
Dependent Variable:	SOLD PRICE	AIC:	-3759.6502
Date:	2023-06-25 16:57	BIC:	-3633.4893
No. Observations:	97	Log-Likelihood:	1928.8
Df Model:	48	F-statistic:	5.324e+29
Df Residuals:	48	Prob (F-statistic):	0.00
R-squared:	1.000	Scale:	6.3291e-19

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	-0.0000	0.0000	-0.9645	0.3396	-0.0000	0.0000
AGE	0.0000	0.0000	0.3938	0.6954	-0.0000	0.0000
T-RUNS	0.0000	0.0000	0.2262	0.8220	-0.0000	0.0000
T-WKTS	0.0000	0.0000	0.4842	0.6305	-0.0000	0.0000
ODI-RUNS-S	-0.0000	0.0000	-0.2704	0.7880	-0.0000	0.0000
ODI-SR-B	0.0000	0.0000	0.6456	0.5216	-0.0000	0.0000
ODI-WKTS	-0.0000	0.0000	-0.1718	0.8643	-0.0000	0.0000
ODI-SR-BL	0.0000	0.0000	0.3400	0.7353	-0.0000	0.0000
CAPTAINCY EXP	0.0000	0.0000	1.0221	0.3119	-0.0000	0.0000
RUNS-S	-0.0000	0.0000	-0.6733	0.5040	-0.0000	0.0000
HS	0.0000	0.0000	0.2952	0.7692	-0.0000	0.0000
AVE	0.0000	0.0000	0.4336	0.6665	-0.0000	0.0000
SR-B	-0.0000	0.0000	-0.3383	0.7366	-0.0000	0.0000
SIXERS	-0.0000	0.0000	-0.0050	0.9960	-0.0000	0.0000
RUNS-C	0.0000	0.0000	0.8597	0.3942	-0.0000	0.0000
WKTS	0.0000	0.0000	0.0928	0.9265	-0.0000	0.0000
AVE-BL	-0.0000	0.0000	-0.6362	0.5276	-0.0000	0.0000
ECON	-0.0000	0.0000	-0.2431	0.8089	-0.0000	0.0000
SR-BL	0.0000	0.0000	0.7202	0.4749	-0.0000	0.0000
BASE PRICE	1.0000	0.0000	554448940293501.6875	0.0000	1.0000	1.0000
PREMIUM	1.0000	0.0000	2495837090796881.0000	0.0000	1.0000	1.0000
AVE PRICE BY CAP	-0.0000	0.0000	-0.6630	0.5105	-0.0000	0.0000
AVE PRICE BY COUNTRY	0.0000	0.0000	0.0801	0.9365	-0.0000	0.0000
COUNTRY_BAN	-0.0000	0.0000	-0.0473	0.9625	-0.0000	0.0000
COUNTRY_ENG	0.0000	0.0000	0.9193	0.3625	-0.0000	0.0000
COUNTRY_IND	0.0000	0.0000	0.3006	0.7650	-0.0000	0.0000
COUNTRY_NZ	-0.0000	0.0000	-0.8796	0.3834	-0.0000	0.0000
COUNTRY_PAK	0.0000	0.0000	1.4659	0.1492	-0.0000	0.0000
COUNTRY_SA	0.0000	0.0000	0.4646	0.6443	-0.0000	0.0000
COUNTRY_SL	0.0000	0.0000	0.4440	0.6591	-0.0000	0.0000

COUNTRY_WI	0.0000	0.0000	0.6479	0.5201	-0.0000	0.0000
COUNTRY_ZIM	0.0000	0.0000	1.0850	0.2833	-0.0000	0.0000
TEAM_CSK+	-0.0000	0.0000	-0.2692	0.7890	-0.0000	0.0000
TEAM_DC	-0.0000	0.0000	-0.5239	0.6028	-0.0000	0.0000
TEAM_DC+	0.0000	0.0000	0.1360	0.8924	-0.0000	0.0000
TEAM_DD	0.0000	0.0000	0.3989	0.6917	-0.0000	0.0000
TEAM_DD+	0.0000	0.0000	0.1277	0.8989	-0.0000	0.0000
TEAM_KKR	-0.0000	0.0000	-0.3550	0.7241	-0.0000	0.0000
TEAM_KKR+	-0.0000	0.0000	-0.2209	0.8261	-0.0000	0.0000
TEAM_KXI+	0.0000	0.0000	0.0968	0.9233	-0.0000	0.0000
TEAM_KXIP	-0.0000	0.0000	-0.0197	0.9844	-0.0000	0.0000
TEAM_KXIP+	0.0000	0.0000	0.7651	0.4480	-0.0000	0.0000
TEAM_MI	-0.0000	0.0000	-0.7047	0.4844	-0.0000	0.0000
TEAM_MI+	0.0000	0.0000	0.0000	1.0000	-0.0000	0.0000
TEAM_RCB	0.0000	0.0000	0.1952	0.8460	-0.0000	0.0000
TEAM_RCB+	0.0000	0.0000	0.5155	0.6086	-0.0000	0.0000
TEAM_RR	-0.0000	0.0000	-0.2257	0.8224	-0.0000	0.0000
TEAM_RR+	0.0000	0.0000	0.3587	0.7214	-0.0000	0.0000
PLAYING ROLE_Batsman	-0.0000	0.0000	-0.0696	0.9448	-0.0000	0.0000
PLAYING ROLE_Bowler	-0.0000	0.0000	-0.2562	0.7989	-0.0000	0.0000
PLAYING ROLE_W. Keeper	-0.0000	0.0000	-0.2704	0.7880	-0.0000	0.0000

Omnibus:	4.252	Durbin-Watson:	1.711
Prob(Omnibus):	0.119	Jarque-Bera (JB):	4.262
Skew:	-0.500	Prob(JB):	0.119
Kurtosis:	2.770	Condition No.:	16111950981011584

In [101]:

```
# vif calculator
def get_vif(X):
    #X_matrix = X.as_matrix()
    vif = [variance_inflation_factor(X, i) for i in range(X.shape[1])]
    vif_factors = pd.DataFrame({'COLUMN':X.columns,
                                'VIF': vif})
    return vif_factors

vif_factors = get_vif(X)
vif_factors.head(10)
```

Out[101]:

	COLUMN	VIF
0	const	0.000000
1	AGE	2.412930
2	T-RUNS	10.923642
3	T-WKTS	8.318430
4	ODI-RUNS-S	12.618828
5	ODI-SR-B	1.880863
6	ODI-WKTS	11.433963
7	ODI-SR-BL	2.039362
8	CAPTAINCY EXP	inf
9	RUNS-S	13.207284

In [102]:

```
vif_greater_than_4 = vif_factors[vif_factors.VIF > 4][['COLUMN','VIF']]
vif_greater_than_4[:10]
```

Out[102]:

	COLUMN	VIF
2	T-RUNS	10.923642
3	T-WKTS	8.318430
4	ODI-RUNS-S	12.618828
6	ODI-WKTS	11.433963
8	CAPTAINCY EXP	inf
9	RUNS-S	13.207284
10	HS	11.138260
11	AVE	12.153290
13	SIXERS	8.043308
14	RUNS-C	29.755100

In [103]:

```
columns_keep = ['T-WKTS','HS','WKTS','SR-BL','AVE PRICE BY CAP']

vif_greater_less_4 = list(vif_factors[vif_factors.VIF < 4]['COLUMN'])

#new columns to keep

new_lis = columns_keep + vif_greater_less_4
X_new = X[new_lis]

get_vif(X_new)[:10]
```

Out[103]:

	COLUMN	VIF
0	T-WKTS	1.778871
1	HS	3.065363
2	WKTS	2.085018
3	SR-BL	3.119854
4	AVE PRICE BY CAP	1.960487
5	const	80.201282
6	AGE	1.641878
7	ODI-SR-B	1.603011
8	ODI-SR-BL	1.794891
9	SR-B	1.879593

In [104]:

```
# fit the model again
X_train = train_X[new_lis]

sm_model2 = sm.OLS(train_y, X_train).fit()
sm_model2.summary2()
```

Out[104]:

Model:	OLS	Adj. R-squared:	0.845
Dependent Variable:	SOLD PRICE	AIC:	2626.7042
Date:	2023-06-25 16:57	BIC:	2706.5202
No. Observations:	97	Log-Likelihood:	-1282.4
Df Model:	30	F-statistic:	18.41
Df Residuals:	66	Prob (F-statistic):	6.36e-22
R-squared:	0.893	Scale:	2.6158e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
T-WKTS	240.2722	187.8605	1.2790	0.2054	-134.8035	615.3478
HS	337.7315	787.2649	0.4290	0.6693	-1234.0935	1909.5565
WKTS	-149.9109	1117.5763	-0.1341	0.8937	-2381.2240	2081.4022
SR-BL	983.1346	1891.9654	0.5196	0.6051	-2794.2962	4760.5654
AVE PRICE BY CAP	0.4747	0.1899	2.4994	0.0149	0.0955	0.8539
const	-135270.5382	168087.8531	-0.8048	0.4238	-470868.7649	200327.6885
AGE	34136.7208	39321.9766	0.8681	0.3885	-44372.1386	112645.5801
ODI-SR-B	371.0501	845.7380	0.4387	0.6623	-1317.5204	2059.6206
ODI-SR-BL	86.1725	792.5428	0.1087	0.9137	-1496.1901	1668.5352
SR-B	-112.8751	719.1595	-0.1570	0.8758	-1548.7233	1322.9731
ECON	-797.3292	5077.5843	-0.1570	0.8757	-10935.0534	9340.3950
PREMIUM	1.0549	0.0623	16.9215	0.0000	0.9304	1.1793
COUNTRY_SA	-79038.8987	59154.2824	-1.3361	0.1861	-197144.2342	39066.4368
TEAM_CSK+	-66970.5205	99465.6887	-0.6733	0.5031	-265560.1754	131619.1345
TEAM_DC	-78176.0203	88668.4243	-0.8817	0.3812	-255208.2413	98856.2007
TEAM_DC+	-65700.3678	78101.1031	-0.8412	0.4033	-221634.2513	90233.5156
TEAM_DD	-2817.0213	100669.0695	-0.0280	0.9778	-203809.3035	198175.2608
TEAM_DD+	-73511.5556	88361.8381	-0.8319	0.4084	-249931.6574	102908.5463
TEAM_KKR	-44396.7075	91518.2692	-0.4851	0.6292	-227118.8274	138325.4123
TEAM_KKR+	-94968.0128	78384.8869	-1.2116	0.2300	-251468.4890	61532.4633
TEAM_KXI+	-7476.1221	179596.1248	-0.0416	0.9669	-366051.3545	351099.1103
TEAM_KXIP	-12089.9523	120504.0722	-0.1003	0.9204	-252684.0951	228504.1904
TEAM_KXIP+	-52273.3740	87359.5718	-0.5984	0.5516	-226692.3866	122145.6386
TEAM_MI	-93056.9764	95185.1854	-0.9776	0.3318	-283100.3307	96986.3780
TEAM_MI+	-119389.9828	140683.8222	-0.8486	0.3991	-400274.2965	161494.3310
TEAM_RCB	-172527.3593	84495.2575	-2.0419	0.0452	-341227.5839	-3827.1347
TEAM_RCB+	43065.5905	74110.4635	0.5811	0.5632	-104900.7240	191031.9050
TEAM_RR	-59475.8065	108068.9339	-0.5504	0.5839	-275242.3946	156290.7816
TEAM_RR+	-81968.7103	91344.9749	-0.8974	0.3728	-264344.8369	100407.4164

PLAYING ROLE_Bowler	4964.7157	61998.6486	0.0801	0.9364	-118819.5802	128749.0116
PLAYING ROLE_W. Keeper	-19479.7699	72087.7641	-0.2702	0.7878	-163407.6347	124448.0950

Omnibus:	122.027	Durbin-Watson:	2.047
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3168.743
Skew:	4.153	Prob(JB):	0.000
Kurtosis:	29.740	Condition No.:	9666126

In [105]:

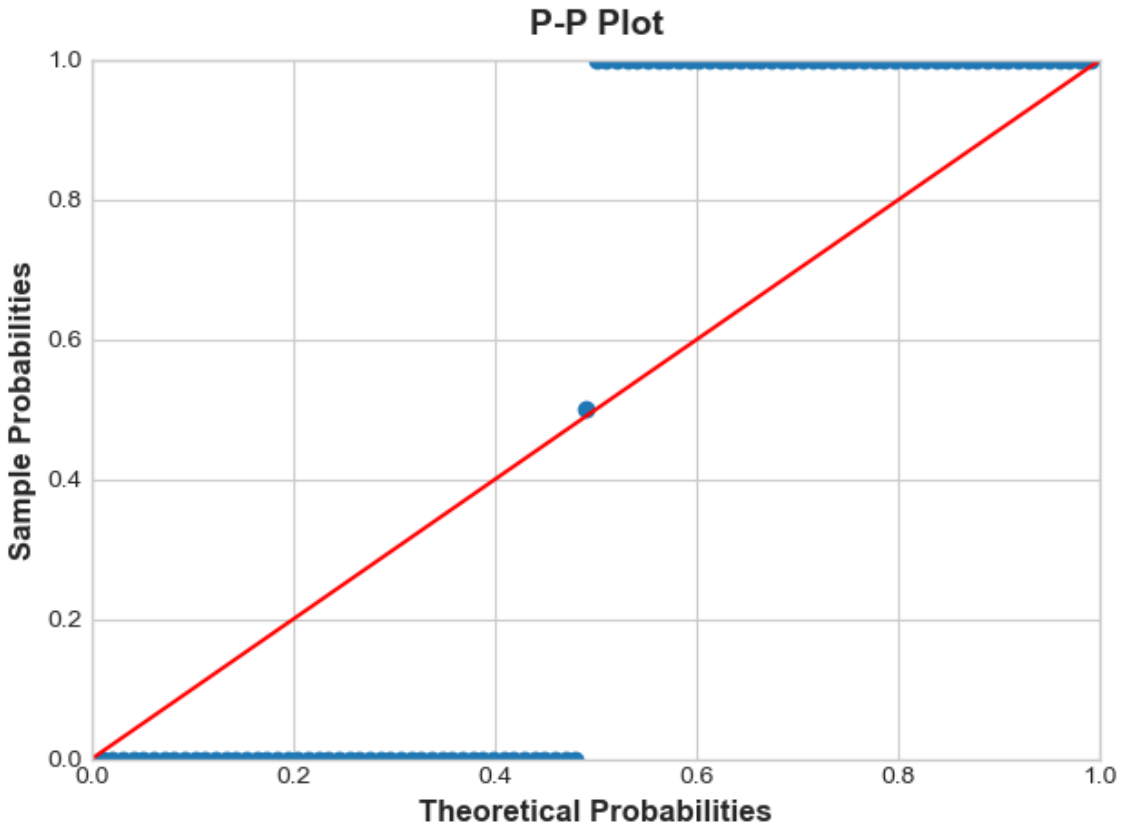
```
# validate on testing dataset

pred_y = sm_model2.predict(val_X[new_lis])

# scoring
def scorer(pred, val):
    RMSE = np.sqrt(mean_squared_error(pred_y,val_y))
    R2 = r2_score(pred_y,val_y)
    return RMSE, R2


print("RMSE and R2 are as follow: ", scorer(pred_y,val_y))
```

```
RMSE and R2 are as follow:  (87902.76366840841, 0.963711833236872)
```

In [106]:

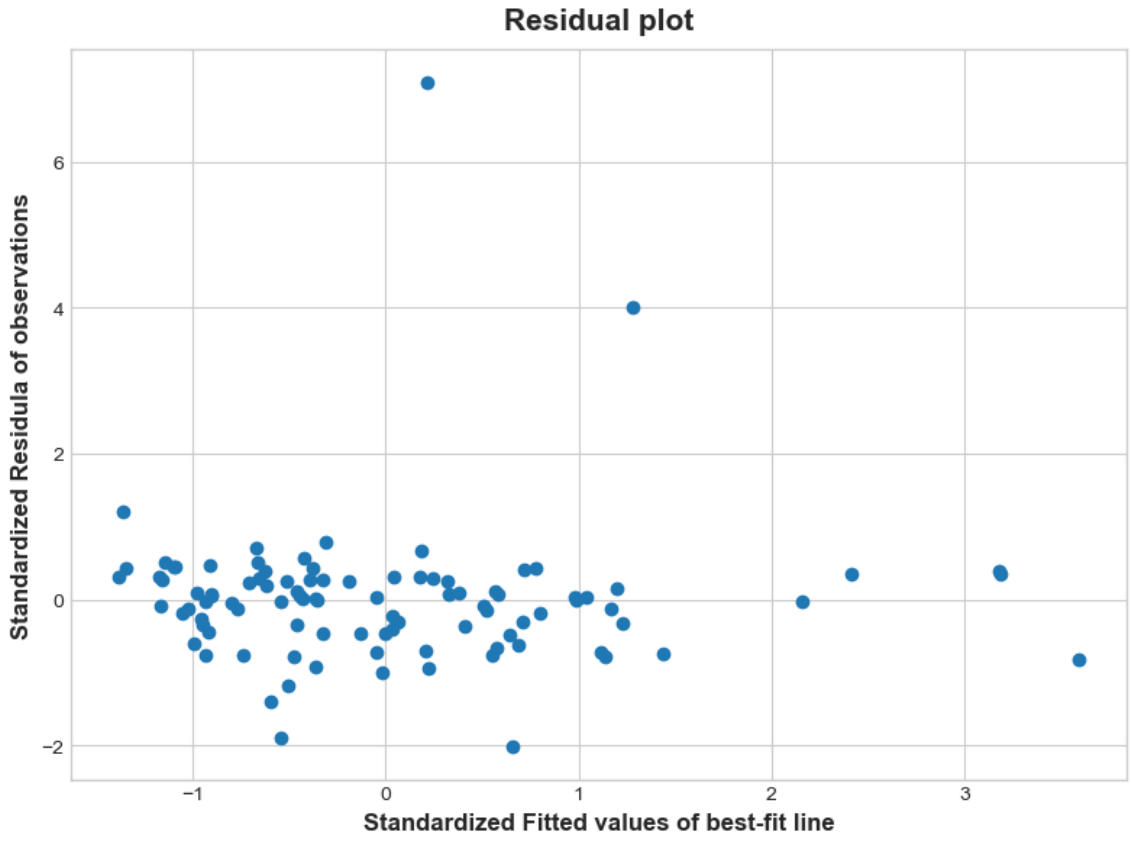
```
# P-P plot
def pp_plot(model, title):
    plt.figure(figsize=(8,6))
    probplot = sm.ProbPlot(model.resid)
    probplot.ppplot(line='45')
    plt.title(title)
    plt.show()
pp_plot(sm_model2, 'P-P Plot')
```

```
<Figure size 800x600 with 0 Axes>
```



In [107]:

```
def get_std_values(val):
    std_val = (val - val.mean())/val.std()
    return std_val

# Residule plot
def residual_plot(fitted, residual, title):
    plt.figure(figsize=(8,6))
    plt.scatter(get_std_values(fitted),
                get_std_values(residual))
    plt.title(title, fontsize=15)
    plt.xlabel("Standardized Fitted values of best-fit line")
    plt.ylabel("Standardized Residula of observations")
    plt.show()

residual_plot(sm_model2.fittedvalues, sm_model2.resid, 'Residual plot')
```



In [109]:

```
# Models dict
X_trn = train_X[new_lis].drop('const', axis=1)
X_tst = val_X[new_lis].drop('const', axis=1)
train_y, val_y

def model_function(mod,**kwargs):
    model = mod(**kwargs)
    model.fit(X_trn,train_y)
    pre_y = model.predict(X_tst)
    RMSE = round(np.sqrt(mean_squared_error(pre_y,val_y)), 3)
    R2 = round(r2_score(pre_y,val_y)*100, 3)
    return RMSE, R2

print("RMSE and R2 of Linear regression model are: ",
      model_function(LinearRegression, normalize=True))
```

```
RMSE and R2 of Linear regression model are:  (87902.764, 96.371)
```

In [110]:

```
#Ridge regression
print("RMSE and R2 of Ridge regression model are: ", model_function(Ridge, alpha=0.15))
```

```
RMSE and R2 of Ridge regression model are:  (84222.506, 96.655)
```

In [111]:

```
#SGD regression
print("RMSE and R2 of SGD regression model are: ", model_function(SGDRegressor, early_st
```

```
RMSE and R2 of SGD regression model are:  (9.105128821847534e+20, -14.178)
```

In [112]:

```
#Decision tree regression
print("RMSE and R2 of Decision tree regression model are: ",
      model_function(DecisionTreeRegressor, max_features='auto', random_state=0))
```

```
RMSE and R2 of Decision tree regression model are:  (397666.493, 37.026)
```

In [113]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [114]:

```
random=RandomForestRegressor().fit(X_trn,train_y)
```

In [115]:

```
yp=random.predict(X_tst)
```

In [116]:

```
RMSE = round(np.sqrt(mean_squared_error(yp,val_y)), 3)
R2 = round(r2_score(yp,val_y)*100, 3)
```

In [119]:

```
accuracy=accuracy_score(yp,val_y)
```

In [117]:

```
print(RMSE,R2)
```

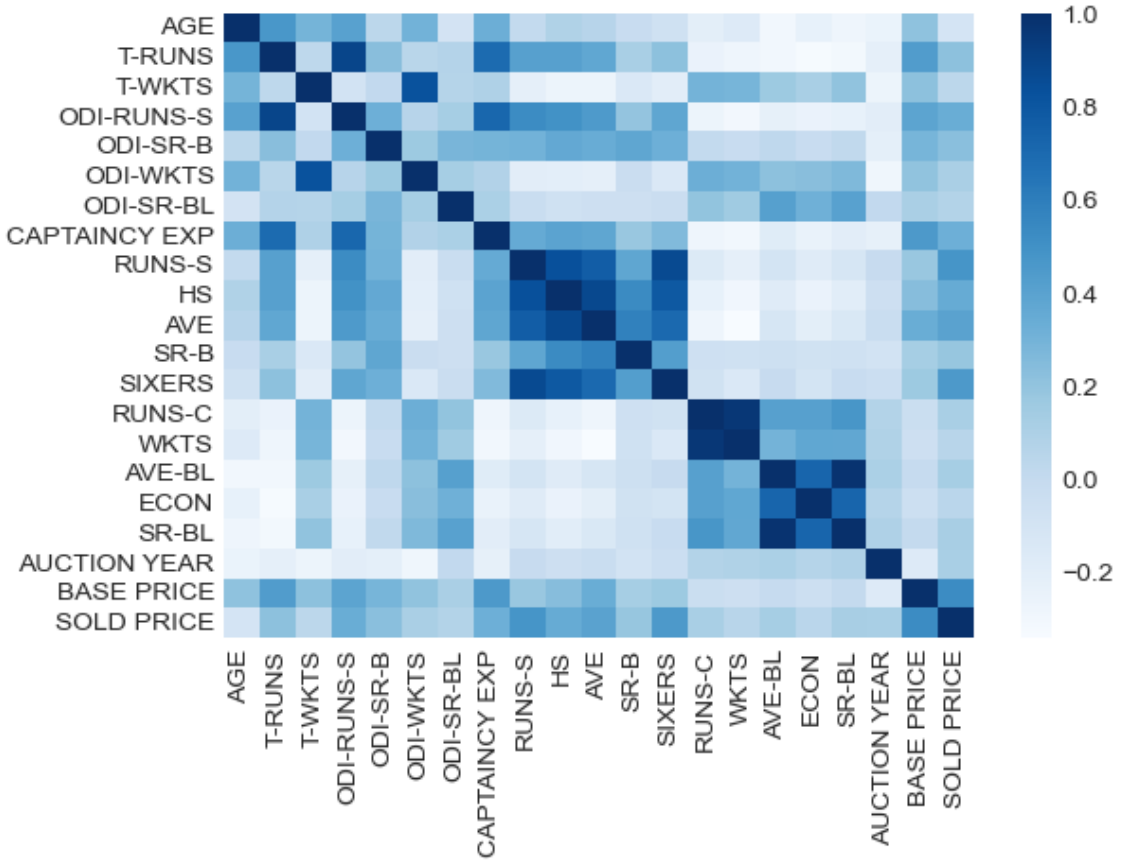
```
134766.813 90.152
```

In [118]:

```
sns.heatmap(df.corr(),cmap="Blues")
```

Out[118]:

```
<AxesSubplot:>
```



In [128]:

```
m=['Ridge','RandomForest','SGDRegressor','LinearRegression','DecisionTreeRegressor']
v=[96.655,90.152,-14.178,96.371,37.026]
r=[84222.506,134766.813,9.105128821847534e+20,87902.764,397666.493]
models=pd.DataFrame([m,v,r]).transpose()
```

In [130]:

```
models.rename(columns={0:'model',1:'r2_score',2:'RMSE'},inplace=True)
```

In [131]:

```
models
```

Out[131]:

	model	r2_score	RMSE
0	Ridge	96.655	84222.506
1	RandomForest	90.152	134766.813
2	SGDRegressor	-14.178	910512882184753381376.0
3	LinearRegression	96.371	87902.764
4	DecisionTreeRegressor	37.026	397666.493

In [ ]:

```

```