In [1]:
```
import pandas as pd
```

In [2]:
```
pd.read_csv?
```

In [2]:
```
ipl_df = pd.read_csv('IPL IMB381IPL2013.csv')
```

In [3]:
```
type(ipl_df)#to identify what type is the variable
```

Out[3]:
```
pandas.core.frame.DataFrame
```

In [4]:
```
pd.set_option('display.max_columns', 7)#setting the max columns to be viewed
```

In [5]:
```
ipl_df.head(5)# checking first 5 rows of the dataframe
```

Out[5]:

	Sl.NO.	PLAYER NAME	AGE	...	AUCTION YEAR	BASE PRICE	SOLD PRICE
0	1	Abdulla, YA	2	...	2009	50000	50000
1	2	Abdur Razzak	2	...	2008	50000	50000
2	3	Agarkar, AB	2	...	2008	200000	350000
3	4	Ashwin, R	1	...	2011	100000	850000
4	5	Badrinath, S	2	...	2011	100000	800000

5 rows × 26 columns

In [6]:
```
list(ipl_df.columns)#identifying the column names with the list function
```

Out[6]:
```
['Sl.NO.',
 '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']
```

In [7]:
```
ipl_df.head(5).transpose()
```

Out[7]:

	0	1	2	3	4
Sl.NO.	1	2	3	4	5

PLAYER NAME	Abdulla, YA	Abdur Razzak	Agarkar, AB	Ashwin, R	Badrinath, S
AGE	2	2	2	1	2
COUNTRY	SA	BAN	IND	IND	IND
TEAM	KXIP	RCB	KKR	CSK	CSK
PLAYING ROLE	Allrounder	Bowler	Bowler	Bowler	Batsman
T-RUNS	0	214	571	284	63
T-WKTS	0	18	58	31	0
ODI-RUNS-S	0	657	1269	241	79
ODI-SR-B	0.0	71.41	80.62	84.56	45.93
ODI-WKTS	0	185	288	51	0
ODI-SR-BL	0.0	37.6	32.9	36.8	0.0
CAPTAINCY EXP	0	0	0	0	0
RUNS-S	0	0	167	58	1317
HS	0	0	39	11	71
AVE	0.0	0.0	18.56	5.8	32.93
SR-B	0.0	0.0	121.01	76.32	120.71
SIXERS	0	0	5	0	28
RUNS-C	307	29	1059	1125	0
WKTS	15	0	29	49	0
AVE-BL	20.47	0.0	36.52	22.96	0.0
ECON	8.9	14.5	8.81	6.23	0.0
SR-BL	13.93	0.0	24.9	22.14	0.0
AUCTION YEAR	2009	2008	2008	2011	2011
BASE PRICE	50000	50000	200000	100000	100000
SOLD PRICE	50000	50000	350000	850000	800000

In [8]:
```
ipl_df.shape#checking the number of rows and columns. Shows 130 rows and 26 columns
```

Out[8]:
```
(130, 26)
```

In [9]:
```
ipl_df.info()#checking the summary of the dataframe. Here 130 against each column shows
#missing values. object 4 means that 4 categorical variables are there
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130 entries, 0 to 129
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Sl.NO.          130 non-null    int64 
 1   PLAYER NAME     130 non-null    object
 2   AGE             130 non-null    int64 
 3   COUNTRY         130 non-null    object
 4   TEAM            130 non-null    object
 5   PLAYING ROLE    130 non-null    object
 6   T-RUNS          130 non-null    int64 
 7   T-WKTS          130 non-null    int64 
 8   ODI-RUNS-S      130 non-null    int64 
```

```
 9   ODI-SR-B        130 non-null    float64
 10  ODI-WKTS        130 non-null    int64
 11  ODI-SR-BL       130 non-null    float64
 12  CAPTAINCY EXP   130 non-null    int64
 13  RUNS-S          130 non-null    int64
 14  HS              130 non-null    int64
 15  AVE             130 non-null    float64
 16  SR-B            130 non-null    float64
 17  SIXERS          130 non-null    int64
 18  RUNS-C          130 non-null    int64
 19  WKTS            130 non-null    int64
 20  AVE-BL          130 non-null    float64
 21  ECON            130 non-null    float64
 22  SR-BL           130 non-null    float64
 23  AUCTION YEAR    130 non-null    int64
 24  BASE PRICE      130 non-null    int64
 25  SOLD PRICE      130 non-null    int64
dtypes: float64(7), int64(15), object(4)
memory usage: 26.5+ KB
```

In [12]:
```
ipl_df[0:5]#shows only first 5 rows of the dataframe
```

Out[12]:

	SI.NO.	PLAYER NAME	AGE	...	AUCTION YEAR	BASE PRICE	SOLD PRICE
0	1	Abdulla, YA	2	...	2009	50000	50000
1	2	Abdur Razzak	2	...	2008	50000	50000
2	3	Agarkar, AB	2	...	2008	200000	350000
3	4	Ashwin, R	1	...	2011	100000	850000
4	5	Badrinath, S	2	...	2011	100000	800000

5 rows × 26 columns

In [13]:
```
ipl_df[-5:]#shows last 5 rows of the dataframe
```

Out[13]:

	SI.NO.	PLAYER NAME	AGE	...	AUCTION YEAR	BASE PRICE	SOLD PRICE
125	126	Yadav, AS	2	...	2010	50000	750000
126	127	Younis Khan	2	...	2008	225000	225000
127	128	Yuvraj Singh	2	...	2011	400000	1800000
128	129	Zaheer Khan	2	...	2008	200000	450000
129	130	Zoysa, DNT	2	...	2008	100000	110000

5 rows × 26 columns

In [14]:
```
ipl_df['PLAYER NAME'][0:5]#displaying only the player name of first 5 rows of the datafr
```

Out[14]:
```
0     Abdulla, YA
1    Abdur Razzak
2     Agarkar, AB
3       Ashwin, R
4    Badrinath, S
Name: PLAYER NAME, dtype: object
```

In [15]:
```
ipl_df[['PLAYER NAME', 'COUNTRY']][0:5]#viewing two columns player name and country with
```

Out[15]:

	PLAYER NAME	COUNTRY
0	Abdulla, YA	SA

1	Abdur Razzak	BAN
2	Agarkar, AB	IND
3	Ashwin, R	IND
4	Badrinath, S	IND

In [16]:
```
ipl_df.iloc[4:9,1:4]#using iloc function to view row indexed 4 to 9 but not 9 and column
```

Out[16]:

	PLAYER NAME	AGE	COUNTRY
4	Badrinath, S	2	IND
5	Bailey, GJ	2	AUS
6	Balaji, L	2	IND
7	Bollinger, DE	2	AUS
8	Botha, J	2	SA

In [17]:
```
ipl_df.COUNTRY.value_counts()#gives the list of players from each country. value counts
#categorical variable
```

Out[17]:
```
IND    53
AUS    22
SA     16
SL     12
PAK     9
NZ      7
WI      6
ENG     3
BAN     1
ZIM     1
Name: COUNTRY, dtype: int64
```

In [18]:
```
ipl_df.COUNTRY.value_counts(normalize= True)#gives the percentage of players of each cou
```

Out[18]:
```
IND    0.407692
AUS    0.169231
SA     0.123077
SL     0.092308
PAK    0.069231
NZ     0.053846
WI     0.046154
ENG    0.023077
BAN    0.007692
ZIM    0.007692
Name: COUNTRY, dtype: float64
```

In [19]:
```
pd.crosstab(ipl_df['AGE'], ipl_df['PLAYING ROLE'])#creating a crosstab of playing role a
```

Out[19]:

PLAYING ROLE	Allrounder	Batsman	Bowler	W. Keeper
AGE				
1	4	5	7	0
2	25	21	29	11
3	6	13	8	1

In [20]:
```
ipl_df[['PLAYER NAME', 'SOLD PRICE']].sort_values('SOLD PRICE')[0:5]#sorting players pri
#to descending and printing only 5 records
```

Out[20]:

	PLAYER NAME	SOLD PRICE
73	Noffke, AA	20000
46	Kamran Khan	24000
0	Abdulla, YA	50000
1	Abdur Razzak	50000
118	Van der Merwe	50000

In [21]:
```
ipl_df[['PLAYER NAME', 'SOLD PRICE']].sort_values('SOLD PRICE', ascending= False)[0:5]#s
#in descending orders and printing 5 records
```

Out[21]:

	PLAYER NAME	SOLD PRICE
93	Sehwag, V	1800000
127	Yuvraj Singh	1800000
50	Kohli, V	1800000
111	Tendulkar, SR	1800000
113	Tiwary, SS	1600000

In [22]:
```
ipl_df['premium'] = ipl_df['SOLD PRICE'] - ipl_df['BASE PRICE']#creating a new column
```

In [23]:
```
ipl_df[['PLAYER NAME', 'SOLD PRICE', 'BASE PRICE', 'premium']][0:5]
```

Out[23]:

	PLAYER NAME	SOLD PRICE	BASE PRICE	premium
0	Abdulla, YA	50000	50000	0
1	Abdur Razzak	50000	50000	0
2	Agarkar, AB	350000	200000	150000
3	Ashwin, R	850000	100000	750000
4	Badrinath, S	800000	100000	700000

In [24]:
```
ipl_df[['PLAYER NAME', 'SOLD PRICE', 'BASE PRICE', 'premium']].sort_values('premium', as
#sorting players premium wise in descending order and printing top 5 players with premiu
```

Out[24]:

	PLAYER NAME	SOLD PRICE	BASE PRICE	premium
50	Kohli, V	1800000	150000	1650000
113	Tiwary, SS	1600000	100000	1500000
127	Yuvraj Singh	1800000	400000	1400000
111	Tendulkar, SR	1800000	400000	1400000
93	Sehwag, V	1800000	400000	1400000

In [19]:
```
ipl_df.groupby('AGE')['SOLD PRICE'].mean()#grouping the players age wise and finding the
#each age category
```

Out[19]:
```
AGE
1    720250.000000
2    484534.883721
3    520178.571429
Name: SOLD PRICE, dtype: float64
```

In [26]:
```
sold_price_agewise = ipl_df.groupby('AGE')['SOLD PRICE'].mean().reset_index()#creating a
#selling price age wise
print(sold_price_agewise)
```

```
   AGE     SOLD PRICE
0    1  720250.000000
1    2  484534.883721
2    3  520178.571429
```

In [27]:
```
sold_price_roleagewise = ipl_df.groupby(['AGE', 'PLAYING ROLE'])['SOLD PRICE'].mean().re
#selling price age wise and role wise
print(sold_price_roleagewise)
```

```
    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 [28]:
```
soldprice_comparison = sold_price_roleagewise.merge(sold_price_agewise, on = 'AGE', how
soldprice_comparison
```

Out[28]:

	AGE	PLAYING ROLE	SOLD PRICE_x	SOLD PRICE_y
0	1	Allrounder	5.875000e+05	720250.000000
1	1	Batsman	1.110000e+06	720250.000000
2	1	Bowler	5.177143e+05	720250.000000
3	2	Allrounder	4.494000e+05	484534.883721
4	2	Batsman	6.547619e+05	484534.883721
5	2	Bowler	3.979310e+05	484534.883721
6	2	W. Keeper	4.677273e+05	484534.883721
7	3	Allrounder	7.666667e+05	520178.571429
8	3	Batsman	4.576923e+05	520178.571429
9	3	Bowler	4.143750e+05	520178.571429
10	3	W. Keeper	7.000000e+05	520178.571429

In [29]:
```
soldprice_comparison.rename(columns={'SOLD PRICE_x': 'soldprice_age_role', 'SOLD PRICE_y
                            inplace= True)#renaming the columns
soldprice_comparison.head(5)
```

Out[29]:

	AGE	PLAYING ROLE	soldprice_age_role	soldprice_age
0	1	Allrounder	5.875000e+05	720250.000000
1	1	Batsman	1.110000e+06	720250.000000
2	1	Bowler	5.177143e+05	720250.000000
3	2	Allrounder	4.494000e+05	484534.883721
4	2	Batsman	6.547619e+05	484534.883721

In [30]:
```
soldprice_comparison['change'] = soldprice_comparison.apply(lambda rec:(rec.soldprice_ag
                                 rec.soldprice_age) / rec.soldprice_age, axis = 1)
soldprice_comparison
```

Out[30]:

	AGE	PLAYING ROLE	soldprice_age_role	soldprice_age	change
0	1	Allrounder	5.875000e+05	720250.000000	-0.184311
1	1	Batsman	1.110000e+06	720250.000000	0.541132
2	1	Bowler	5.177143e+05	720250.000000	-0.281202
3	2	Allrounder	4.494000e+05	484534.883721	-0.072513
4	2	Batsman	6.547619e+05	484534.883721	0.351320
5	2	Bowler	3.979310e+05	484534.883721	-0.178736
6	2	W. Keeper	4.677273e+05	484534.883721	-0.034688
7	3	Allrounder	7.666667e+05	520178.571429	0.473853
8	3	Batsman	4.576923e+05	520178.571429	-0.120125
9	3	Bowler	4.143750e+05	520178.571429	-0.203399
10	3	W. Keeper	7.000000e+05	520178.571429	0.345692

In [21]:
```
ipl_df[ipl_df['SIXERS'] > 80][['PLAYER NAME', 'SIXERS']]#filtering out players who hit s
```

Out[21]:

	PLAYER NAME	SIXERS
26	Gayle, CH	129
28	Gilchrist, AC	86
82	Pathan, YK	81
88	Raina, SK	97
97	Sharma, RG	82

In [54]:
```
ipl_df.drop('Sl.NO.', inplace= True, axis = 1)#dropping column sl no and replacing the o
#inplace set as true
```

In [32]:
```
ipl_df.columns
```

Out[32]:
```
Index(['Sl.NO.', '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', 'premium'],
      dtype='object')
```

In [33]:
```
autos = pd.read_csv('auto-mpg.data', sep= '\s+', header= None)
autos.head(5)
```

Out[33]:

	0	1	2	...	6	7	8
0	18.0	8	307.0	...	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	...	70	1	buick skylark 320
2	18.0	8	318.0	...	70	1	plymouth satellite
3	16.0	8	304.0	...	70	1	amc rebel sst
4	17.0	8	302.0	...	70	1	ford torino

5 rows × 9 columns

In [34]:
```
autos.columns = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'accelerati
                 'year', 'origin', 'name']#assigning names to the columns
autos.head(5)
```

Out[34]:

	mpg	cylinders	displacement	...	year	origin	name
0	18.0	8	307.0	...	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	...	70	1	buick skylark 320
2	18.0	8	318.0	...	70	1	plymouth satellite
3	16.0	8	304.0	...	70	1	amc rebel sst
4	17.0	8	302.0	...	70	1	ford torino

5 rows × 9 columns

In [35]:
```
autos.info()#checking the summary of the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    float64
 5   acceleration  398 non-null    float64
 6   year          398 non-null    int64  
 7   origin        398 non-null    int64  
 8   name          398 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

In [36]:
```
autos['horsepower'] = pd.to_numeric(autos['horsepower'], errors= 'coerce')#converting ho
#with removal of errors
autos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    float64
 5   acceleration  398 non-null    float64
 6   year          398 non-null    int64  
 7   origin        398 non-null    int64  
 8   name          398 non-null    object 
dtypes: float64(5), int64(3), object(1)
memory usage: 28.1+ KB
```

In [37]:
```
autos[autos.horsepower.isnull()]#checking null values in horsepower column
```

Out[37]:

	mpg	cylinders	displacement	...	year	origin	name
32	25.0	4	98.0	...	71	1	ford pinto

126	21.0	6	200.0	...	74	1	ford maverick
330	40.9	4	85.0	...	80	2	renault lecar deluxe
336	23.6	4	140.0	...	80	1	ford mustang cobra
354	34.5	4	100.0	...	81	2	renault 18i
374	23.0	4	151.0	...	82	1	amc concord dl

6 rows × 9 columns

In [38]:
```
autos = autos.dropna(subset= ['horsepower'])#dropping null values in horsepower data
```

In [39]:
```
autos[autos.horsepower.isnull()]
```

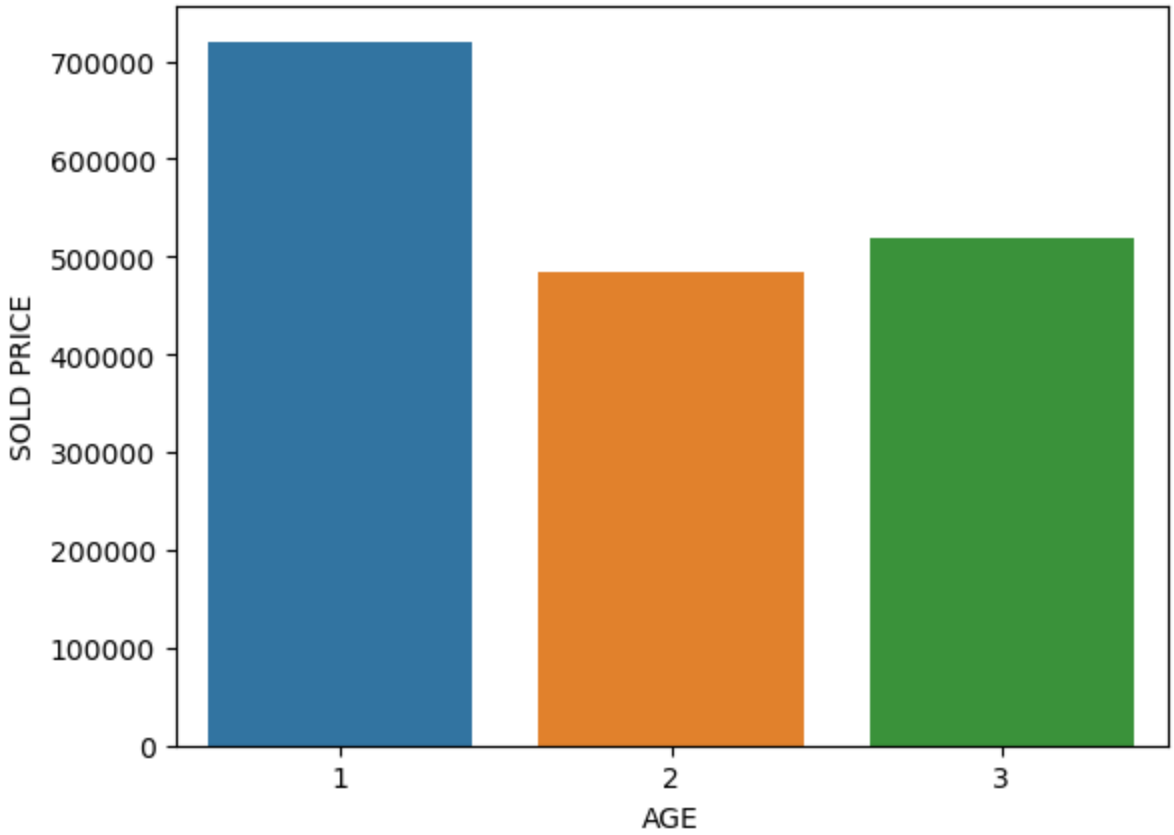
Out[39]:

	mpg	cylinders	displacement	...	year	origin	name

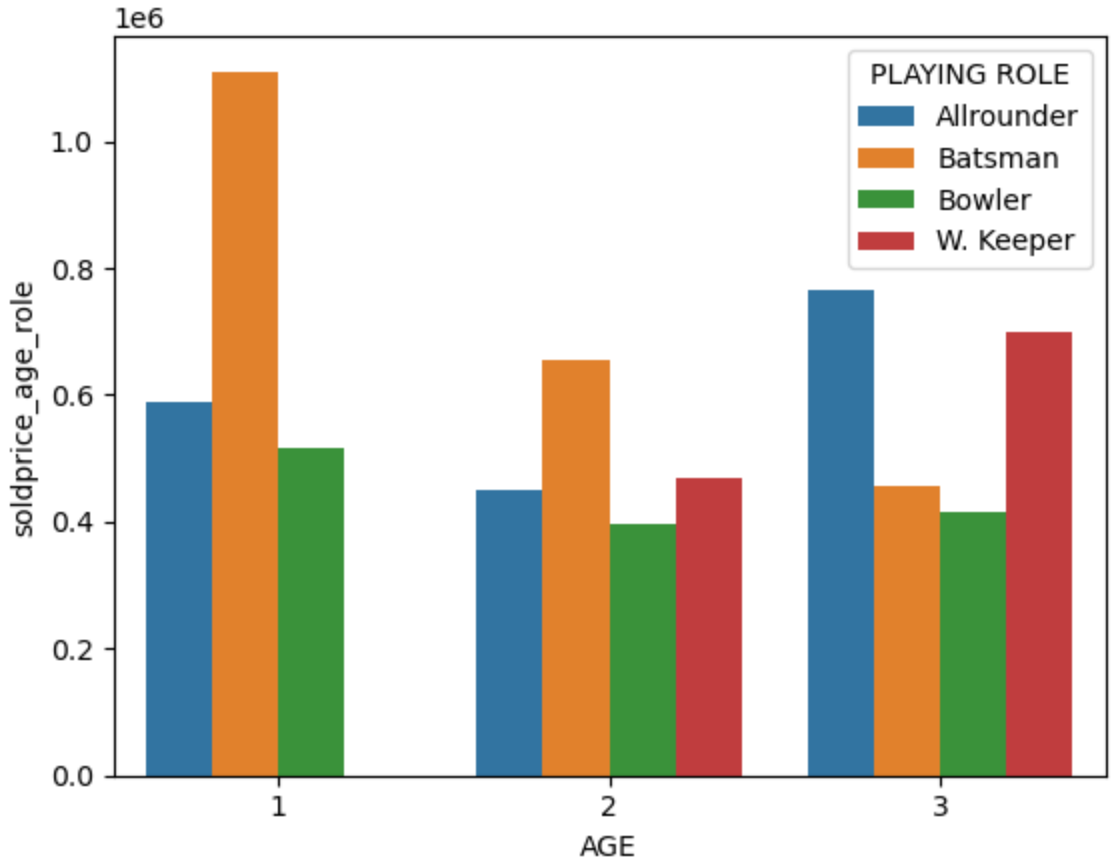
0 rows × 9 columns

In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

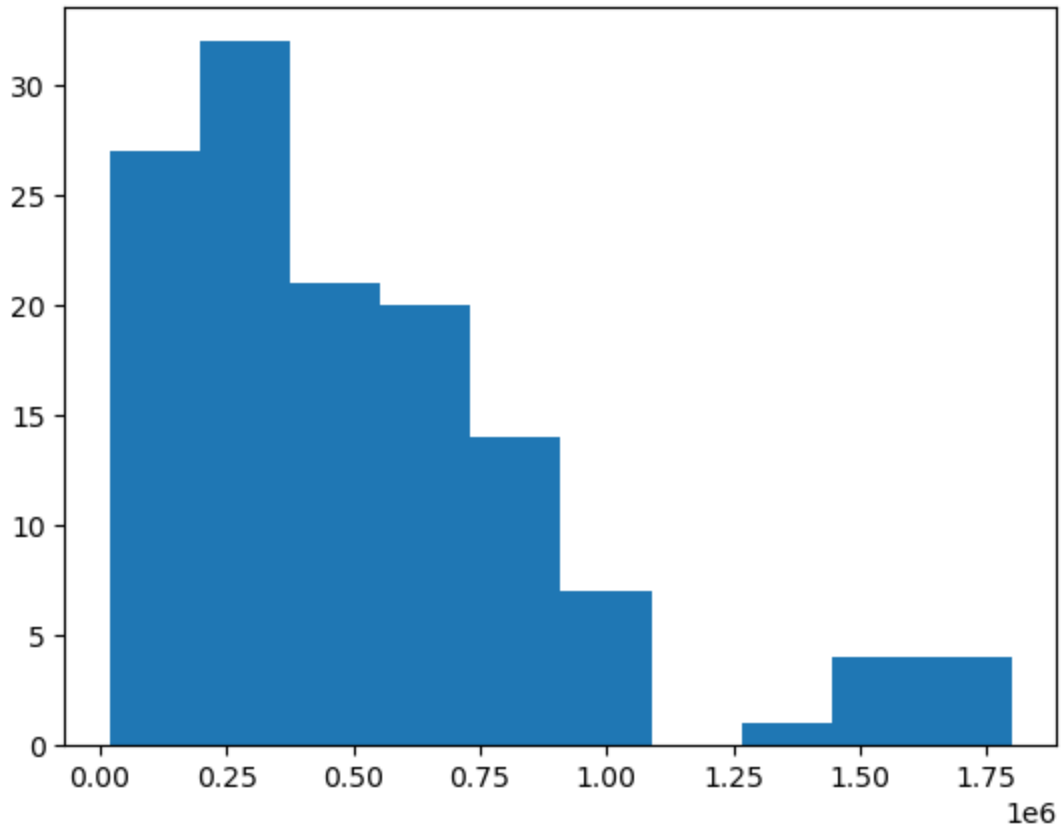
In [41]:
```
sn.barplot(x = 'AGE', y = 'SOLD PRICE', data= sold_price_agewise);
```



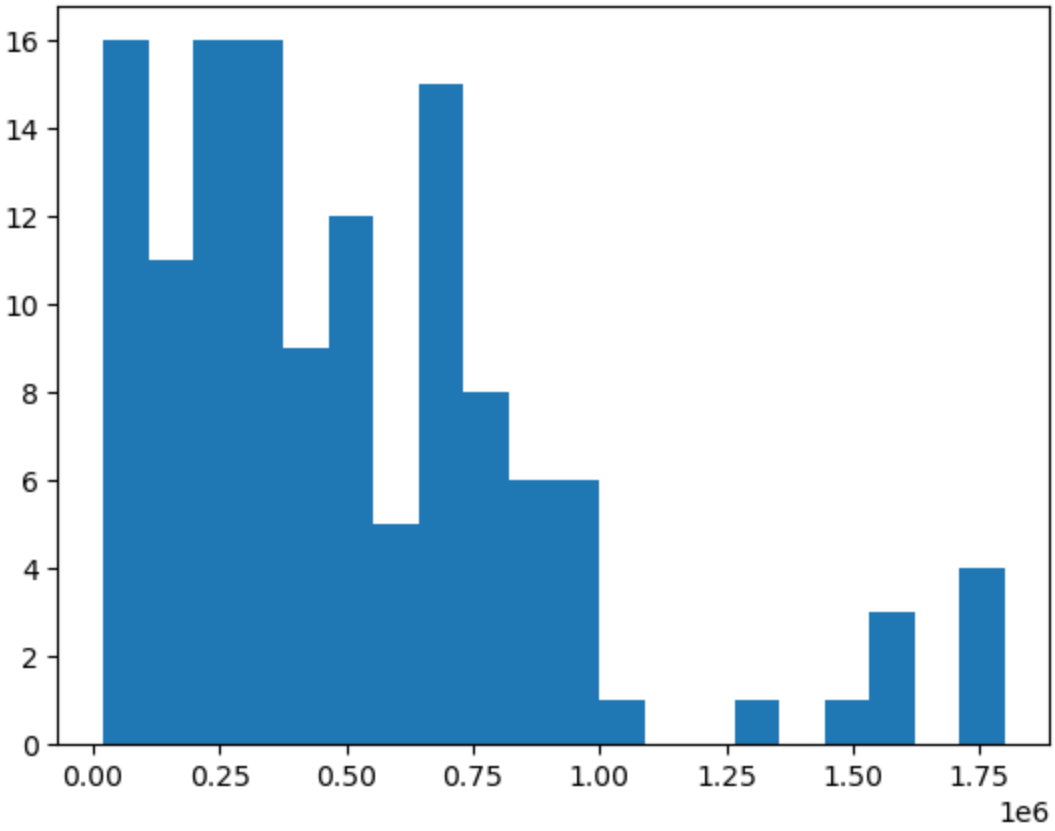
In [42]:
```
sn.barplot(x = 'AGE', y = 'soldprice_age_role', hue= 'PLAYING ROLE', data= soldprice_com
```



In [43]:
```
plt.hist(ipl_df['SOLD PRICE']);#shows that the most of the players are sold at a lesser
```

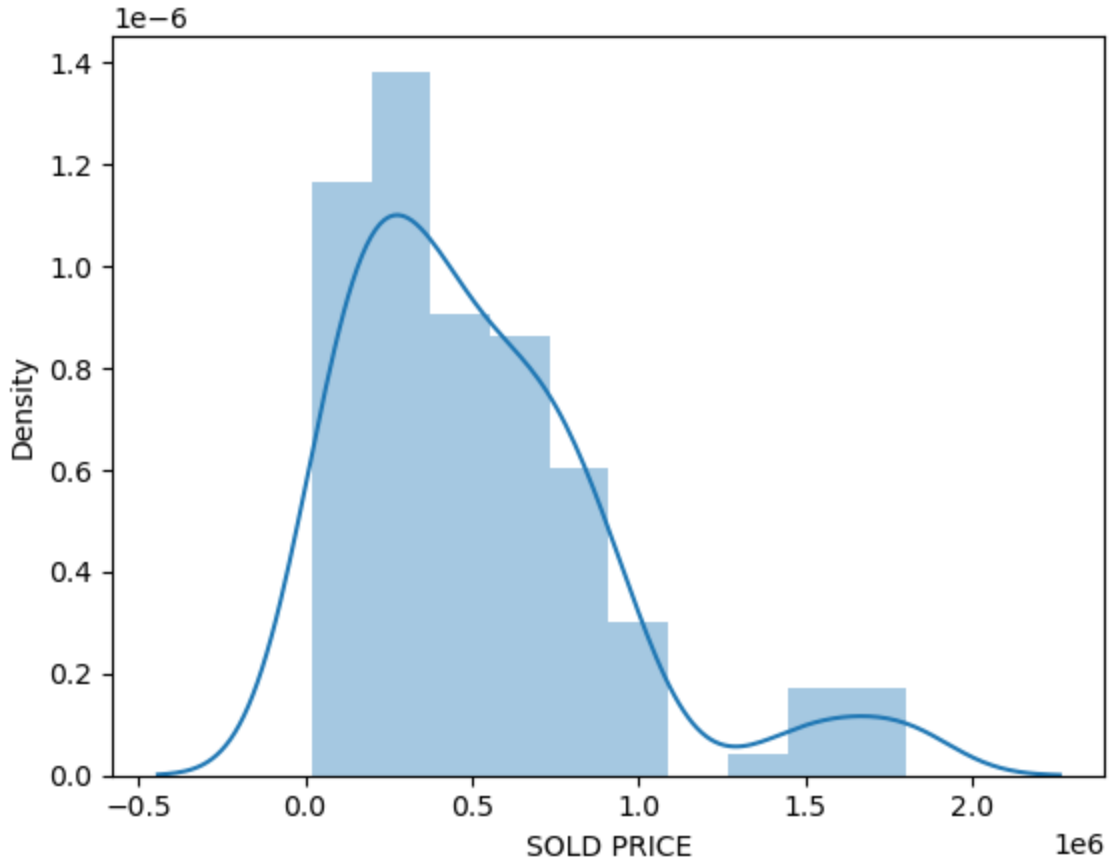


In [44]:
```
plt.hist(ipl_df['SOLD PRICE'], bins= 20);#by default histogram creates 10 bins. we incre
```



In [45]:
```
sn.distplot(ipl_df['SOLD PRICE']);
```

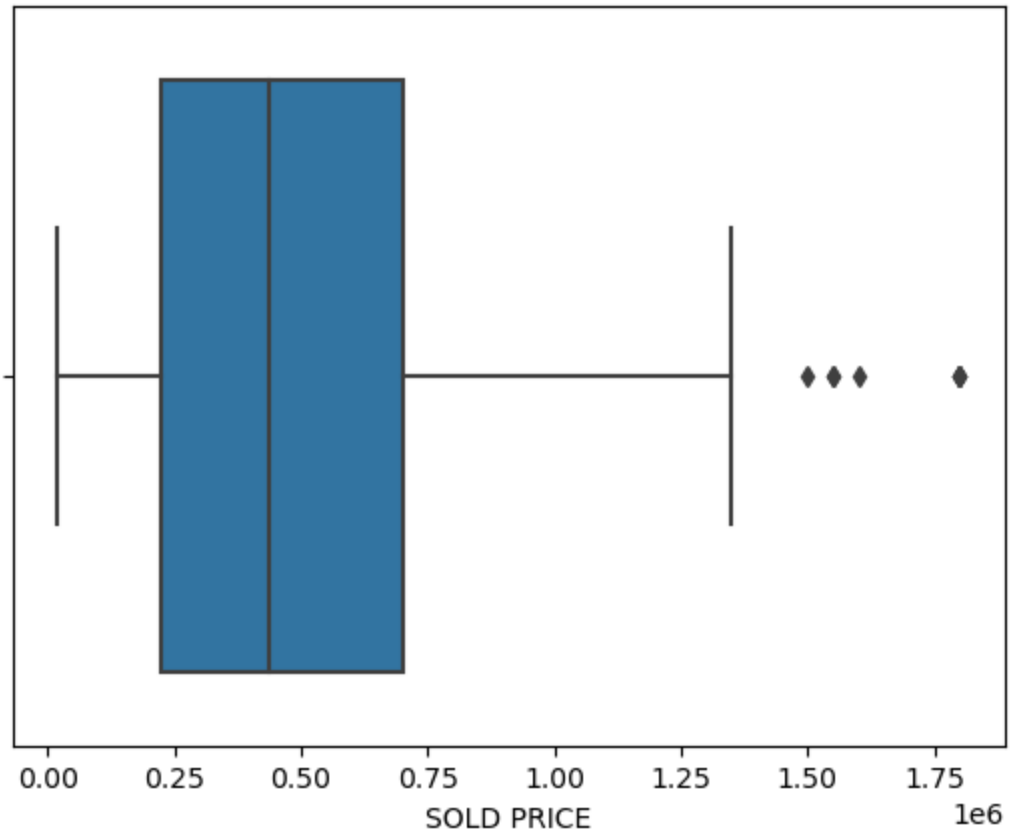
```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



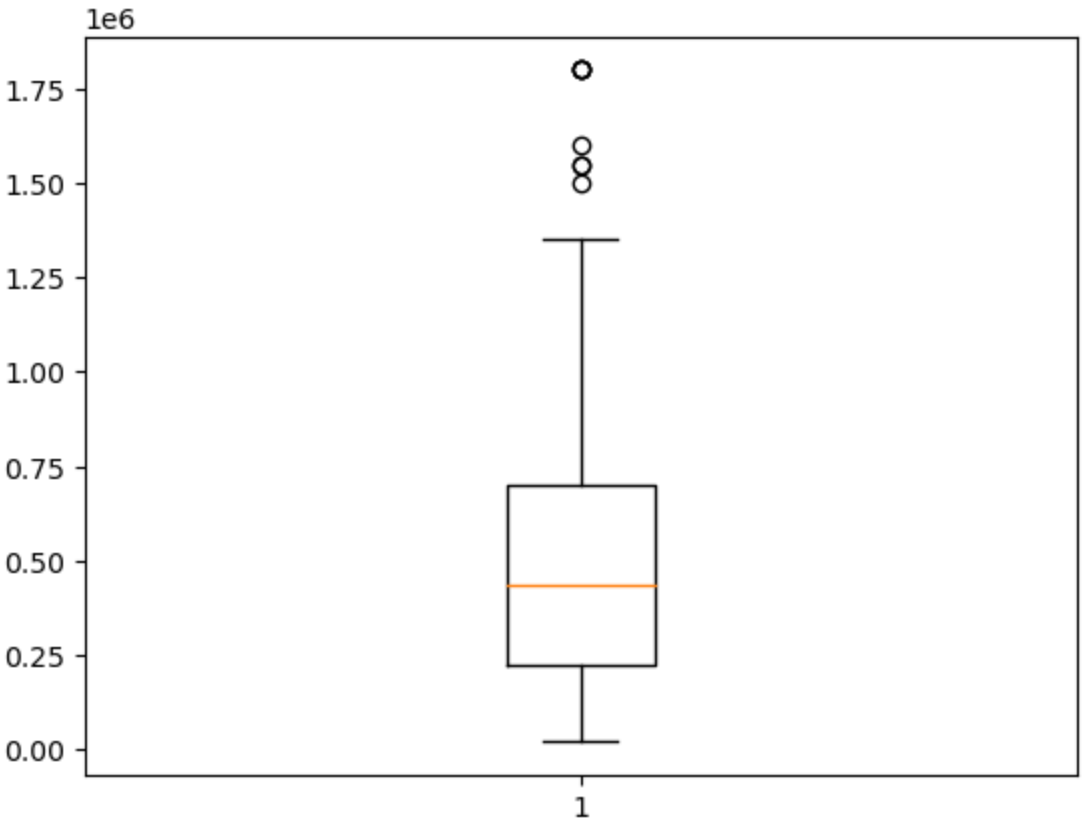
In [46]:
```
box = sn.boxplot(ipl_df['SOLD PRICE']);#shows the observations which are potential outli
```

```
#and Q3 which are the whiskers or the threads of the box plot
```

```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [27]:
```
box = plt.boxplot(ipl_df['SOLD PRICE']);
```



In [48]:
```
[item.get_ydata()[0] for item in box['caps']]#shows min and max offered price in auction
```

Out[48]:
```
[20000, 1350000]
```

In [50]:
```
[item.get_ydata()[0] for item in box['whiskers']]#shows the 25 and 75 quantiles of the a
```

Out[50]:
```
[225000.0, 700000.0]
```

In [51]:
```
[item.get_ydata()[0] for item in box['medians']]#shows the median auction price
```

Out[51]:
```
[437500.0]
```

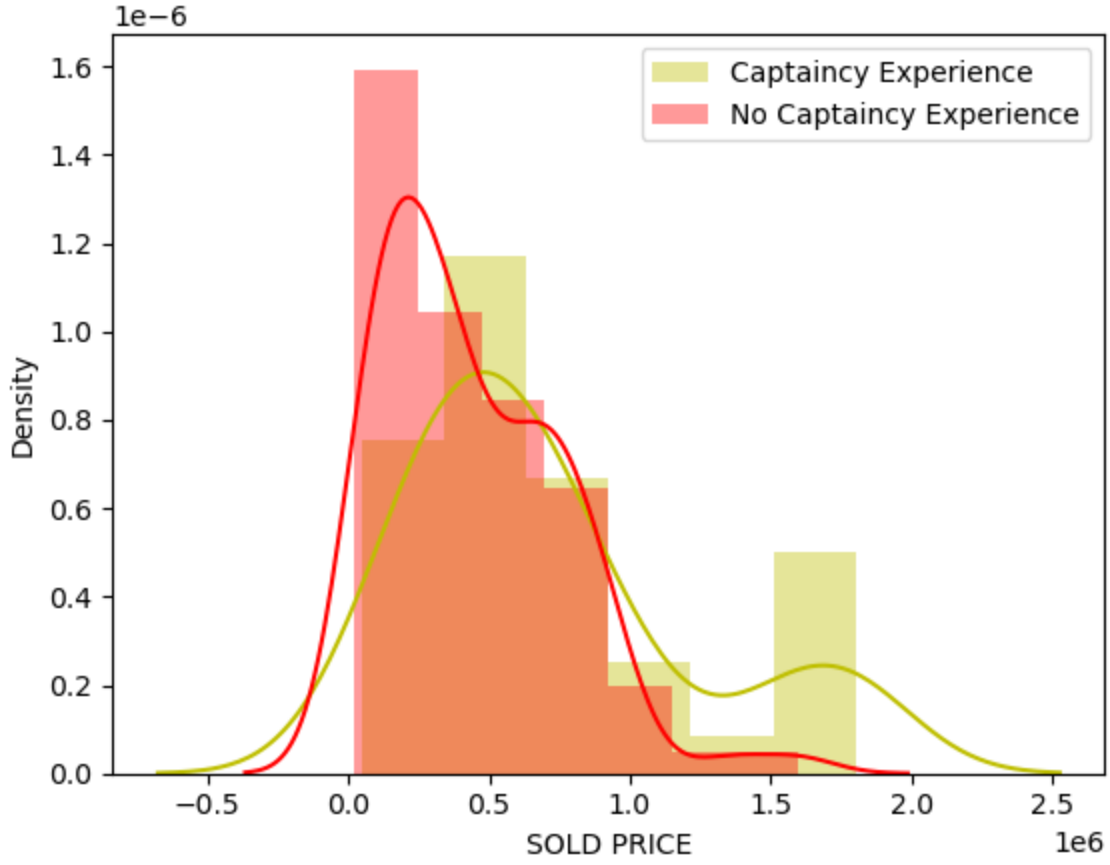
In [54]:
```
ipl_df[ipl_df['SOLD PRICE'] > 1350000][['PLAYER NAME', 'PLAYING ROLE', 'SOLD PRICE']]#ta
#that is players sold at a price greater than 13.50L
```

Out[54]:

	PLAYER NAME	PLAYING ROLE	SOLD PRICE
15	Dhoni, MS	W. Keeper	1500000
23	Flintoff, A	Allrounder	1550000
50	Kohli, V	Batsman	1800000
83	Pietersen, KP	Batsman	1550000
93	Sehwag, V	Batsman	1800000
111	Tendulkar, SR	Batsman	1800000
113	Tiwary, SS	Batsman	1600000
127	Yuvraj Singh	Batsman	1800000

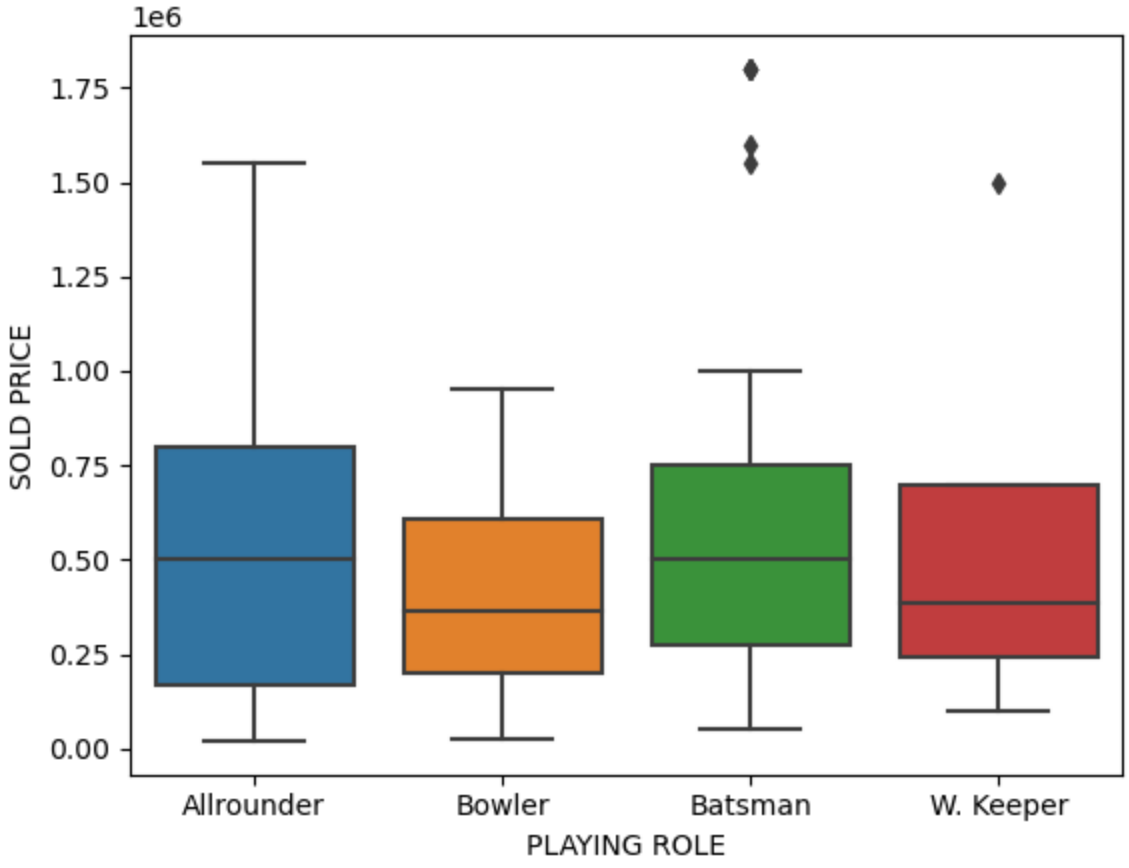
In [58]:
```
sn.distplot(ipl_df[ipl_df['CAPTAINCY EXP']==1]['SOLD PRICE'], color='y', label = 'Captai
sn.distplot(ipl_df[ipl_df['CAPTAINCY EXP']==0]['SOLD PRICE'], color='r', label = 'No Cap
plt.legend();#plotting the sold price of players with having captaincy experience and no
#having captaincy experience
```

```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:26
19: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:26
19: FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with sim
ilar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

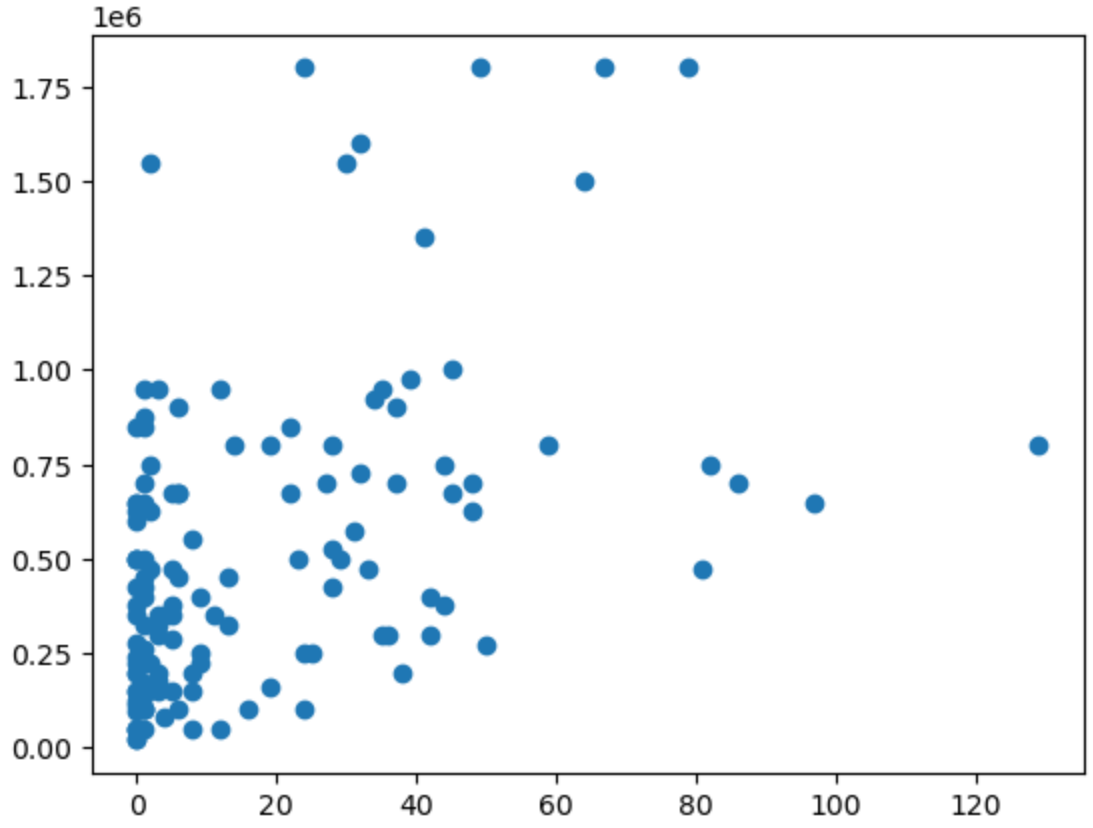


In [59]:
```
sn.boxplot(x= 'PLAYING ROLE', y= 'SOLD PRICE', data= ipl_df)#plotting the sold price rol
#The median SOLD PRICE for allrounders and batsmen are higher than bowlers and wicket ke
#Allrounders who are paid more than 1,35,0000 USD are not considered outliers. Allrounde
#have relatively high variance.
#There are outliers in batsman and wicket keeper category.
```

Out[59]:
```
<AxesSubplot:xlabel='PLAYING ROLE', ylabel='SOLD PRICE'>
```

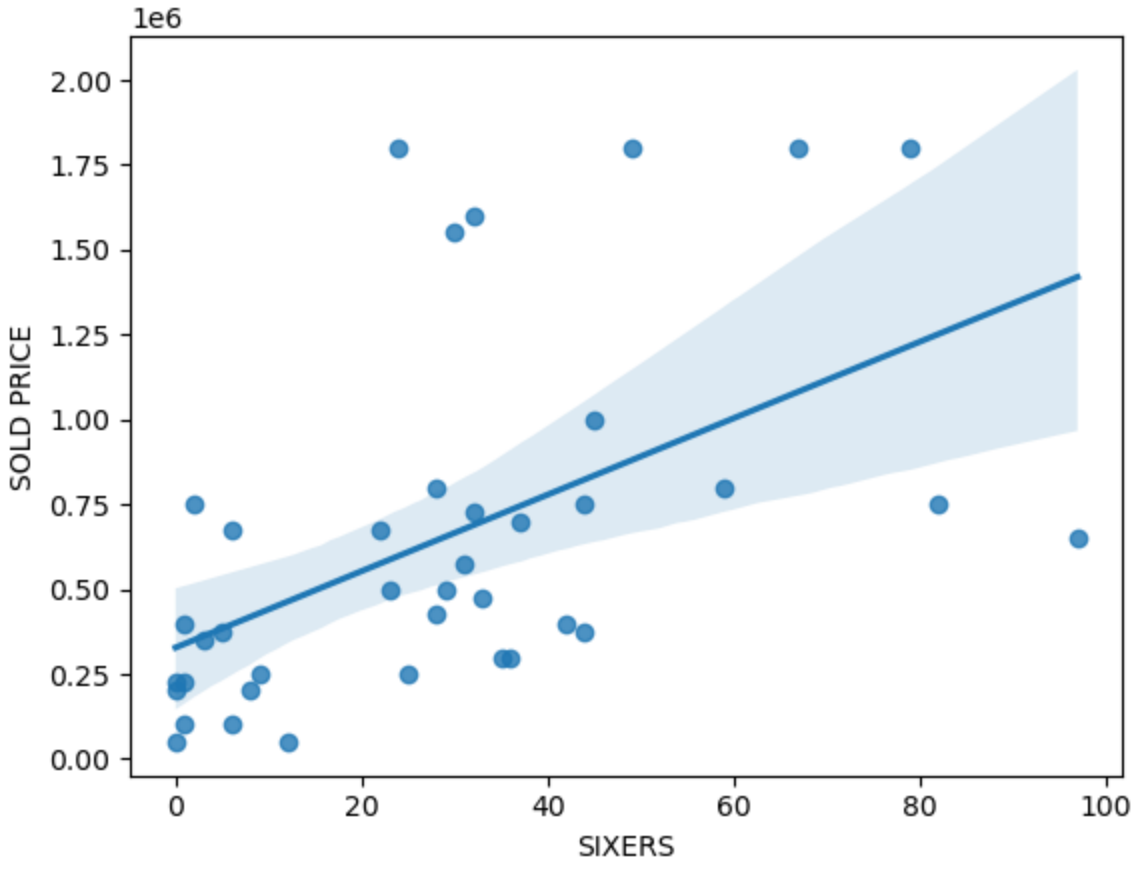


In [65]:
```
ipl_batsman = ipl_df[ipl_df['PLAYING ROLE']== 'Batsman']
plt.scatter(x= ipl_df['SIXERS'], y= ipl_df['SOLD PRICE']);#deriving the relation between
#the batsman vs the number of sixers he hit in the IPL season
```



In [66]:
```
sn.regplot(x= 'SIXERS', y= 'SOLD PRICE', data= ipl_batsman)#the line here shows the posi
#between sixers and the selling price of the batsman
```

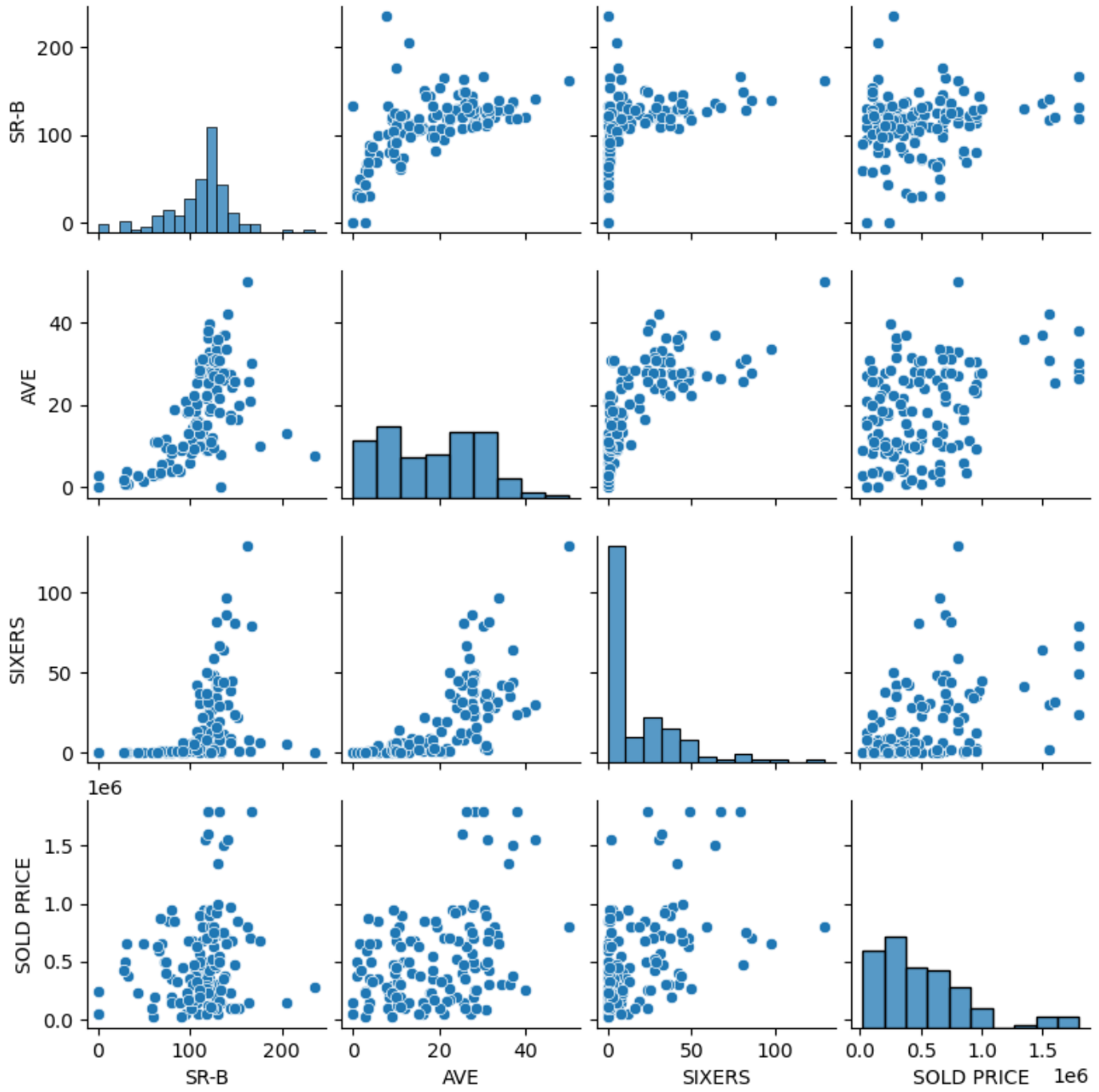
Out[66]:
```
<AxesSubplot:xlabel='SIXERS', ylabel='SOLD PRICE'>
```



In [67]:
```
features = ['SR-B', 'AVE', 'SIXERS', 'SOLD PRICE']
sn.pairplot(ipl_df[features], size= 2)#creating a pairplot to check the correlation betw
```

```
/Users/ishutejwani/opt/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:2076: U
serWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[67]:
```
<seaborn.axisgrid.PairGrid at 0x7f9943cd4880>
```



In [68]:
```
ipl_df[features].corr()#shows a correlation(-1 to +1) between 2 variables
```

Out[68]:

	SR-B	AVE	SIXERS	SOLD PRICE
SR-B	1.000000	0.583579	0.425394	0.184278
AVE	0.583579	1.000000	0.705365	0.396519
SIXERS	0.425394	0.705365	1.000000	0.450609
SOLD PRICE	0.184278	0.396519	0.450609	1.000000

In [69]:
```
sn.heatmap(ipl_df[features].corr(), annot= True)#shows the correlation via heatmap
```

```
#sr-b and sold price are not closely correlated whereas average and sixers have a positi
```

Out[69]: <AxesSubplot:>