In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv ("baseball.csv")
df.head(10)
```

Out[2]:

	W	R	AB	H	2B	3B	HR	BB	SO	SB	RA	ER	ERA	CG	SHO	SV	E
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86
5	93	891	5509	1480	308	17	232	570	1151	88	670	609	3.80	7	10	34	88
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97

In [ ]:
```
```

In [3]:
```
#  here we can see that all data provided in the dataset is numerical.
# the given column names are in shortform. the meaning of it is as follows :
#  because before solving an problem we have to understand the data first.
# "W" = No. of wins
# 'R' = Runs Scored
#'AB' = At Bat (batters's turn , battting against pitcher)
# 'H' = 'Base hit' (when the batter safely reaches to the FIRST BASE after hitting)
# '2B' = 2nd Base hit (when the bstter dafely reaches to the SECOND BASE after hitting)
# '3B' = 3rd Base hit ('THREE BAGGERS - when the batter safely reches to the third base after hitting')
# 'HR' = Home Run (complete circle , reaches home plate safely after touching all thre bases)
# 'BB' = Walk (awarded to turn to FIRST BASE, after receiving 4 balls, without out)
# 'SO' = 'K' 'Strike Out' (batter accumulate three strikes, i.e batter is out)
# 'SB' = Stolen Base (number of base advanced that should be credited to the action of the runner)
# 'RA' = Run Average (the rate at which runs are scored)
# 'ER' = Earned Run (No. of runs scored, without as a result of eroor)
# 'ERA' = Earned Run Average (average od earned run per 9 innings)
# 'CG' = Complete Game ( number of games where the player was the only pitcher for their game)
# 'SHO' = Shutout (no. of compelete game pitched with no runs allowed)
# 'SV' = Save (pitchers who fineshes a game for winning)
# 'E' = Error (a fielder missplaying a ball in a manner , that allow a batter to advance one or more bases)

#  so here above we descccribed all column meaning, according to this we are going forward and analyse more.
```

In [4]:
```
# first of all we are going to assign all fullforms inplace of shotcuts so it is going to be easy to understand.
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['W', 'R', 'AB', 'H', '2B', '3B', 'HR', 'BB', 'SO', 'SB', 'RA', 'ER',
       'ERA', 'CG', 'SHO', 'SV', 'E'],
      dtype='object')
```

In [6]:
```
df.columns.nunique()
#  here it is cleared that we have to assign total 17 columns with their meaning ful names
```

Out[6]:
```
17
```

In [7]:
```
column = ['no. of wins', 'run scored', 'at bat','1st base hit','2nd base hit','3rd base hit','home run','walk','strike o
```

In [8]:
```
df.columns=column
```

In [9]:
```
df.columns
# here succefully we have change all shotcut column names with their meaningful names.
```

Out[9]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

In [10]:
```
df.head(5)
# here we can see the difference clearly
```

Out[10]:

	no. of wins	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned run	earned run average	complete game	shutout	save	error
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86

In [11]:
```
df.columns.unique()
#  here we can find the total unique number of columns of our dataset
```

Out[11]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

In [12]:
```
df.columns.nunique()
# here we can find the total number of unique columns present. i.e-17
```

Out[12]:
```
17
```

In [13]:
```
df.shape
#  here we can see that there are total 17 columns and 30 rows are present in the dataset.
```

Out[13]:
```
(30, 17)
```

In [14]:
```
df.dtypes
# here in the follwing data we can find that the datatype for all the columns in 'int64', except (earned run average) wh
#  'float64' beacause it is an average of earned run by the team, so it can in decimal form.
```

Out[14]:
```
no. of wins             int64
run scored              int64
at bat                  int64
1st base hit            int64
2nd base hit            int64
3rd base hit            int64
home run                int64
walk                    int64
strike out              int64
stolen base             int64
run average             int64
earned run              int64
earned run average    float64
complete game           int64
shutout                 int64
save                    int64
error                   int64
dtype: object
```

In [15]:
```
df.info()
#  here we can see that
# 1) total number for columns present : 17
# 2) total number of rows presnet : 30
# 3) total "data types present in data set" : 2 (i.e "int64 & float64")
#  out of which   16 columns of - int64
#                 1 column of - float64
# 4) NULL VALUES are may not present in dataset.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   no. of wins         30 non-null     int64  
 1   run scored          30 non-null     int64  
 2   at bat              30 non-null     int64  
 3   1st base hit        30 non-null     int64  
 4   2nd base hit        30 non-null     int64  
 5   3rd base hit        30 non-null     int64  
 6   home run            30 non-null     int64  
 7   walk                30 non-null     int64  
 8   strike out          30 non-null     int64  
 9   stolen base         30 non-null     int64  
 10  run average         30 non-null     int64  
 11  earned run          30 non-null     int64  
 12  earned run average  30 non-null     float64
 13  complete game       30 non-null     int64  
 14  shutout             30 non-null     int64  
 15  save                30 non-null     int64  
 16  error               30 non-null     int64  
dtypes: float64(1), int64(16)
memory usage: 4.1 KB
```

In [ ]:
```

```

CHECKING NULL VALUES

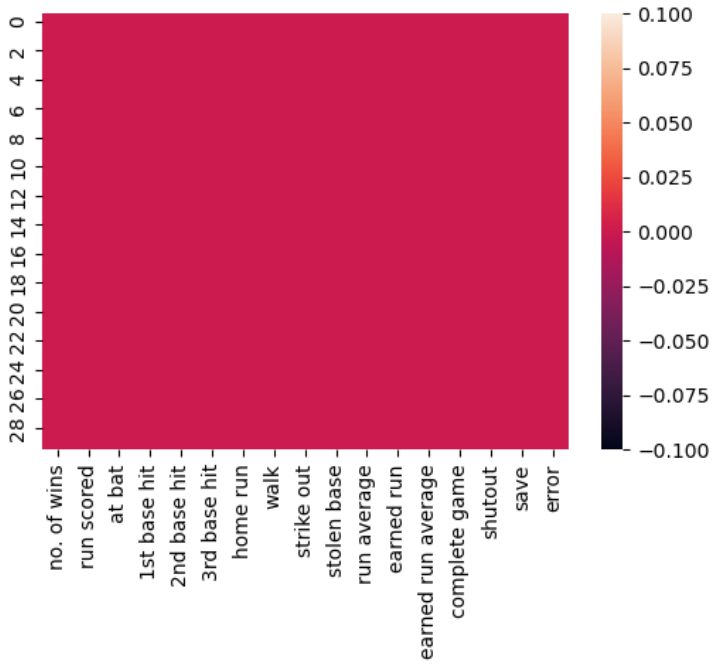
==========

In [16]:
```
df.isnull().sum()
# here clearly we can see that there is no presence of any null values.
```

Out[16]:
```
no. of wins           0
run scored            0
at bat                0
1st base hit          0
2nd base hit          0
3rd base hit          0
home run              0
walk                  0
strike out            0
stolen base           0
run average           0
earned run            0
earned run average    0
complete game         0
shutout               0
save                  0
error                 0
dtype: int64
```

In [17]:
```
plt.figure(figsize=(6,4))
sns.heatmap(df.isnull())

# here with the help of heatmap also we can see that there is not any presence of null values in the given dataset.
```

Out[17]: <AxesSubplot:>



In [ ]:
```

```

CHECKING UNIQUE VALUES & UNIVARIATE ANALYSIS

==========

In [18]:
```
df.nunique()
#  here we can find the number of unique values in each column.
# as we know that there are total 30 rows are present in the given dataset.
# and here we can see that the column which consist very less number of unique value is 'complete game' i.e 9, so we can
# 'complete game' column as a categorical column. rest of the columns are not considered as a categorical column.d.
```

Out[18]:
```
no. of wins           24
run scored            28
at bat                29
1st base hit          29
2nd base hit          22
3rd base hit          23
home run              27
walk                  29
strike out            29
stolen base           27
run average           30
earned run            30
earned run average    30
complete game          9
shutout               12
save                  20
error                 21
dtype: int64
```

In [19]:
```
#  1) ANALYSING COLUMN-1 (No. of wins) =====>
```

In [20]:
```
df['no. of wins'].unique()
# here we can find the uniqye values present in the mentioned column.
# this column represents the number of wins by different teams.
#  total no. of unique values present are 24, outof total 30 row's.
```

Out[20]:
```
array([ 95,  83,  81,  76,  74,  93,  87,  80,  78,  88,  86,  85,  68,
       100,  98,  97,  64,  90,  71,  67,  63,  92,  84,  79], dtype=int64)
```

In [21]:
```
df['no. of wins'].nunique()
# out of 30 there are 24 unique vlaues are present in the column.
```

Out[21]: 24

In [22]:
```
df['no. of wins'].max()
#  the maximum 'no. of wins by a team' present in the column is 100.
```

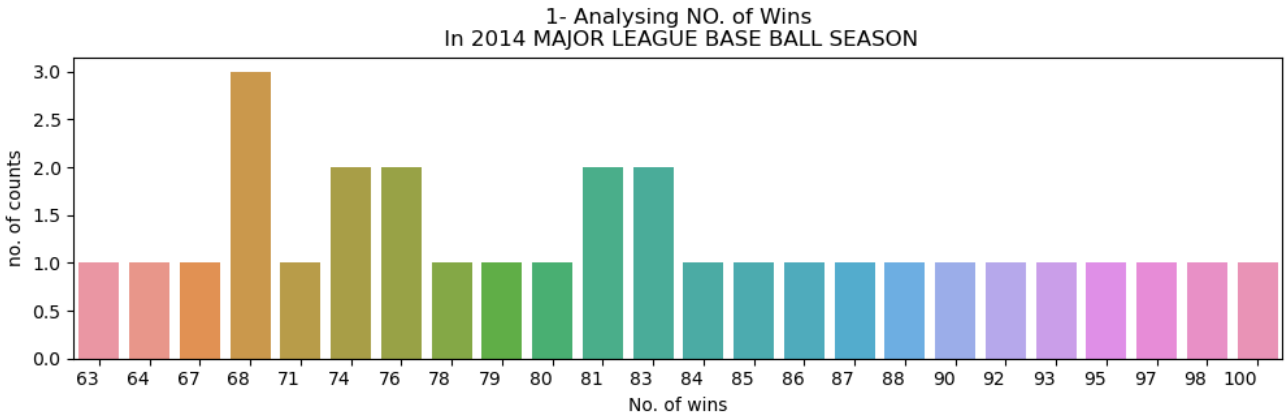
Out[22]: 100

In [23]:
```
df['no. of wins'].min()
#  the minimum 'no. of wins' by a pitcher is 63
```

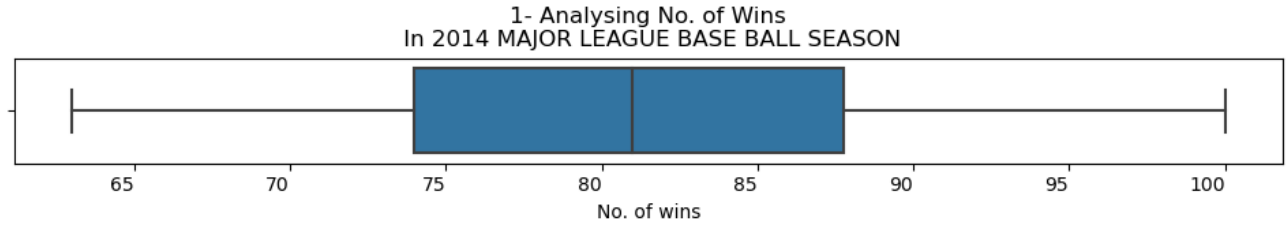
Out[23]: 63

In [24]:
```
# here in the given dataset, the 'no. of wins' column is our TARGET COLUMN which is dependent on other independent variab
#  so we have to predict the 'no. of wins' on the basis of other independent variabel present.
```

In [25]:
```
plt.figure(figsize = (12,3), facecolor='white')
plt.title('1- Analysing NO. of Wins\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.countplot(x='no. of wins', data=df)
plt.xlabel('No. of wins', fontsize = 10)
plt.xticks(rotation=0,ha ='right')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of count plot we can find that the hieghest no. of wins is 68. then after it is 74,76,81,83
# that means 68 wins are achived by 3 teams in the tournament.
```



In [26]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('1- Analysing No. of Wins\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.boxplot(x='no. of wins', data=df)
plt.xlabel('No. of wins', fontsize = 10)
plt.xticks(rotation=0,ha ='right')
# plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here from the below boxplot we can find that ,
#  The Minimum winnings of any team is bleow then 65
#  The Maximum winnings of any team is 100
#  & The Average Winnings of most of the team is lying in between 75 - 87 winnings.
```



In [ ]:

In [27]:
```
#  2) ANALYSING COLUMN-2  =====>
```

In [28]:
```
df['run scored'].unique()
```

Out[28]:
```
array([724, 696, 669, 622, 689, 891, 764, 713, 644, 748, 751, 729, 661,
       656, 694, 647, 697, 655, 640, 683, 703, 613, 573, 626, 667, 720,
       650, 737], dtype=int64)
```

In [29]:
```
df['run scored'].nunique()
```

Out[29]:
```
28
```

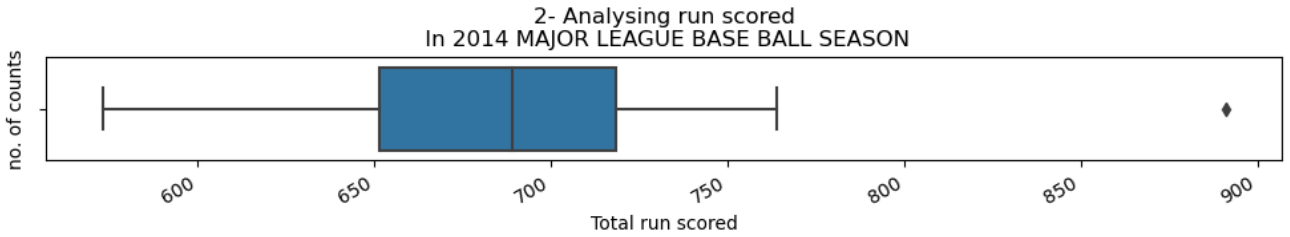
In [30]:
```
df['run scored'].min()
# here we can see the minimun runs scored by any team in MAJOR LEAGUE 2014
```

Out[30]:
```
573
```

In [31]:
```
df['run scored'].max()
# here we can see the maximum runs scored by any team in MAJOR LEAGUE 2014
```

Out[31]:
```
891
```

In [32]:
```
plt.figure(figsize = (12,1), facecolor='white')
plt.title('2- Analysing run scored\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.boxplot(x='run scored', data=df)
plt.xlabel('Total run scored', fontsize = 10)
plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of box plot we can find that the AVERAGE RUN'S scored by the teams is lying i between 650 - 7
# we can also say that 50% or most of the teams are scored in between 650 - 720 score.
# There may be presence of OUTLIERS
```



In [33]:
```
#  3) ANALYSING COLUMN-3  =====>
```

In [34]:
```
df['at bat'].nunique()
# here we find that there are no unique values in this column, because the total no. of uniques are 29 out of 30.
```

Out[34]:
```
29
```

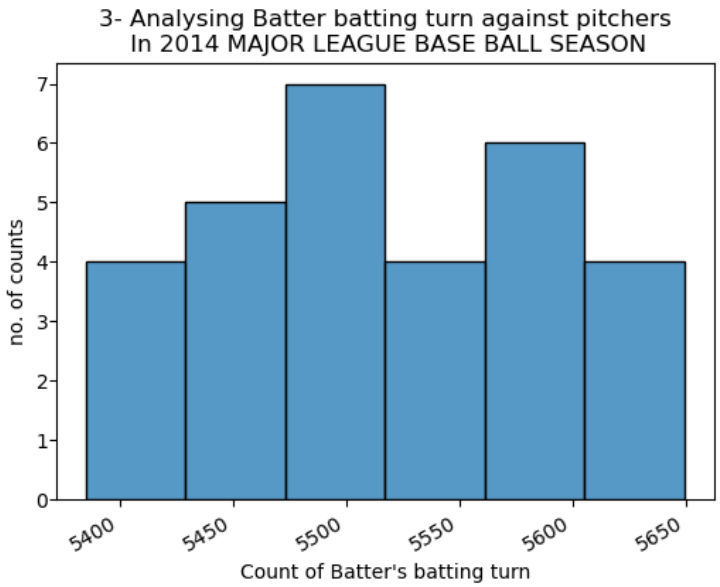
In [35]:
```
df['at bat'].min()
#  here the minimum turns of a batter against the pitchers are 5385
```

Out[35]:
```
5385
```

In [36]:
```
df['at bat'].max()
# and the maximum turns of batter against the pitchers is 5649. there is not much difference between the minimum and the
```

Out[36]:
```
5649
```

In [37]:
```
plt.figure(figsize = (6,4), facecolor='white')
plt.title('\n3- Analysing Batter batting turn against pitchers\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='at bat',data= df)
plt.xlabel("Count of Batter's batting turn", fontsize = 10)
plt.xticks(rotation=30,ha ='right')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum number of batting turn is 5500 & 5600
```



In [38]:
```
#  4) Analysing collumn 4 ====>
```

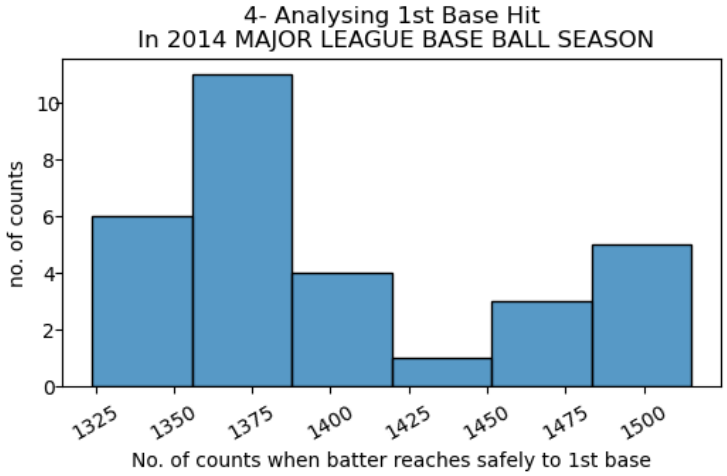
In [39]:
```
df['1st base hit'].unique()
```

Out[39]:
```
array([1497, 1349, 1395, 1381, 1515, 1480, 1397, 1370, 1383, 1495, 1419,
       1363, 1331, 1379, 1405, 1386, 1462, 1341, 1378, 1382, 1351, 1420,
       1361, 1374, 1346, 1486, 1494, 1324, 1479], dtype=int64)
```

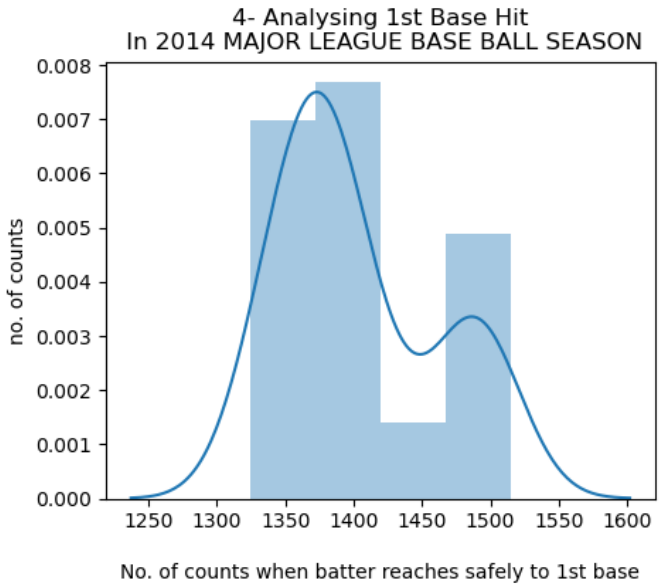
In [40]:
```
df['1st base hit'].nunique()
```

Out[40]:
```
29
```

In [41]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('4- Analysing 1st Base Hit\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='1st base hit', data=df)
plt.xlabel('No. of counts when batter reaches safely to 1st base', fontsize = 10)
plt.xticks(rotation=30,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter reaches safely
#  1st base without getting out is MAXIMUM in between 1350 - 1375.
# that means out of total counts 1350-1375 times is a maximum count when A BATTER REACHES SAFELY TO THE 1st BASE.
```



In [42]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('4- Analysing 1st Base Hit\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['1st base hit'])
plt.xlabel('\nNo. of counts when batter reaches safely to 1st base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of runs scored at FIRST BASE is be
# 1350-1400 times.
# here we can also find the second highest count is 1500.
```



In [ ]:

In [43]:
```
#  5) Analysing 2nd Base hit ====>
```

In [44]:
```
df['2nd base hit'].unique()
```

Out[44]:
```
array([300, 277, 303, 260, 289, 308, 272, 246, 278, 294, 279, 243, 262,
       288, 292, 274, 257, 295, 265, 236, 251, 263], dtype=int64)
```

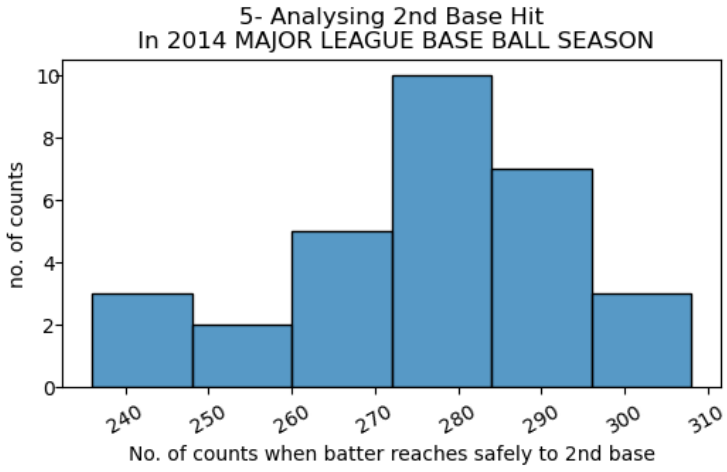
In [45]:
```
df['2nd base hit'].nunique()
#  here out of 30 values there are 22 unique values are present in the column.
```

Out[45]:
```
22
```

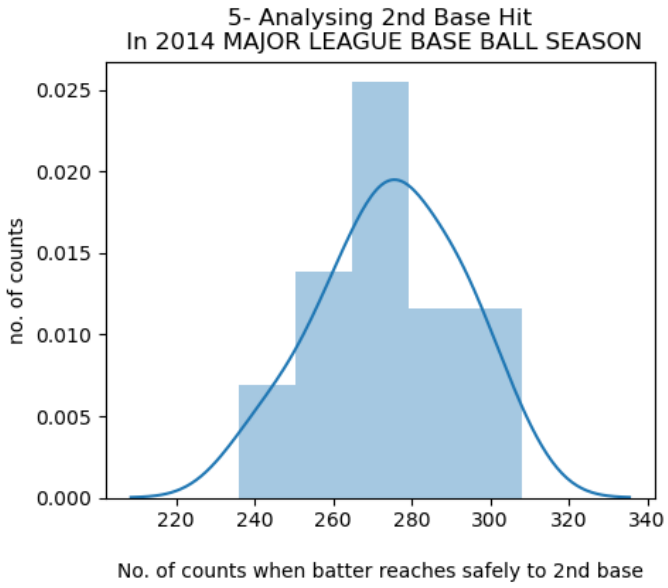
In [46]:
```
df['2nd base hit'].value_counts()
```

Out[46]:
```
272    3
260    2
289    2
278    2
277    2
274    2
288    2
300    1
292    1
251    1
236    1
265    1
295    1
257    1
243    1
262    1
279    1
294    1
246    1
308    1
303    1
263    1
Name: 2nd base hit, dtype: int64
```

In [47]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('5- Analysing 2nd Base Hit\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='2nd base hit', data=df)
plt.xlabel('No. of counts when batter reaches safely to 2nd base', fontsize = 10)
plt.xticks(rotation=30,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter reaches safely
#  2nd base without getting out is MAXIMUM in between 272-282.
# that means out of total counts 272-282 times is a maximum count when A BATTER REACHES SAFELY TO THE 2nd BASE.
```



In [48]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('5- Analysing 2nd Base Hit\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['2nd base hit'])
plt.xlabel('\nNo. of counts when batter reaches safely to 2nd base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of runs scored at SECOND BASE is l
# 271-280 times.
# here we can also find the second highest count is near about 260 times.
```



In [ ]:
```

```

In [49]:
```
# 6) Analysing 3rd BSE HIT column =====>
```

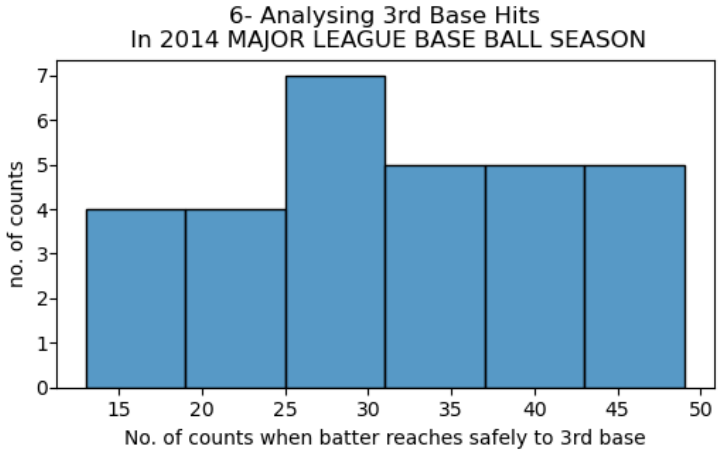
In [50]:
```
df['3rd base hit'].unique()
```

Out[50]:
```
array([42, 44, 29, 27, 49, 17, 19, 20, 32, 33, 26, 21, 22, 46, 39, 30, 34,
       13, 40, 18, 37, 48, 36], dtype=int64)
```

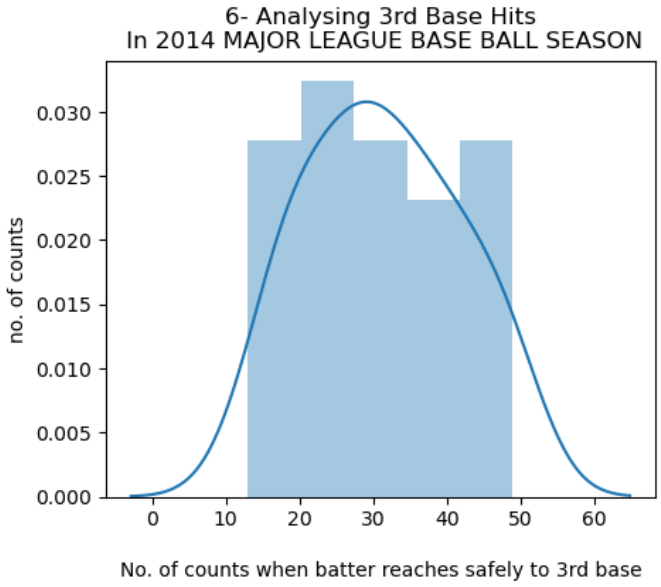
In [51]:
```
df['3rd base hit'].nunique()
# here out of total 30 values there are 23 unique vlaues are present in the column.
```

Out[51]:
```
23
```

In [52]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('6- Analysing 3rd Base Hits\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='3rd base hit', data=df)
plt.xlabel('No. of counts when batter reaches safely to 3rd base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter reaches safely
#  3rd base without getting out is MAXIMUM in between 25-30.
# that means out of total counts 25-30 times count when A BATTER REACHES SAFELY TO THE 3rd BASE.
```



In [53]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('6- Analysing 3rd Base Hits\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['3rd base hit'],bins=5)
plt.xlabel('\nNo. of counts when batter reaches safely to 3rd base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of runs scored at THIRD BASE is be
# 25-30 times.
```



In [ ]:

In [ ]:

In [54]:
```
# 7) Analysing HOME RUN 7th column =====>
```

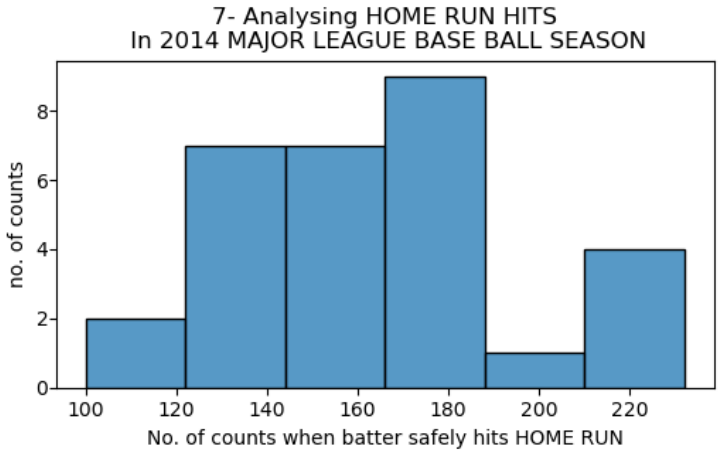
In [55]:
```
df['home run'].unique()
```

Out[55]:
```
array([139, 156, 141, 136, 151, 232, 212, 217, 167, 161, 172, 230, 176,
       198, 146, 137, 140, 171, 145, 177, 120, 100, 130, 187, 154, 148,
       186], dtype=int64)
```

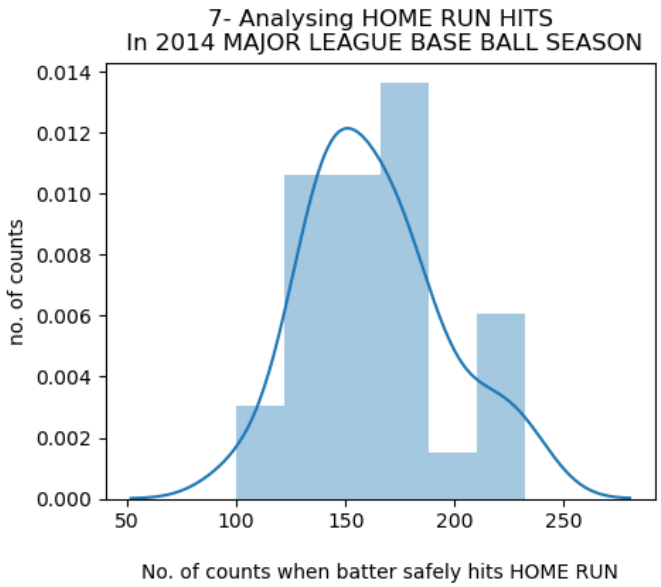
In [56]:
```
df['home run'].nunique()
# out of 30 there are 27 unique values are present
```

Out[56]:
```
27
```

In [57]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('7- Analysing HOME RUN HITS\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='home run', data=df)
plt.xlabel('No. of counts when batter safely hits HOME RUN', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter safely hits H
#  without getting out is MAXIMUM in between 165-180.
# that means out of total counts 165-180 times count when A BATTER HITS HOME RUN SAFELY .
```



In [58]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('7- Analysing HOME RUN HITS\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['home run'])
plt.xlabel('\nNo. of counts when batter safely hits HOME RUN', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of HOME RUN'S hitted by the batsm
#  in between 165-180
```



In [ ]:
```

```

In [59]:
```
#  8) Analysing no. of WALKS by batter =====>
```

In [60]:
```
# here the meaning of walk  / Base on balls, that means when a batter receive's 4 pitches or balls succefully, then he
#  is awarded to walk to 1st FIRST BASE  without possibility of beign OUT.
```

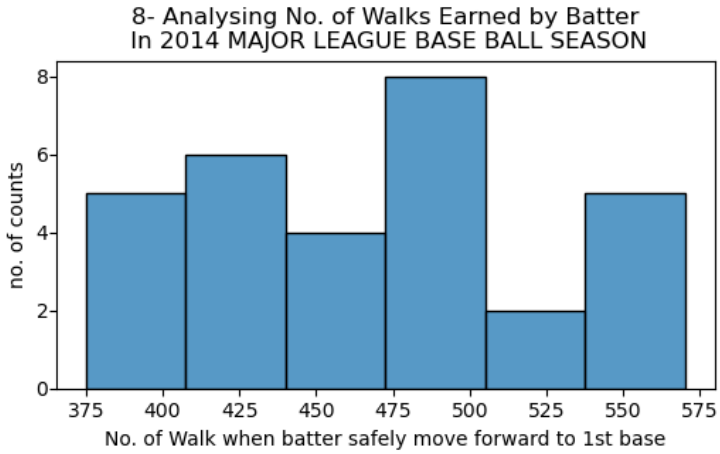
In [61]:
```
df['walk'].unique()
```

Out[61]:
```
array([383, 439, 533, 404, 455, 570, 554, 418, 436, 478, 503, 486, 435,
       475, 506, 461, 567, 412, 496, 488, 539, 375, 471, 387, 563, 457,
       490, 426, 388], dtype=int64)
```

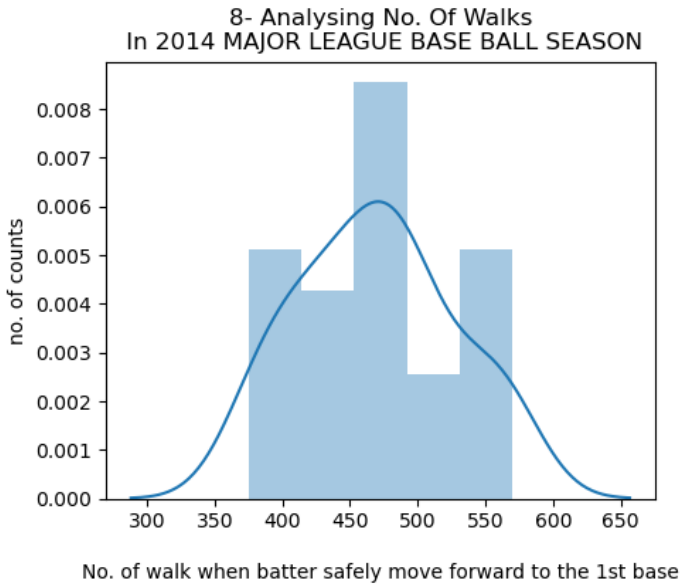
In [62]:
```
df['walk'].nunique()
# out of 30 there are 29 unique values are present in the column, that means there are no such unique vlaues are present
```

Out[62]:
```
29
```

In [63]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('8- Analysing No. of Walks Earned by Batter\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='walk', data=df)
plt.xlabel('No. of Walk when batter safely move forward to 1st base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter safely WALK to
#  1st Base  as a reward ,without getting out is MAXIMUM in between 475-500 times.
# that means out of total counts 475-500 times count when A BATTER RECEIVES 4 PITCHES SUCCEFULLY AND AS A REWARD HE FORAI
#  THE 1st BASE WITHOUT CALLED OUT.
```



In [64]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('8- Analysing No. Of Walks\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['walk'])
plt.xlabel('\nNo. of walk when batter safely move forward to the 1st base', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of walks earned by the batsman is
#  in near at the count of 475.
```



In [ ]:

In [65]:
```
# 9) Analysing STRIKE OUT =====>
```

In [66]:
```
#  Strike out is actually when a batter accumulates three strkkes without hitting the ball then the batter is out.
```

In [67]:
```
df['strike out'].unique()
```

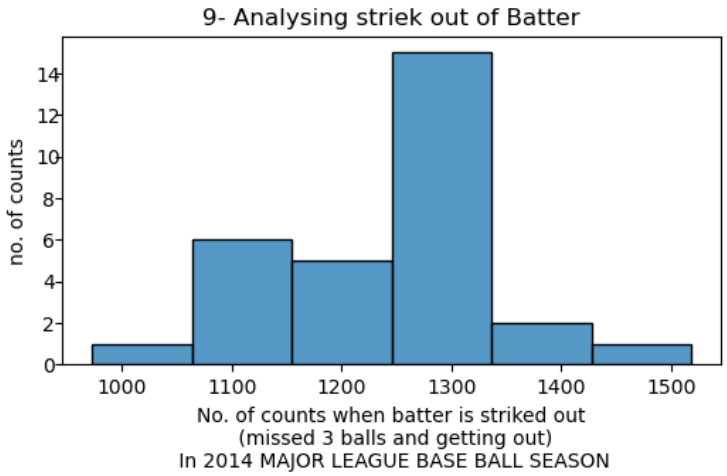
Out[67]:
```
array([ 973, 1264, 1157, 1231, 1259, 1151, 1227, 1331, 1310, 1148, 1233,
       1392, 1150, 1336, 1119, 1267, 1322, 1518, 1299, 1255, 1290, 1344,
       1107, 1274, 1258, 1159, 1312, 1327, 1283], dtype=int64)
```

In [68]:
```
df['strike out'].nunique()
#  there are no such unique values in the column.
```

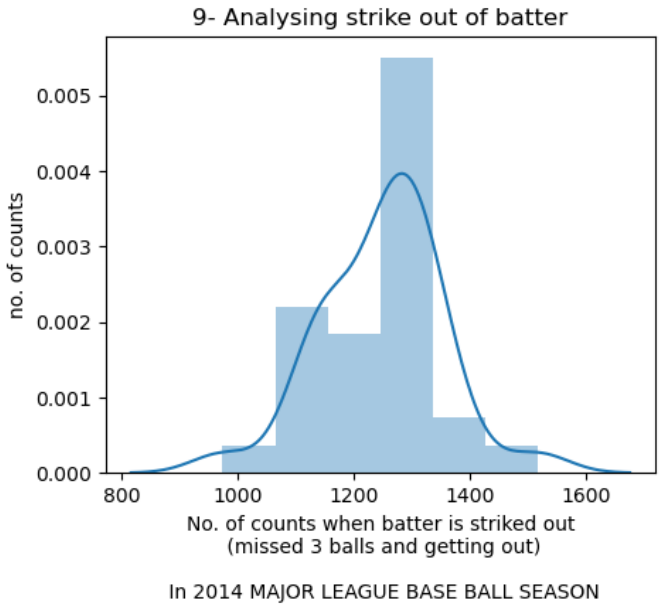
Out[68]:
```
29
```

In [69]:
```
#  here we want to find that in the given dataset howmany maximum counts occurs when the is 'strikeout'.
```

In [70]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('9- Analysing striek out of Batter')
sns.histplot(x='strike out', data=df)
plt.xlabel('No. of counts when batter is striked out\n (missed 3 balls and getting out)\n In 2014 MAJOR LEAGUE BASE BALL
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the total no. of maximum counts when batter is STRIKED OU
#  in between 1275 - 1325 times.
# we called a striked out when a batter missed 3 balls, that mean in the whole tournamnet 1300 x 3 = 3900 balls are misse
```



In [71]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('9- Analysing strike out of batter')
sns.distplot(df['strike out'])
plt.xlabel('No. of counts when batter is striked out\n (missed 3 balls and getting out)\n\n In 2014 MAJOR LEAGUE BASE BA
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot also we can clearly see that the maximum no. of batters are striked out
#  in near at the count of 1275.
```



In [ ]:

In [ ]:

In [72]:
```
#  10) Analysing Stolen Base =====>
```

In [73]:
```
#  Stolen Base means is - when a runner advances to a base , and the advaces should be credited to the action of the runn
#  or we can also say 'the number of advances by the runner while the ball is in defence possession'
```

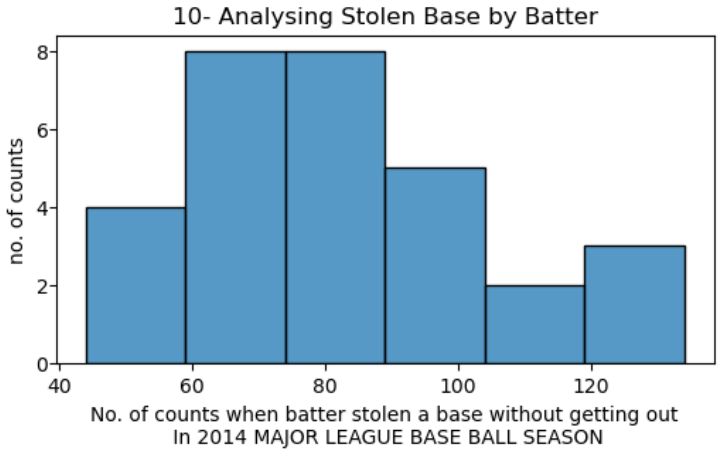
In [74]:
```
df['stolen base'].unique()
```

Out[74]:
```
array([104,  70,  86,  68,  83,  88,  63,  44,  87,  71, 101, 121,  52,
        69,  78,  98,  95,  84, 134,  51,  57, 112,  59,  93, 132,  82,
        97], dtype=int64)
```

In [75]:
```
df['stolen base'].nunique()
#  out of 30 values there are 27 uniques values are present in the column, that means there are no such unique values are
```

Out[75]:
```
27
```

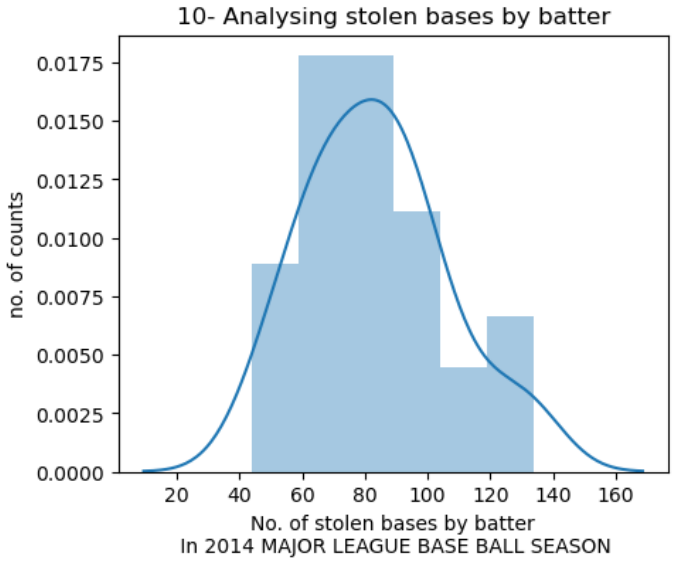
In [76]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('10- Analysing Stolen Base by Batter')
sns.histplot(x='stolen base', data=df)
plt.xlabel('No. of counts when batter stolen a base without getting out\n In 2014 MAJOR LEAGUE BASE BALL SEASON', fontsi
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can see that 60 - 85 times is a maximum counts when  batters are stolen a base and
#  (advace credited to the runner) wihtout getting out
```



In [ ]:

In [ ]:

In [77]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('10- Analysing stolen bases by batter')
sns.distplot(df['stolen base'])
plt.xlabel('No. of stolen bases by batter\n In 2014 MAJOR LEAGUE BASE BALL SEASON', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, maximum no. bases stolen by batter is in between
```



In [ ]:

In [ ]:

In [78]:
```
#  11) Analysing Run Average ====>
```

In [79]:
```
#  The Run average is the Rate at which the runs are scored by any team.
```

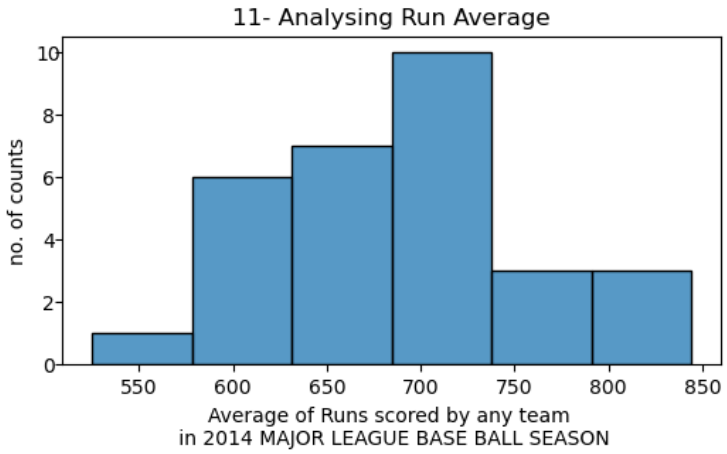
In [80]:
```
df['run average'].unique()
```

Out[80]:
```
array([641, 700, 640, 701, 803, 670, 698, 693, 642, 753, 733, 618, 675,
       726, 729, 525, 596, 608, 737, 754, 613, 635, 678, 760, 809, 595,
       627, 713, 731, 844], dtype=int64)
```

In [81]:
```
df['run average'].nunique()
# there are no such any unique values are there.
```

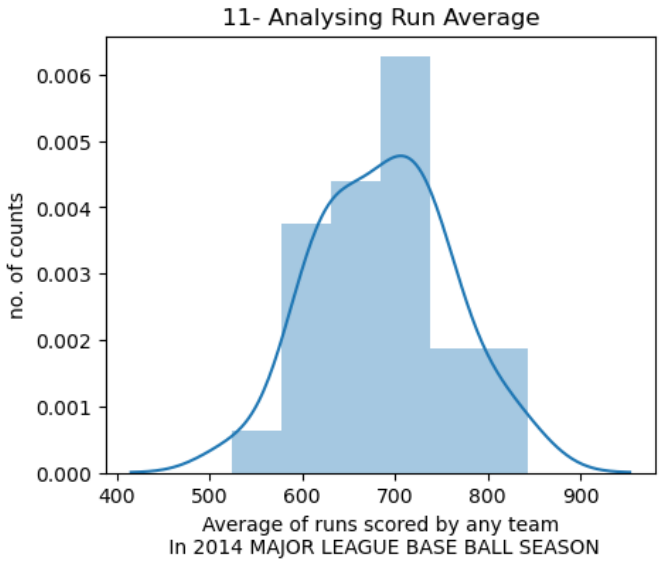
Out[81]:
```
30
```

In [82]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('11- Analysing Run Average')
sns.histplot(x='run average', data=df)
plt.xlabel('Average of Runs scored by any team \n in 2014 MAJOR LEAGUE BASE BALL SEASON', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can see that the maximum run average scored by some teams is highest in between 675-725
# that mean in 2014 MAJOR LEAGUE BASE BALL SEASON highest run average acheived a some teams is in between 675-725
#  and the SECONDLY  it is in between 575-675
#  and then THIRDLY it is in between 750-875
```



In [ ]:

In [83]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('11- Analysing Run Average')
sns.distplot(df['run average'])
plt.xlabel('Average of runs scored by any team\n In 2014 MAJOR LEAGUE BASE BALL SEASON', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, maximum run average is near to 700
```



In [ ]:
```

```

In [84]:
```
#  12) Analysing Earned Run =====>
```

In [85]:
```
# The Earned Runs are those run's which are earned by the batter which are not occurs as a result of error or passed bal
#  here we are going to analyse how many runs are earned by any team in the tournamnet.
```

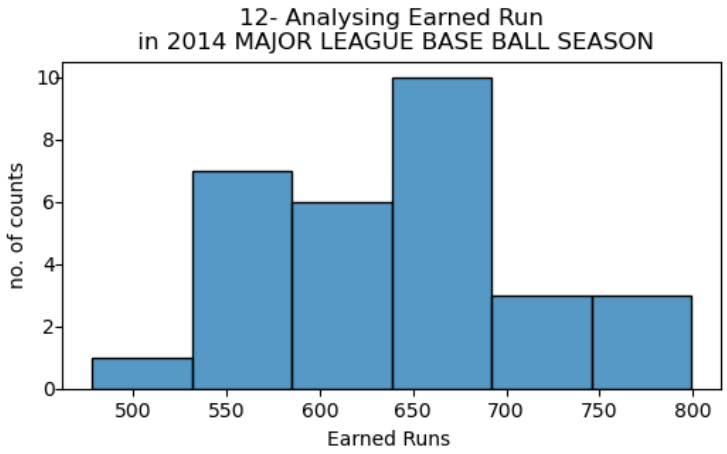
In [86]:
```
df['earned run'].unique()
```

Out[86]:
```
array([601, 653, 584, 643, 746, 609, 652, 646, 604, 694, 680, 572, 630,
       677, 664, 478, 532, 546, 682, 700, 557, 577, 638, 698, 749, 553,
       597, 659, 655, 799], dtype=int64)
```

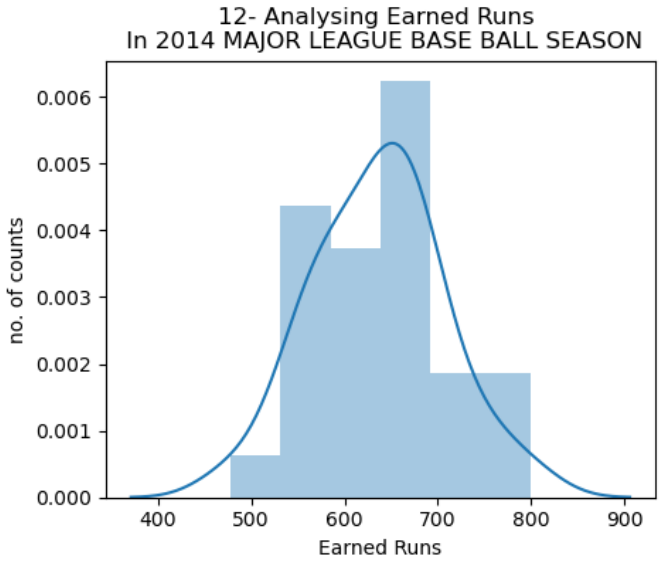
In [87]:
```
df['earned run'].nunique()
#  there are not such any un ique values are present in the given data.
```

Out[87]:
```
30
```

In [88]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('12- Analysing Earned Run\n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='earned run', data=df)
plt.xlabel('Earned Runs ', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here we can see that the maximum count of Earned Runs in between 650-700
```



In [89]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('12- Analysing Earned Runs \n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['earned run'])
plt.xlabel('Earned Runs', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, maximum Earned Runs are near by 675.
```



In [ ]:

In [90]:
```
#  13) Analysis of Earned Run Average =====>
```

In [91]:
```
# The Earned Run Average is refers to the average of earned runs allowed by a pitcher per NINE(9) INNINGS
#  It can be calculated as DIVIDING THE NUMBER OF EARNED RUN by / NUMBER OF INNINGS * multiplies by NINE
```

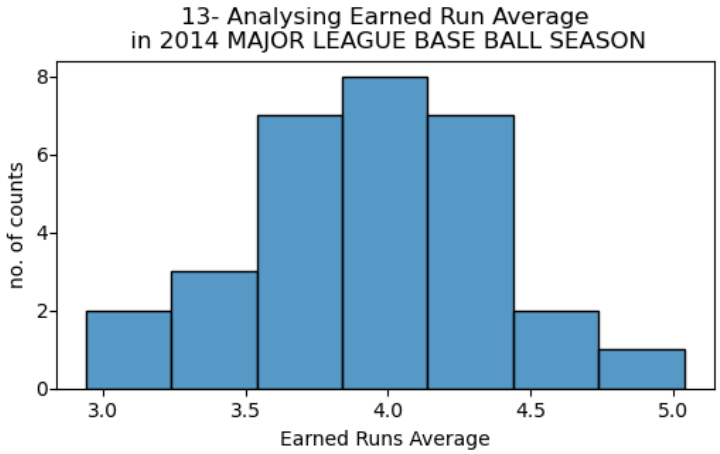
In [92]:
```
df['earned run average'].unique()
```

Out[92]:
```
array([3.73, 4.07, 3.67, 3.98, 4.64, 3.8 , 4.03, 4.05, 3.74, 4.31, 4.24,
       3.57, 3.94, 4.16, 4.14, 2.94, 3.21, 3.36, 4.28, 4.33, 3.43, 3.62,
       4.02, 4.41, 4.69, 3.44, 3.72, 4.04, 4.09, 5.04])
```

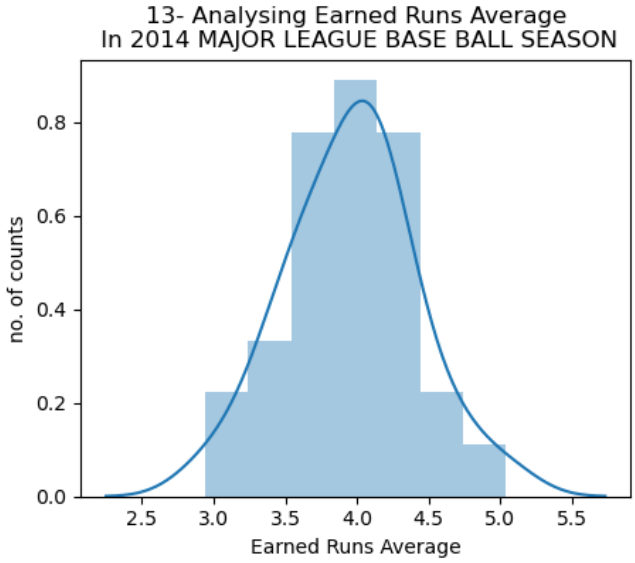
In [93]:
```
df['earned run average'].nunique()
# no such uniques values are present
```

Out[93]:
```
30
```

In [94]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('13- Analysing Earned Run Average\n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='earned run average', data=df)
plt.xlabel('Earned Runs Average', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum count of EARNED RU AVRAGE is lying in between
#  the highest EARNED RUN AVERAGE COUNT is 4.0
#  the highets EARNED RUN AVERAGE SCORE BY ANY TEAM IS 5.0
#  The MINIMUM EARNED RUN AVERAGE SCORE BY ANT TEAM IS BELOW 3.0
```



In [95]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('13- Analysing Earned Runs Average\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['earned run average'])
plt.xlabel('Earned Runs Average', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, the maximum count of Earned Runs average is 4.0.
#  and the maximum earned run average scored by any team is 5.5
#  and the minimum earned run average scored by any team is below 2.5
```



In [ ]:
```

```

In [96]:
```
#  14) Analysing Complete Game =====>
```

In [97]:
```
#  Complete Game is - act of a pitcher pitching an entire game wihtout the benfit of the releife pitcher,
#  or we can also say - Number of games where player was the only pitcher of their team.
```

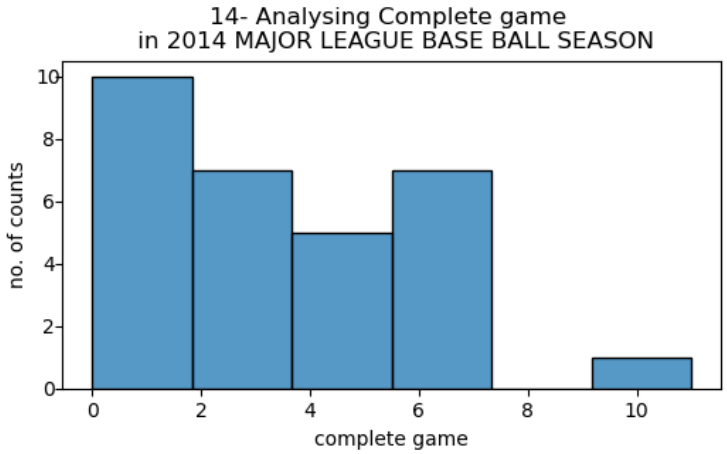
In [98]:
```
df['complete game'].unique()
```

Out[98]:
```
array([ 2, 11,  7,  3,  0,  1,  5,  6,  4], dtype=int64)
```

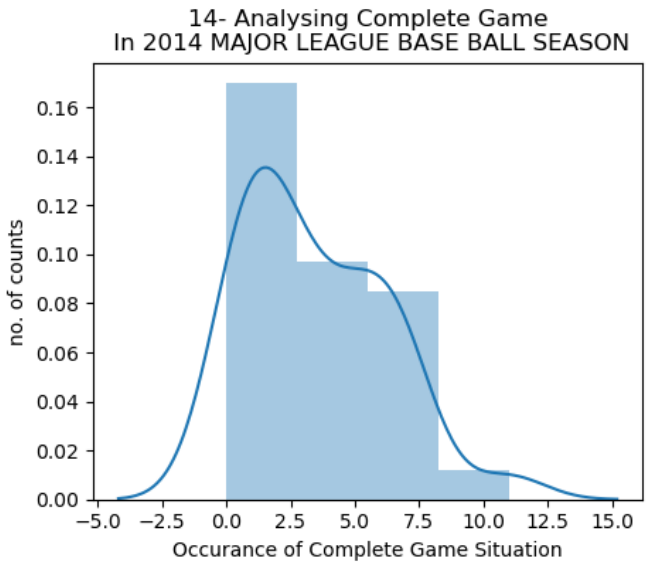
In [99]:
```
df['complete game'].nunique()
```

Out[99]:
```
9
```

In [100]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('14- Analysing Complete game \n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='complete game', data=df)
plt.xlabel('complete game', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum count of Complete Game is lying in between 0-2
#  that means 0-2 is maximum times when this situation is happen that - Player was the only Pitcher of their team
#  the average count of complet game is lying in between 2-6, i.e in an average 2-6 time the situation occurs when "the p
#  was the only pitcher of their team".
```



In [101]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('14- Analysing Complete Game\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['complete game'])
plt.xlabel('Occurance of Complete Game Situation', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, the maximum count of complete game situation is c
#  in between 0-2 times
# Highest occurance of this situation is 15. (which is vermuch higher as compared to the average count)
```



In [ ]:

In [102]:
```
#  15) Analysing ShutOut =====>
```

In [103]:
```
#  ShutOut is - Number of complte games pitched with no runs allowed.
```

In [104]:
```
df['shutout'].unique()
```

Out[104]:
```
array([ 8, 12, 10,  9,  4, 13, 15, 21,  7, 14, 18,  6], dtype=int64)
```

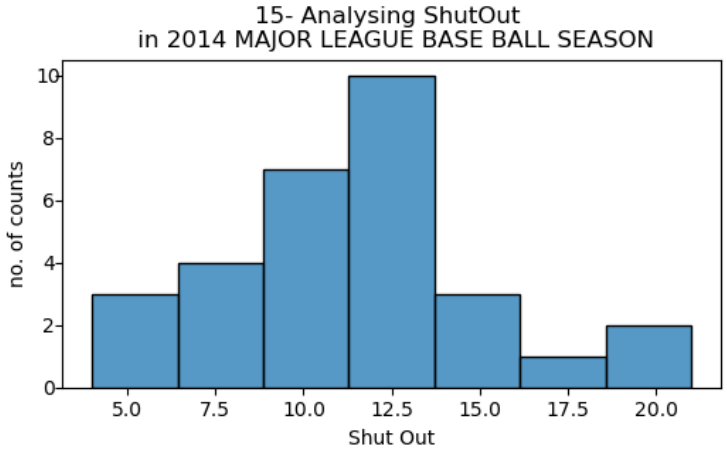
In [105]:
```
df['shutout'].nunique()
# out 0f 30 vales there are 12 numbers of unique values are present.
```

Out[105]:
```
12
```

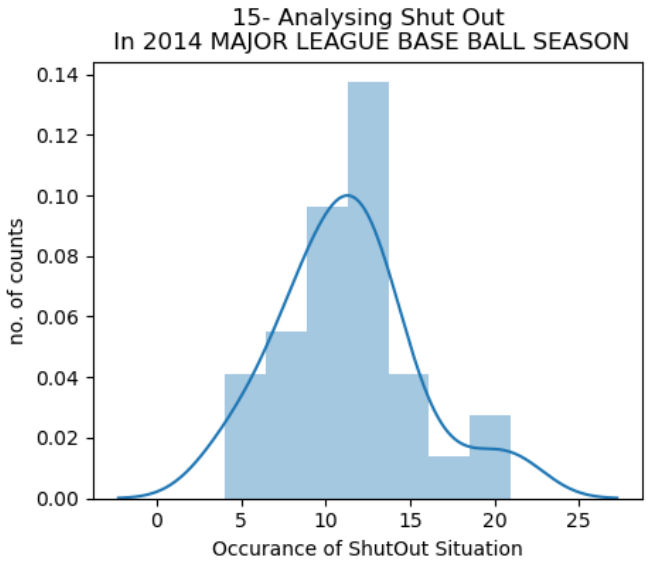
In [106]:
```
df['shutout'].value_counts()
#  here we find that the value counts of no. of shutouts are maximum in between 10-13
```

Out[106]:
```
12    7
10    5
13    3
8     2
9     2
4     2
15    2
21    2
7     2
14    1
18    1
6     1
Name: shutout, dtype: int64
```

In [107]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('15- Analysing ShutOut \n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='shutout', data=df)
plt.xlabel('Shut Out', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum count of shutout is lying in between 10-13
#  that means 10-13 is maximum times when this situation is happen .
# the minimum of this shutout situation is below 5
# and the maximum is above 20
```



In [108]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('15- Analysing Shut Out\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['shutout'])
plt.xlabel('Occurance of ShutOut Situation', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, the maximum count of shutout situation is occures
#  in between 10-13 times
# Highest occurance of this situation is above 25.
```



In [ ]:
```

```

In [109]:
```
# 16) Analysing Save Game =====>
```

In [110]:
```
# Save Game is credited to the pitcher who fineshes a game for winning.
```

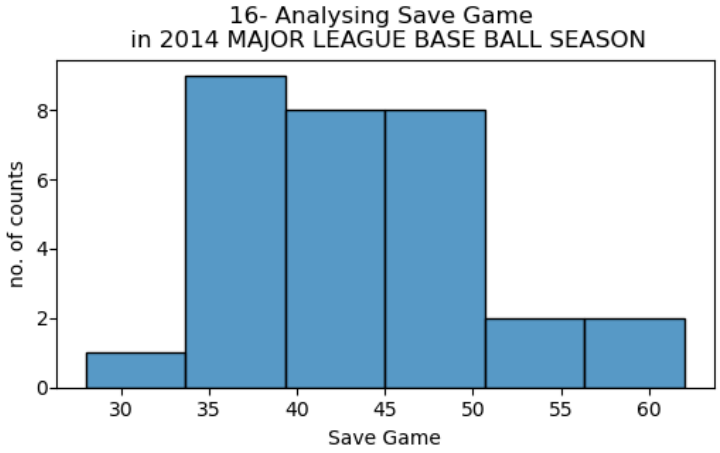
In [111]:
```
df['save'].unique()
```

Out[111]:
```
array([56, 45, 38, 37, 35, 34, 48, 43, 60, 40, 39, 46, 28, 62, 54, 50, 41,
       44, 47, 36], dtype=int64)
```

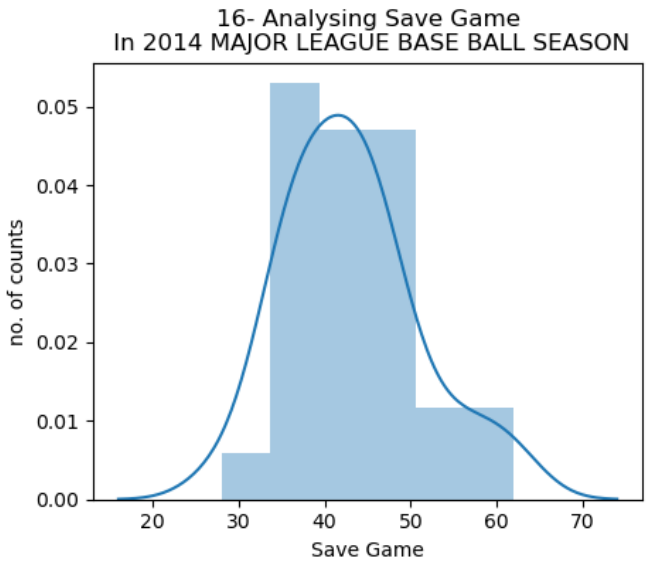
In [112]:
```
df['save'].nunique()
#  out of 30 values there are 20 unique vlaues are present in the column.
```

Out[112]:
```
20
```

In [113]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('16- Analysing Save Game \n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='save', data=df)
plt.xlabel('Save Game', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum count of save game is lying in between 35-50
#  that means 35-50 counts are the highest counsts when a team is saved their game
# maximum no. of teams are saved their games in between 35-50
```



In [114]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('16- Analysing Save Game\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['save'])
plt.xlabel('Save Game', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, the maximum count of save game condition is 35-50
# i.e maximum no. of teams are able to save their games in between 35-50 times.
#  the highesht count is above 70.
# the lowest count is below 20 .
```



In [ ]:

In [115]:
```
# 17) Analysing Error's =====>
```

In [116]:
```
#  Error's are the condition when a fielder missplaying a ball in a manner that allow a batter to advance one or more bas
# i.e the benifit of batter due to missfielding of a fielder.
```

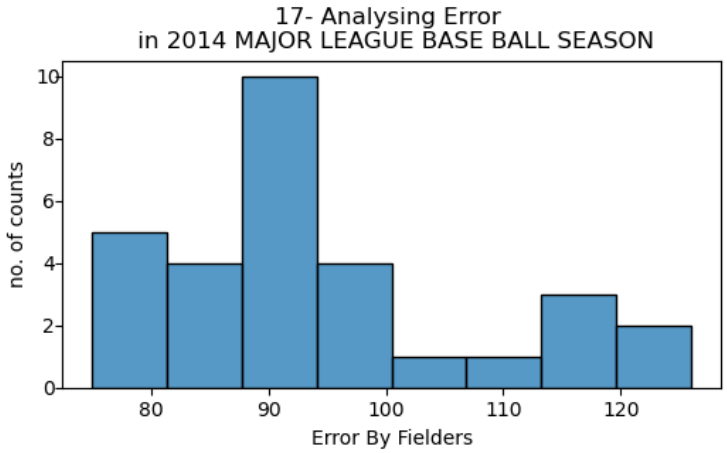
In [117]:
```
df['error'].unique()
```

Out[117]:
```
array([ 88,  86,  79, 101,  93,  77,  95,  97, 119,  85,  94, 126,  96,
       122, 111, 116,  90, 117,  75,  78,  92], dtype=int64)
```

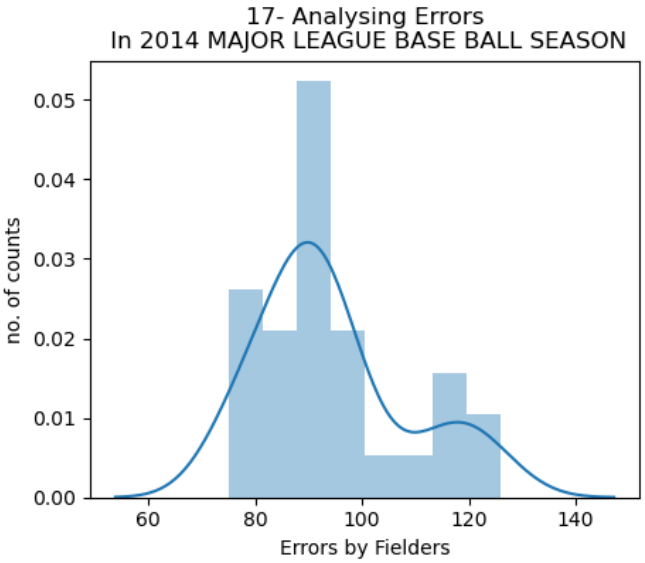
In [118]:
```
df['error'].nunique()
# out of 30 there are 21 unique values are present here.
```

Out[118]:
```
21
```

In [119]:
```
plt.figure(figsize = (6,3), facecolor='white')
plt.title('17- Analysing Error \n in 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.histplot(x='error', data=df)
plt.xlabel('Error By Fielders', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
plt.yticks(rotation=0, ha = 'center')
plt.show()

#  here below with the help of histogram plot we can find that the maximum 85-90 times the errors are done (MISSFIELDING)
#  the fielders
# i.e in the overall league 85-90 times is the maximum counts when the errors are done by the fielders.
```



In [120]:
```
plt.figure(figsize = (5,4), facecolor='white')
plt.title('17- Analysing Errors\n In 2014 MAJOR LEAGUE BASE BALL SEASON')
sns.distplot(df['error'])
plt.xlabel('Errors by Fielders', fontsize = 10)
plt.xticks(rotation=0,ha ='center')
plt.ylabel('no. of counts', fontsize = 10)
# plt.yticks(rotation=0, ha = 'center')
plt.show()

# here with the help of distribution plot we can also clearly see that, the maximum count of errors by fielders is 85-90
# i.e maximum no. of error done by the fielders are in between 85-90 times.
#  the highesht no. of count error is above 140.
# the lowest no. of count of erros is below 60 .
```



========== UNIVARIATE ANALYSIS COMPLETED
==========

====== HERE ABOVE I HAVE COMPLETED UNIVARIATE ANALYSIS IN SIMPLE DISTPLOT & HISTPLOT, BECAUSE I THOUGHT IT IS GOING TO BE EASIER FOR THE CLIENT TO UNDERSTAND THE SITUATION IN SIMPLE MANNER ==========

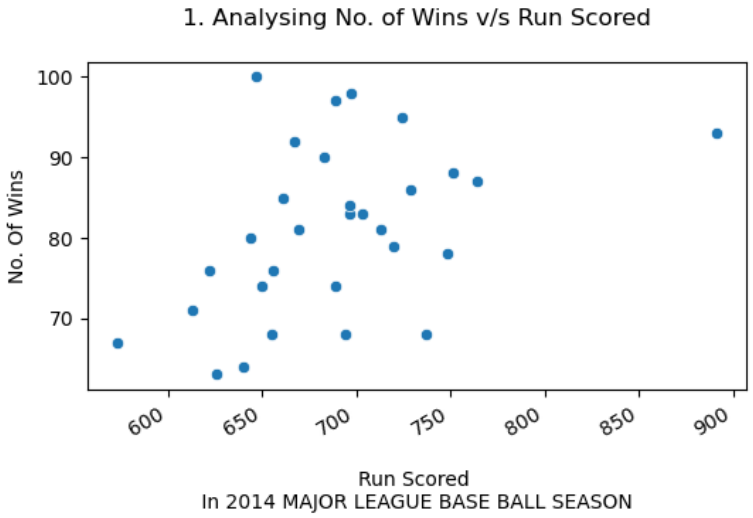
In [ ]:
```
```

BIVARIATE ANALYSIS ==========

In [121]:
```
#  Here as we know that our target column is 'no. of wins', we have to predict 'no. of wins' on the basis of other indepe
#  ...variables.
#  so in this barivariate analysis phase we are going to analyse each column with our target column.
```

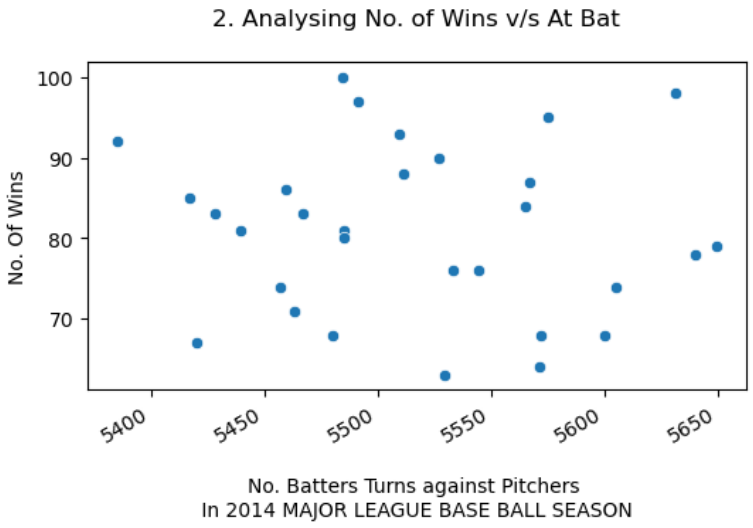
In [122]:
```
# 1) Analysing 'no. of wins' with 'Runs Scored'.
```

In [123]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n1. Analysing No. of Wins v/s Run Scored\n')
sns.scatterplot (x= 'run scored', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nRun Scored \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  hree we can clearly find that the relation of 'no. of wins' with 'runs scored' is positively related..
# ... more the no. of run scored by any team , is more the probability to win the game.
#  nad here we can see it clealrly in the following scatter plot graph.
```



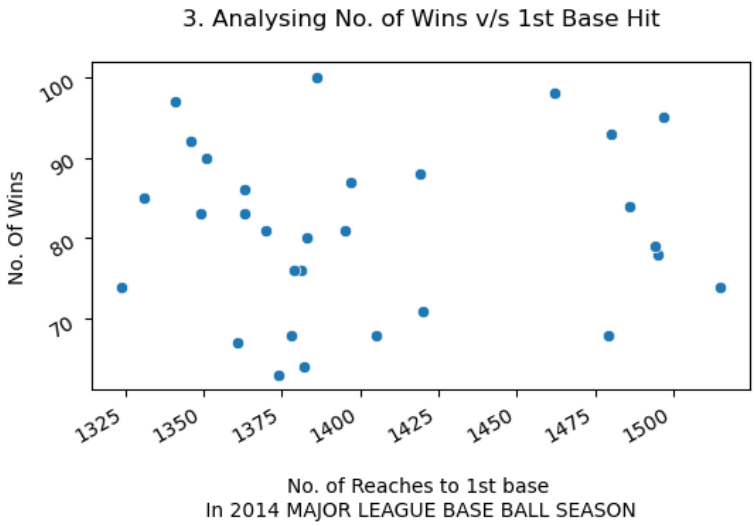
In [124]:
```
# 2) Analysing 'no. of wins' with 'at bat'(batters turns againdt putchers).
```

In [125]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n2. Analysing No. of Wins v/s At Bat\n')
sns.scatterplot (x= 'at bat', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. Batters Turns against Pitchers \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
# plt.yticks (rotation = 0, ha='center')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here we didnt find such strong relation between 'no. of wins' & 'at bat'
#  so we can say that 'no. of wins' is not much related with 'at bat' (batters turn against pitchers)
```



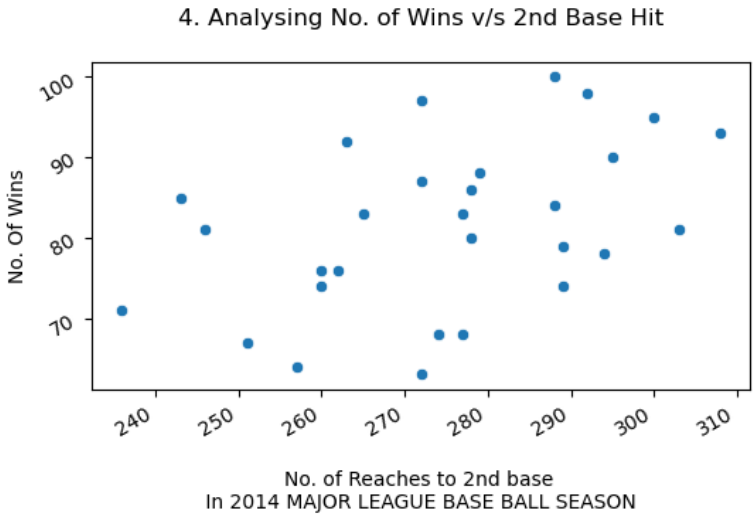
In [126]:
```
# 3) Analysing 'no. of wins' with '1st base hit'.
```

In [127]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n3. Analysing No. of Wins v/s 1st Base Hit\n')
sns.scatterplot (x= '1st base hit', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Reaches to 1st base \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here also we can find the little positvie relation between 'no. of wins' & 'no. of 1st base hit'
#  we find more the 'no. of 1st base hit' is in between 1330-1400.
# there are also '1st base hit' are in between 1475-1500, we can see here probability or no. of wins are higher with more
# ... no . of 1st base hit.
```



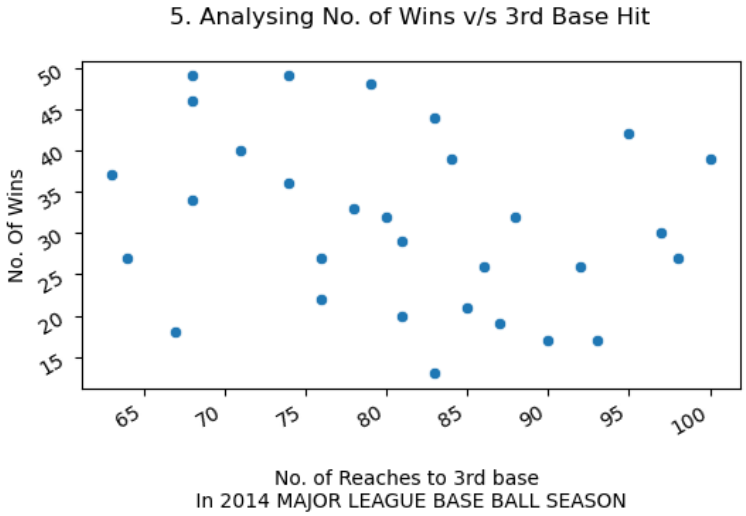
In [128]:
```
# 4) Analysing 'no. of wins' with '2nd Base hit'.
```

In [129]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n4. Analysing No. of Wins v/s 2nd Base Hit\n')
sns.scatterplot (x= '2nd base hit', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Reaches to 2nd base \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here also we can find the positvie relation between 'no. of wins' & 'no. of 2nd base hit'
#  more the 'no. of 2nd base hit' is more the 'no. of winning probability'.
# 'no. of 2nd base hits ' are less as compared to 'no. of 2nd base hits'
```



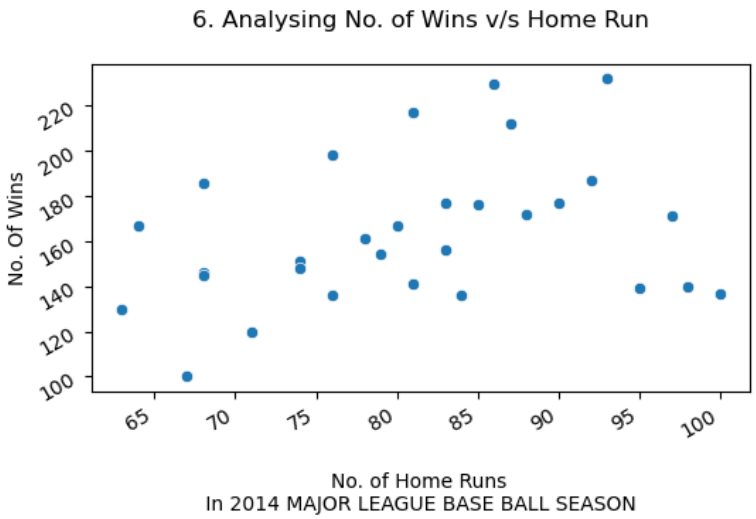
In [130]:
```
#  5) Analysing No. of wins  v/s  3rd Base hit.
```

In [131]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n5. Analysing No. of Wins v/s 3rd Base Hit\n')
sns.scatterplot (x= 'no. of wins', y = '3rd base hit', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Reaches to 3rd base \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can find NOT SOO STRONG RELATION with no. of wins
# here also we can find the 'no of wins' with '3rd base hits' are less as compare to 'no. of wins with 2nd & 1st base'
# the highest no. of wins are upto 100 in 1st & 2nd base hit, but here in 3rd base hitd the highest no. of wins are.....
#  ....upto 50 only...
#  & No. of 3rd base hits are also less as compare to no. 1st & 2nd base hit. which is obevious.
# here in the below scatterplot we can find that 'No. of wins ' are HIGHER in the range of 65-85 of 'no. of 3rd base hit
```

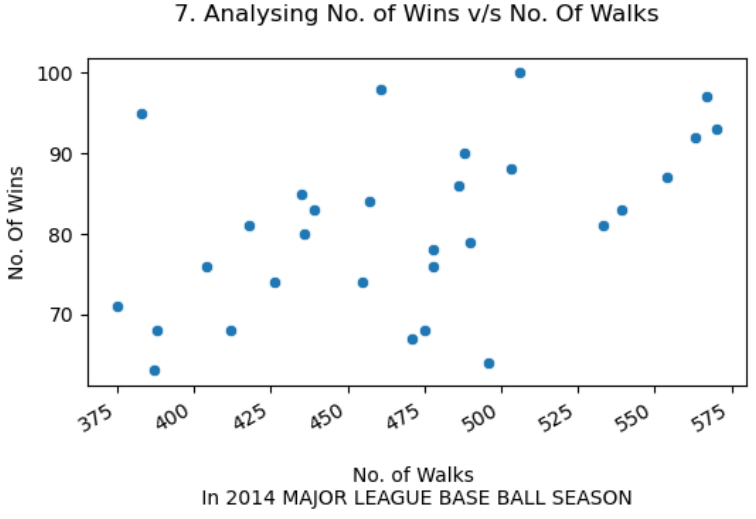


In [132]:
```
#  6.) Analysing No. of wins wih HomeRun ===>
```

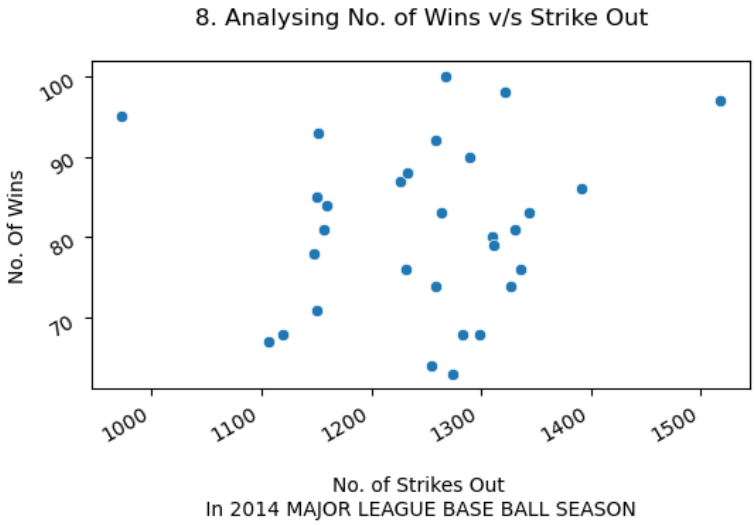
In [133]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n6. Analysing No. of Wins v/s Home Run\n')
sns.scatterplot (x= 'no. of wins', y = 'home run', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Home Runs \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Here we can find a STRONG POSITIVE CORRELATION of 'No. Of Wins' with 'HOME RUN', because here we can see that the 'no.
#  are HIGHEST ABOVE 220, which is approx double as compared to 'No. of 1st, 2nd, or 3rd Base hit'(there we can find the
# ....highest no. of wins are upto 100 & 50 only)
# so here definetly we can say that MORE THE NO. OF HOME RUNS ARE MORE THE NO. OF WINS.
#  And here we can find the MAXIMUM NO. OF HOME RUNS are upto 100.
```



In [134]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n7. Analysing No. of Wins v/s No. Of Walks\n')
sns.scatterplot (x = 'walk', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Walks \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
# plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Here we can clearly see the POSITIVE CORRELATION of N. of Wins with No. of Walks
# & we can also find here more the no. of walks are more the PROBABILITY OF NO. OF WINS.
# here the no of walks in between 525-575 has higher probabbility of no. of wins, & it is very positively correlated .
```

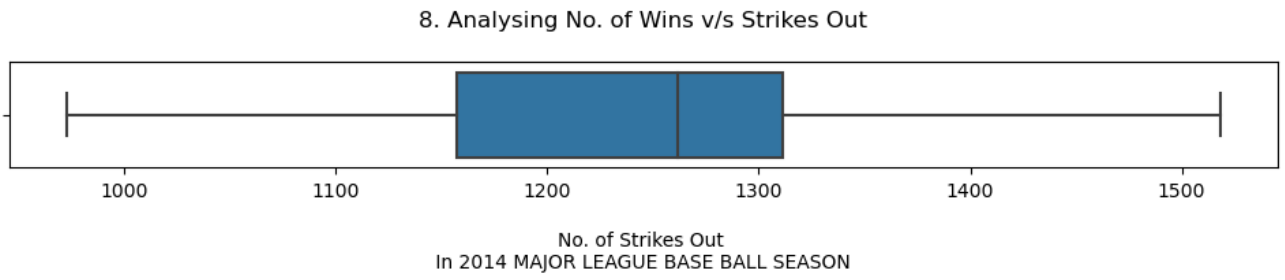


In [135]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n8. Analysing No. of Wins v/s Strike Out\n')
sns.scatterplot (x= 'strike out', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Strikes Out \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Here we find that No. of Strikes Out maximum in between 1100-1350
# we can say that more the no. of strike out LESS CHANCE TO WIN .
```



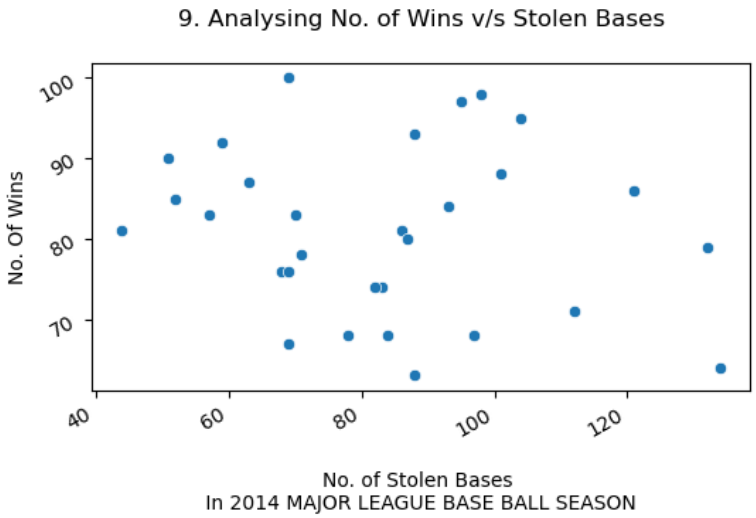
In [136]:
```
#  Analysing StrikeOut with Boxplot ===>>
```

In [137]:
```
plt.figure (figsize = (12,1), facecolor = "white")
plt.title('\n8. Analysing No. of Wins v/s Strikes Out\n')
sns.boxplot (x= 'strike out', data= df,)
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Strikes Out \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
# plt.xticks (rotation = 30, ha= 'right')
# plt.ylabel('No. Of Wins')
# plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Here with the help of BOXPLOT also we find that the maximum no. of strikes out are in-between 1150-1300
# The Highest Strike Out is above 1500
# and the Lowest strike out is betlow 1000
```



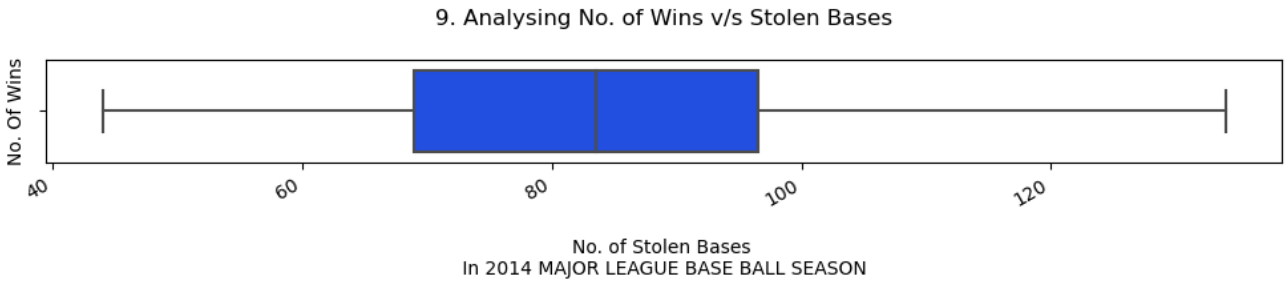
In [138]:
```
#  9) Analysing No. Wins with Stolen Bases ====>>
```

In [139]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n9. Analysing No. of Wins v/s Stolen Bases\n')
sns.scatterplot (x= 'stolen base', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Stolen Bases \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can find there is NO SUCH STRONG RELATION between No. of wins with Stolen Base.
```



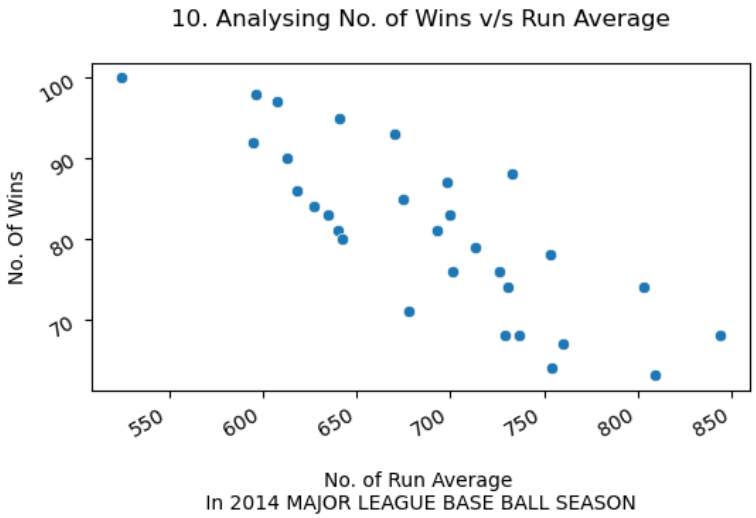
In [140]:
```
#  Analysing Stolen Bases with Boxplot ===>
```

In [141]:
```
plt.figure (figsize = (12,1), facecolor = "white")
plt.title('\n9. Analysing No. of Wins v/s Stolen Bases\n')
sns.boxplot (x= 'stolen base', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Stolen Bases \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here in boxplot we can find that the maximum no. of STOLEN BASES are in-between 70-95
# the Highest No. of stolen base are more then 130
#  and the minimum stolen bases are below 50
```



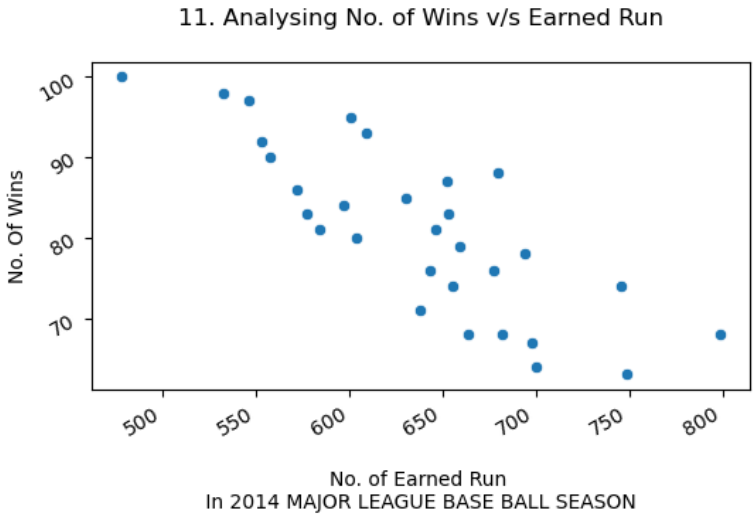
In [142]:
```
#  10) Analysing Run Average with no. wins ====>>
```

In [143]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n10. Analysing No. of Wins v/s Run Average\n')
sns.scatterplot (x= 'run average', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Run Average \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  Here we can find the STRONG NEGATIVE CORRELATION of No. Of Wins with Run
#  as  the run average increasing , the no. of chances to win is DECREASING HERE.
#  the HIGHEST RUN AVERAGE is near about 850.
#  and the MINIMUM RUN AVERAGE is LESS THEN 550.
```



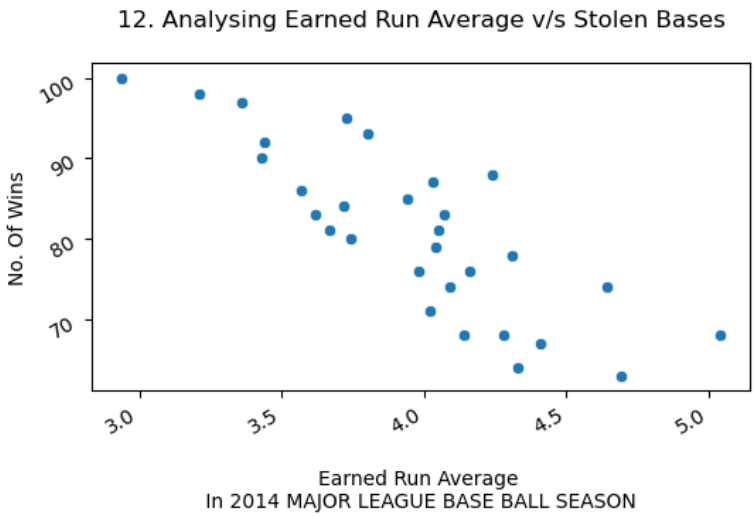
In [144]:
```
#  11) Analysing Earned Run with No. of Wins ===>>>
```

In [145]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n11. Analysing No. of Wins v/s Earned Run\n')
sns.scatterplot (x= 'earned run', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Earned Run \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Here we can see that EARNED RUNS are also NEGALTIVELY CORRELATED with NO. OF WINS
# The Highest No. of EARNED RUN is near about 800
#  and the Lowest No. of EARNED RUN is below 500
```



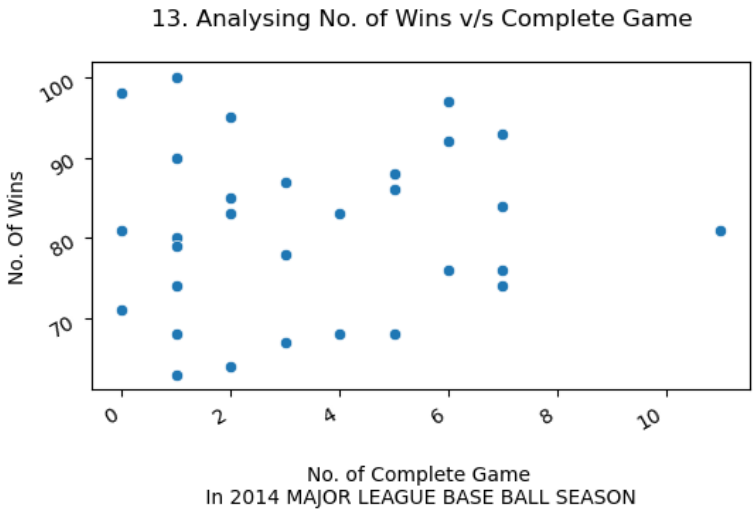
In [146]:
```
#  12) Analysing Earned Run Average with No. Of Wins ====>>
```

In [147]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n12. Analysing Earned Run Average v/s Stolen Bases\n')
sns.scatterplot (x= 'earned run average', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nEarned Run Average \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Earned Run Average is also NEGATIVELY CORRELATED with NO. OF WINS
#  The MAXIMUM & MINIMUM , EARNED RUN AVERAGE IS = above-5.0 & below-3.0
# here we conclude that , higher the earned run average is less the probability to win.
```



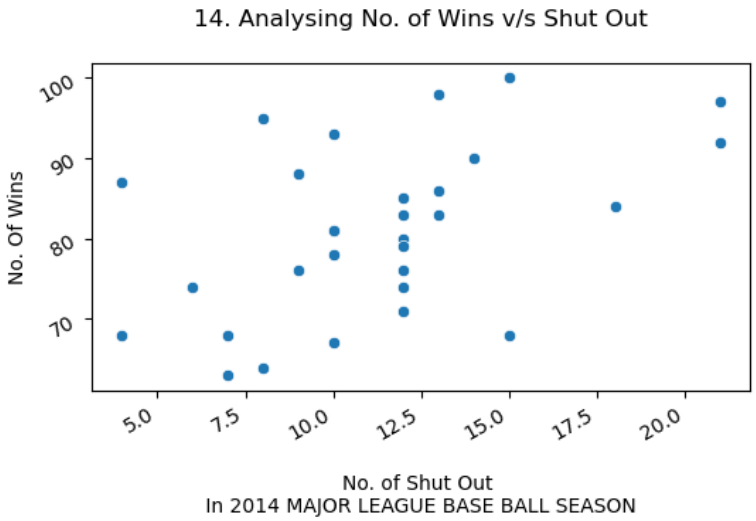
In [148]:
```
# Analysing No. of Wins & Complete Game =====>>
```

In [149]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n13. Analysing No. of Wins v/s Complete Game\n')
sns.scatterplot (x= 'complete game', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Complete Game \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can there no such any relation of Complete Game with No. of Wins.
```



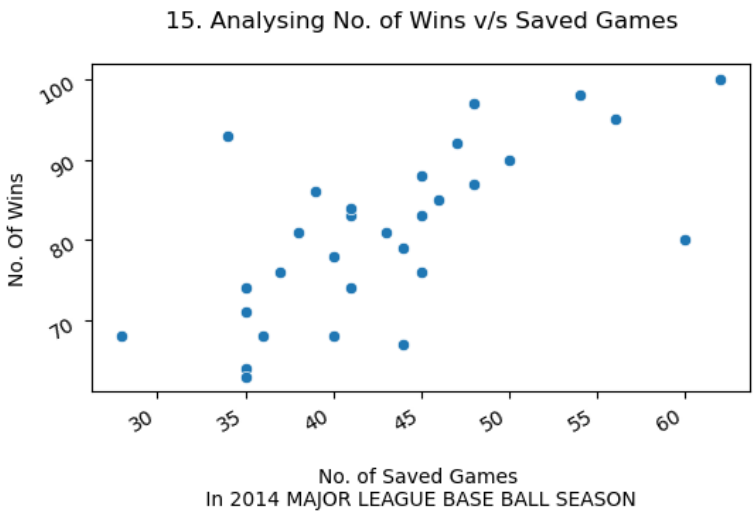
In [150]:
```
#  14) Analysing No. of wins with Shut-out ===>>
```

In [151]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n14. Analysing No. of Wins v/s Shut Out\n')
sns.scatterplot (x= 'shutout', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Shut Out \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

#  here shutout(no. of complete game pitched with no runs aloowed) is don't have have strong positive or negative correla
# here the highest shutouts values are above then 20
#  and the lowest are less then 5.0
# most of the shutout counts are happend in between 7.5-14
```



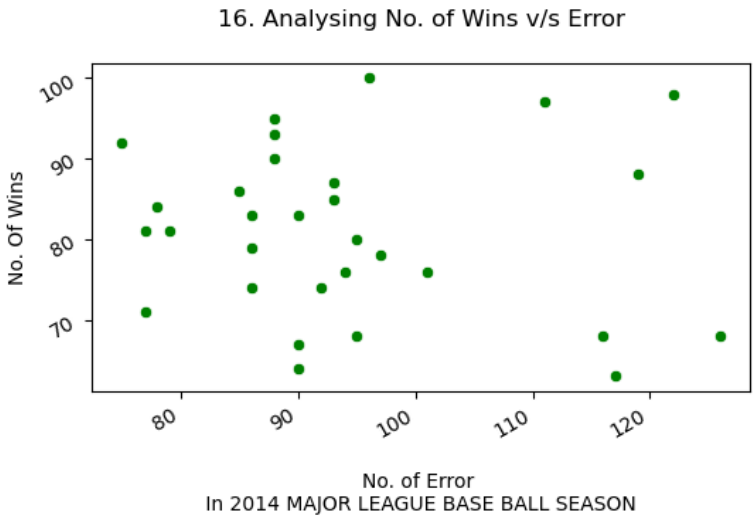
In [152]:
```
#  15) Analysing NO. of Wins with Save Game ====>>>
```

In [153]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n15. Analysing No. of Wins v/s Saved Games\n')
sns.scatterplot (x= 'save', y = 'no. of wins', data= df, palette = "bright")
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Saved Games \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# Offcourse the No. of Wins are STROGLY POSITIVE CORRELATION with the SAVED GAMES (Pitcher who fineshes a game for winni
# Here as the saved games are increased , no. of wins are also increased
```



In [154]:
```
#  16) Analysing No. Of Wins with Errors ====>>
```

In [155]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n16. Analysing No. of Wins v/s Error\n')
sns.scatterplot (x= 'error', y = 'no. of wins', data= df, color='g')
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nNo. of Error \n In 2014 MAJOR LEAGUE BASE BALL SEASON ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('No. Of Wins')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here if we can leave some OUTLIERS then we can find that , the 'errors' are having Slight Neagtive Correlation....
# ...with 'No. of Wins'
#  more the errors less the chances to win.
```



========== UPTO HERE BIVARIATE ANALYSIS IS ALSO COMPLETED
==========

In [ ]:
```

```

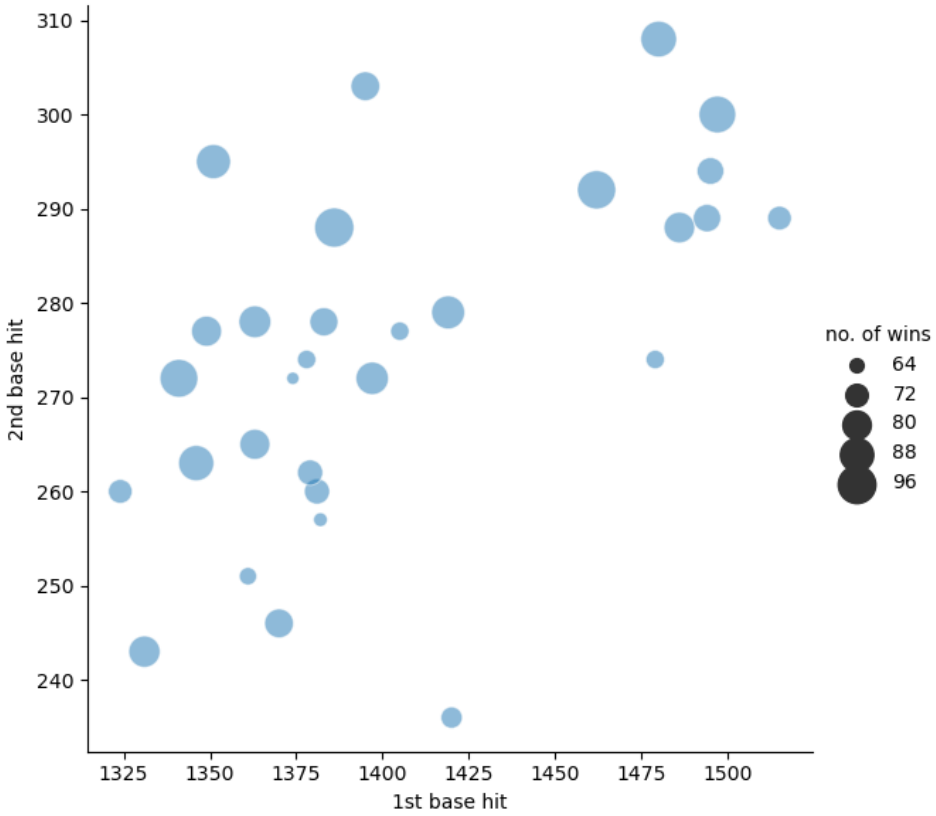
========== ANALYSING THROUGH MULTIVARIATE ANALYSIS ==========

In [156]:
```
#  NOW WE MOVE FORWARD TOWARDS MULTIVARIATE ANALYSIS ====>
```

In [157]:
```
# 1) Analysing No. of wins with 1st base hit & 2nd base hit with No. of wins ====>>
```

In [158]:
```
sns.relplot(x="1st base hit", y="2nd base hit", size="no. of wins",
            sizes=(40, 400), alpha=.5, palette="muted",
            height=6, data=df)

# Here below we have plotted 'relational graph plot', by which we can find relationship b/w '1st & 2nd base hit with no.
# here we can see the Strong POSITIVE CORRELATION of 1st & 2nd base hit with No. of wins.
#  Higher the 1st & 2nd base hit, Higher the probabilty of No. OF WINS.
```

Out[158]: <seaborn.axisgrid.FacetGrid at 0x1892156ec40>

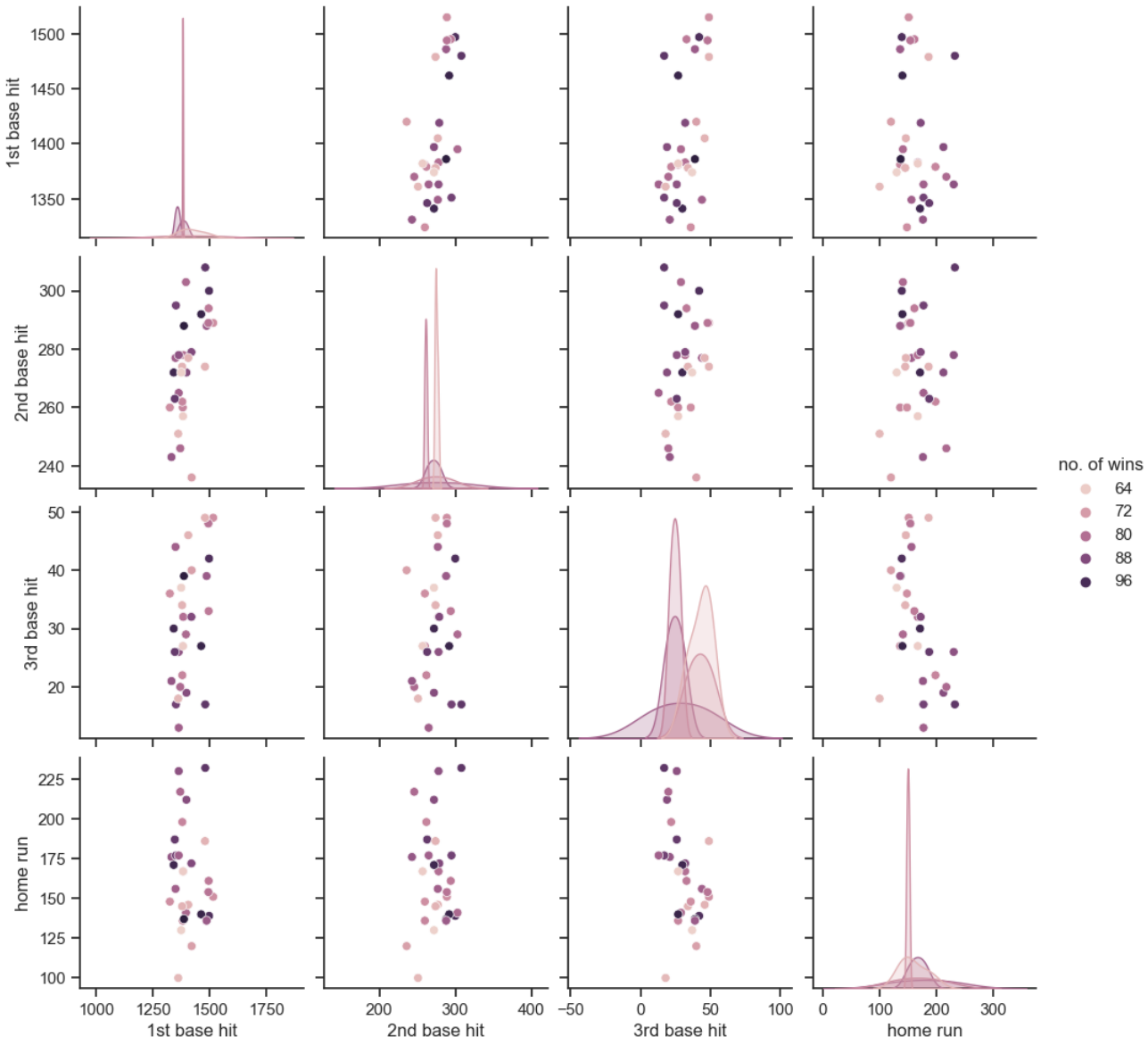


In [159]:
```
# 2) Analysing 1st, 2nd, 3rd base hit & Home run with No. of Wins ====>>>
```

In [160]:
```
df1 = df[['no. of wins','1st base hit', '2nd base hit',
       '3rd base hit', 'home run']]

# here to get clear visuality we are putting 1st, 2nd, 3rd , HomeRun & No. of wins in seprate new 'df1' dataset.
#  it helps us to get clear visulatily of all above mentioned 'independent & dependent variable'
```

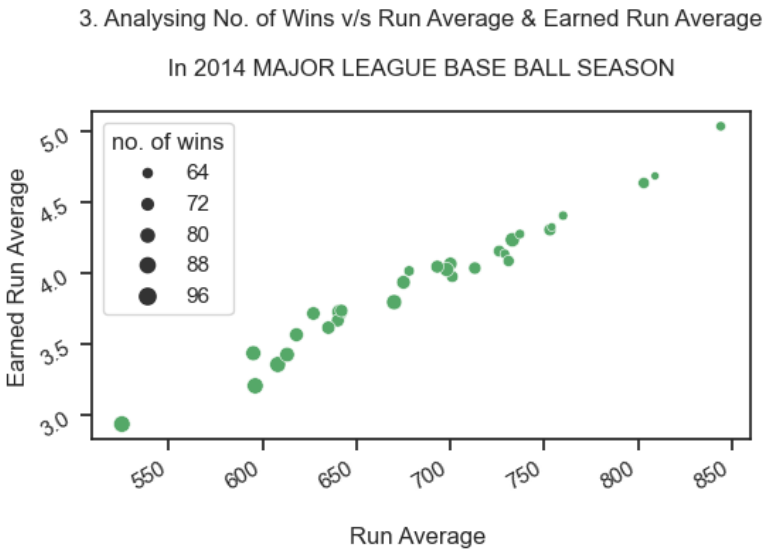
In [161]:
```
sns.set_theme(style="ticks")
sns.pairplot(df1,hue='no. of wins')

#  plotting pairplot for the above mentioned new dataset.
#  plotting with 'No. of wins' as 'hue' points (identifying points), darkest the colour of circle , highest the no. of wi
#                                                                                                    (as shown in legend
# In the given below graphs we can able to find :-
#  1) No. of wins with 1st base hit , 2nd base hit , 3rd base hit & HomeRun
#        with these different combinations we can find the PROBABILITY OF NO. WINS in difeerent conditions.
```

Out[161]: <seaborn.axisgrid.PairGrid at 0x18921380e50>



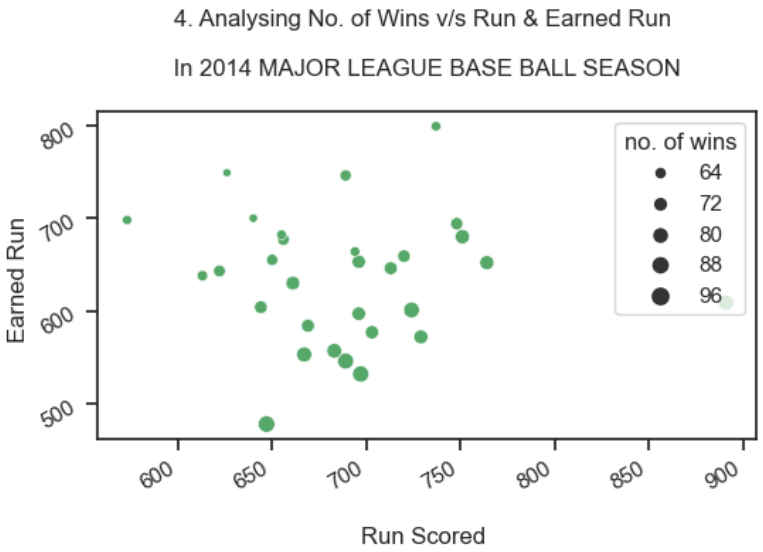
In [162]:
```
#  3) Analysing 'run average', 'Earned run average' with No. of Wins ====>>
```

In [163]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n3. Analysing No. of Wins v/s Run Average & Earned Run Average\n\n In 2014 MAJOR LEAGUE BASE BALL SEASON \n
sns.scatterplot (x= 'run average', y = 'earned run average',size = 'no. of wins', data= df,color='g')
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nRun Average ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('Earned Run Average')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can find that the 'run average' & 'earned run average' is POSITIVELY CORRELATED with each other.
#  but we can see here that 'Npo. of Wins' are NOT POSITIVELY CORELATED with them.
#  more the 'run average' & 'earned run average' is LESS the No. of wins.
# we can clearly see that No. of wins are HIGHER (by the size of circles) in LOWER RANGE's of Run average & Earned Run A
```



In [164]:
```
#  4) Analysing RUN & EARNED RUN with NO. OF WINS ===>>
```

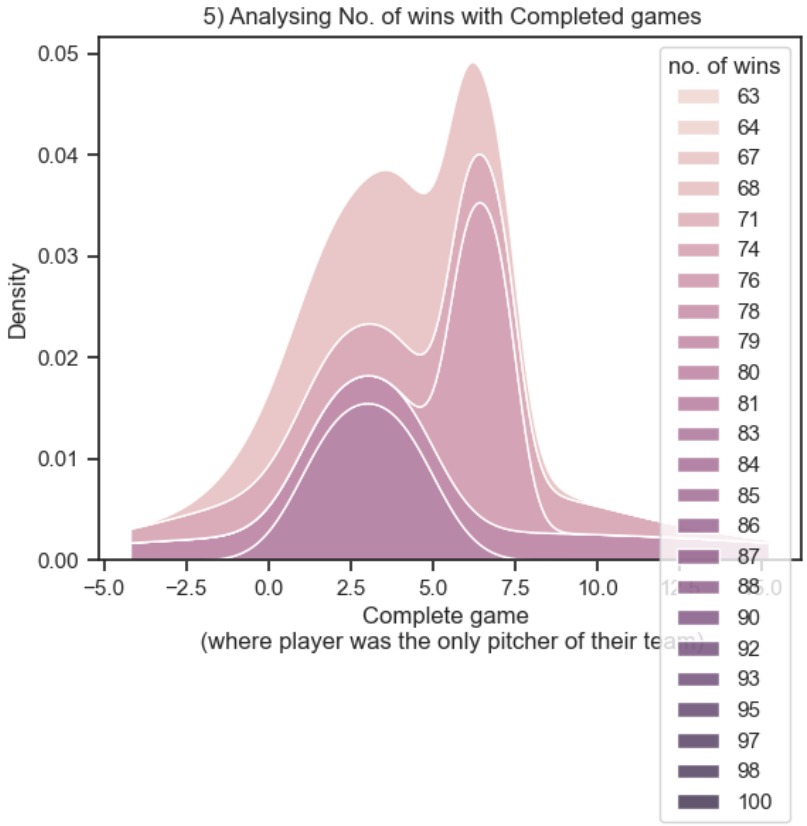
In [165]:
```
plt.figure (figsize = (6,3), facecolor = "white")
plt.title('\n4. Analysing No. of Wins v/s Run & Earned Run \n\n In 2014 MAJOR LEAGUE BASE BALL SEASON \n ')
sns.scatterplot (x= 'run scored', y = 'earned run',size = 'no. of wins', data= df,color='g')
# plt.xticks(rotation=30, ha = 'right')
plt.xlabel('\nRun Scored ')
plt.xticks (rotation = 30, ha= 'right')
plt.ylabel('Earned Run ')
plt.yticks (rotation = 30, ha='right')
# plt.legend(loc= 'center', fontsize=6)
plt.show()

# here we can find that the No. of wins are POISITIVELY CORRELATED with RUN SCORED, because in the following graph we ca
# ...that more the RUN SCORED , NO. OF WINS also Increasing
# but the situation is differen with the EARNED RUN'S , beacuse with eraned run's we can see that, NO. OF WINS ARE LESS
#  ... even higher scores of EARNED RUN'S. so we can say that NO. OF WINS are NEGATIVELY CORELATED with the Earned Run's
```



In [166]:
```
#  5) Analysing No. of wins in complete game ====>>
```

In [167]:
```
plt.title('\n 5) Analysing No. of wins with Completed games')
sns.kdeplot(data=df, x="complete game", hue="no. of wins", multiple="stack")
plt.xlabel('Complete game \n (where player was the only pitcher of their team)')

# here with the help of 'kde plot' we can see that the no. of wins are in the lower range of complete game.
```

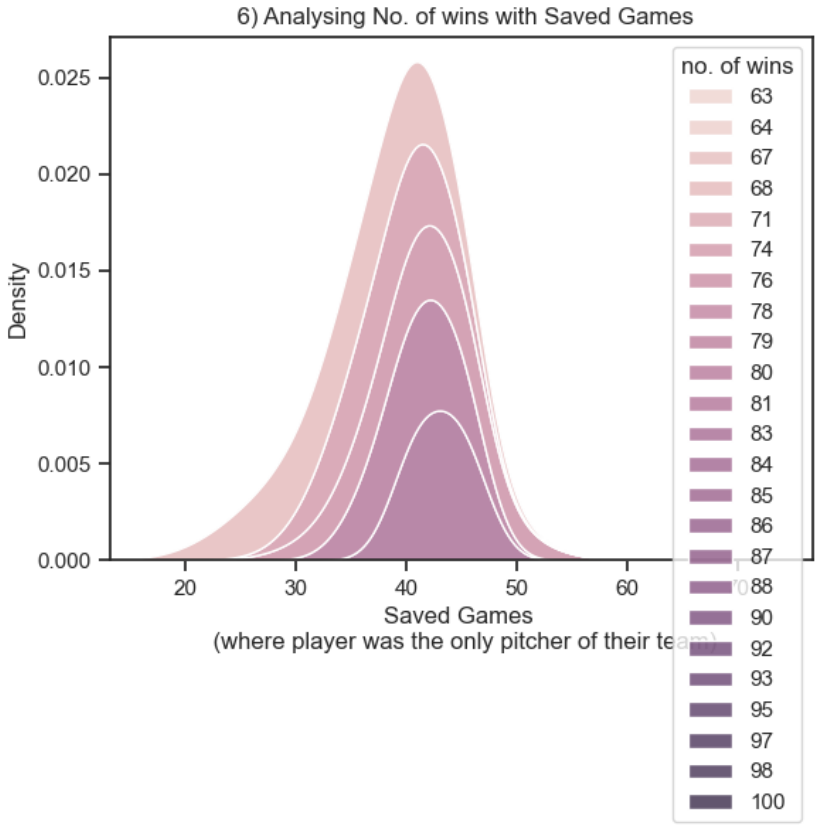
Out[167]:
```
Text(0.5, 0, 'Complete game \n (where player was the only pitcher of their team)')
```



In [168]:
```
#  Analysing No. of wins with Save Game ======>>>
#                            (save game - pitcher who finishesh a game for winning)
```

In [169]:
```
plt.title('\n 6) Analysing No. of wins with Saved Games')
sns.kdeplot(data=df, x="save", hue="no. of wins", multiple="stack")
plt.xlabel('Saved Games \n (where player was the only pitcher of their team)')

# here with the help of 'kde plot' we can see that the no. of wins are Highest in the 40-45 range of 'saved game'.
```

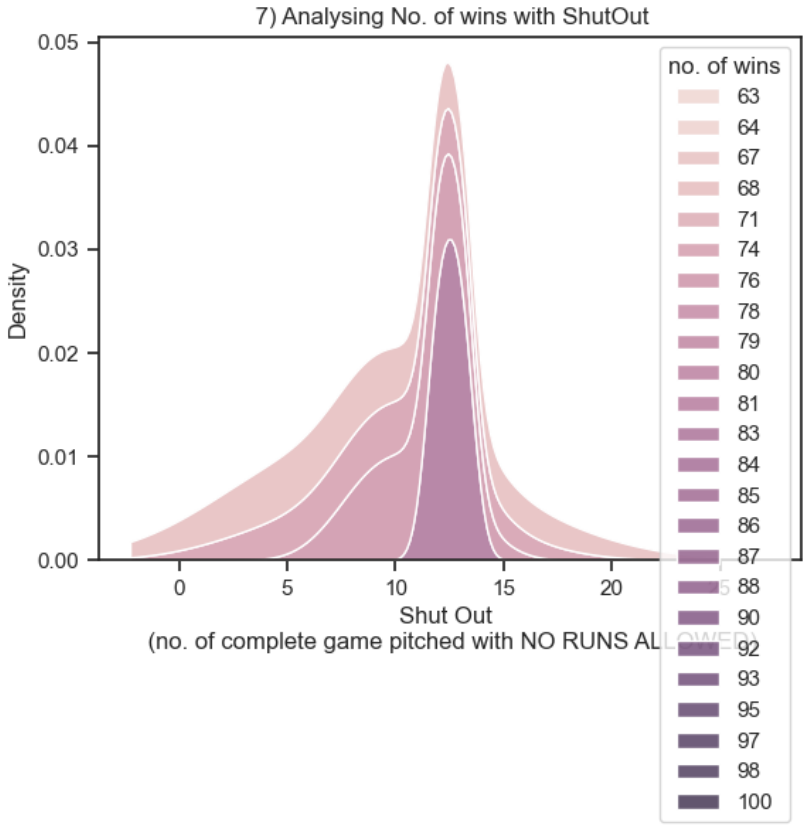
Out[169]:
```
Text(0.5, 0, 'Saved Games \n (where player was the only pitcher of their team)')
```



In [170]:
```
#  7) Analysing No. of Wins with 'ShutOut' ======>>>
#                                    (shutout is no. of complete game pitched with NO RUN'S ALLOWED)
```

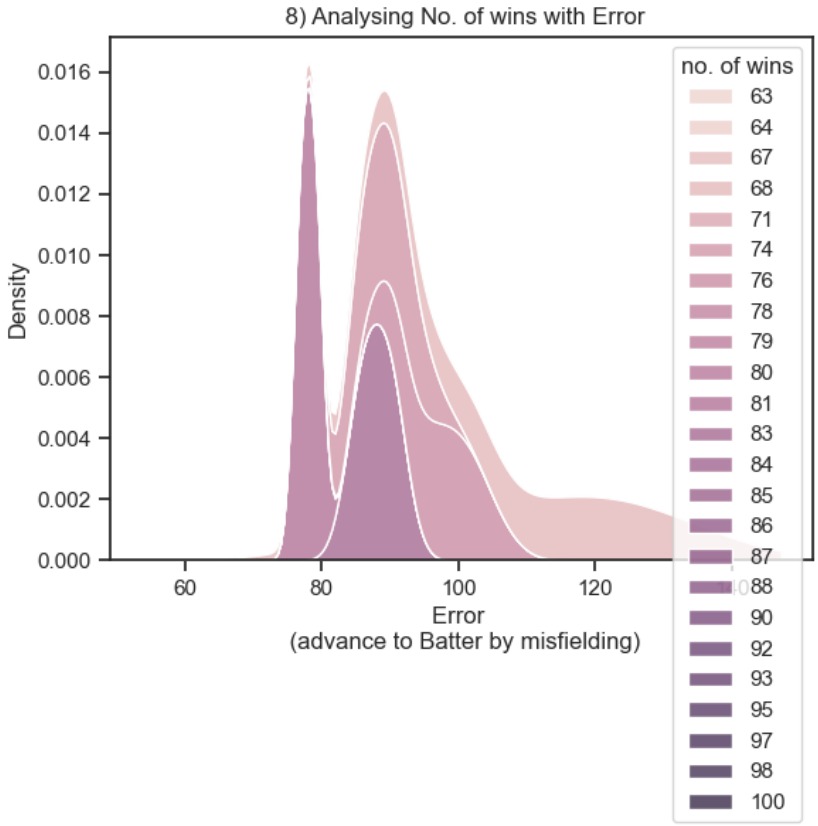
In [171]:
```
plt.title('\n 7) Analysing No. of wins with ShutOut')
sns.kdeplot(data=df, x="shutout", hue="no. of wins", multiple="stack")
plt.xlabel('Shut Out \n (no. of complete game pitched with NO RUNS ALLOWED)')

# here with the help of 'kde plot' we can see that the Highest no. of wins are in the 10-15 range of 'shut out'.
```

Out[171]:
```
Text(0.5, 0, 'Shut Out \n (no. of complete game pitched with NO RUNS ALLOWED)')
```



In [172]:
```
plt.title('\n 8) Analysing No. of wins with Error')
sns.kdeplot(data=df, x="error", hue="no. of wins", multiple="stack")
plt.xlabel('Error \n (advance to Batter by misfielding)')

# here with the help of 'kde plot' we can see that the Highest no. of wins are in the 10-15 range of 'shut out'.
```

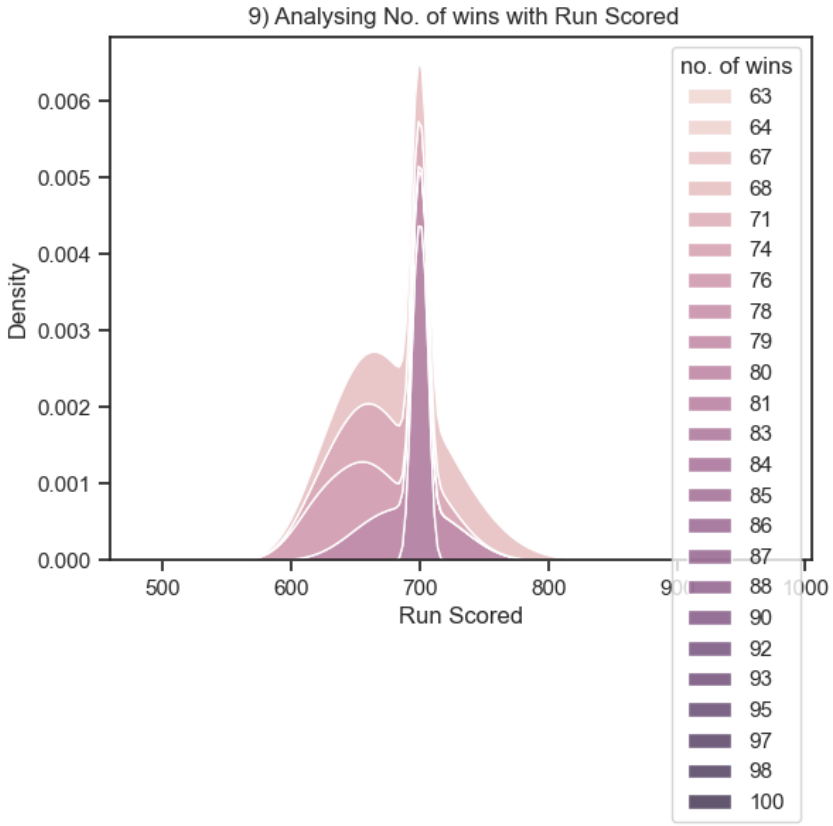
Out[172]:
```
Text(0.5, 0, 'Error \n (advance to Batter by misfielding)')
```



In [173]:
```
# 9) Analysing No. of Wins with Run Scored =====>>
```

In [174]:
```
plt.title('\n 9) Analysing No. of wins with Run Scored')
sns.kdeplot(data=df, x="run scored", hue="no. of wins", multiple="stack")
plt.xlabel('Run Scored')

# here with the help of 'kde plot' we can see that the HIGHEST WINNING A GAME SCORE is HIGHER with the RUN SCORE OF 700.
```

Out[174]:
```
Text(0.5, 0, 'Run Scored')
```



========== UPTO HERE COMPLETED UNIVARIATE / BIVARIATE / MUTLIVARIATE ANALYSIS OF DATA ==========

In [ ]:
```

```

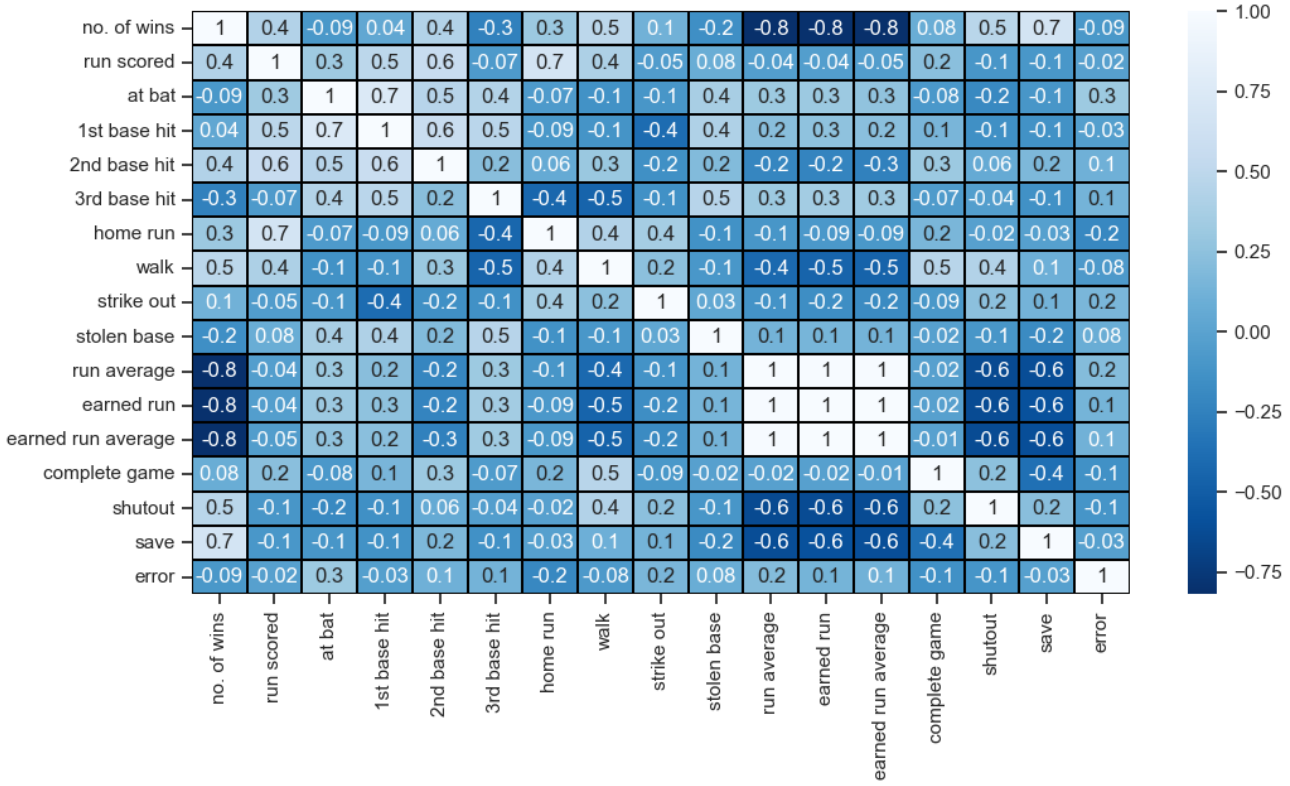
========== FINDING CORRELATION IN DATASET ==========

In [175]:
```
cor = df.corr()
cor

# non graphically finding correlation, here we can see that it is difficult to understand this kind of correlation....
# ....so further we find the correlation graphically by HEAT MAP.
```

Out[175]:

	no. of wins	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned run	earned run average
no. of wins	1.000000	0.430751	-0.087947	0.037612	0.427797	-0.251118	0.307407	0.484342	0.111850	-0.157234	-0.812952	-0.809435	-0.819600
run scored	0.430751	1.000000	0.319464	0.482856	0.560084	-0.070072	0.671283	0.402452	-0.054726	0.081367	-0.041623	-0.041245	-0.049281
at bat	-0.087947	0.319464	1.000000	0.739122	0.453370	0.435422	-0.066983	-0.136414	-0.106022	0.372618	0.316010	0.309686	0.255551
1st base hit	0.037612	0.482856	0.739122	1.000000	0.566847	0.478694	-0.090855	-0.118281	-0.398830	0.413444	0.224324	0.252489	0.231172
2nd base hit	0.427797	0.560084	0.453370	0.566847	1.000000	0.220490	0.056292	0.302700	-0.150752	0.195027	-0.218160	-0.235531	-0.254854
3rd base hit	-0.251118	-0.070072	0.435422	0.478694	0.220490	1.000000	-0.430915	-0.454949	-0.141196	0.457437	0.314125	0.340225	0.330951
home run	0.307407	0.671283	-0.066983	-0.090855	0.056292	-0.430915	1.000000	0.425691	0.359923	-0.136567	-0.103903	-0.085922	-0.090917
walk	0.484342	0.402452	-0.136414	-0.118281	0.302700	-0.454949	0.425691	1.000000	0.233652	-0.098347	-0.416445	-0.452663	-0.459832
strike out	0.111850	-0.054726	-0.106022	-0.398830	-0.150752	-0.141196	0.359923	0.233652	1.000000	0.030968	-0.129745	-0.161612	-0.180368
stolen base	-0.157234	0.081367	0.372618	0.413444	0.195027	0.457437	-0.136567	-0.098347	0.030968	1.000000	0.132290	0.143068	0.126063
run average	-0.812952	-0.041623	0.316010	0.224324	-0.218160	0.314125	-0.103903	-0.416445	-0.129745	0.132290	1.000000	0.991018	0.986674
earned run	-0.809435	-0.041245	0.309686	0.252489	-0.235531	0.340225	-0.085922	-0.452663	-0.161612	0.143068	0.991018	1.000000	0.997248
earned run average	-0.819600	-0.049281	0.255551	0.231172	-0.254854	0.330951	-0.090917	-0.459832	-0.180368	0.126063	0.986674	0.997248	1.000000
complete game	0.080533	0.232042	-0.080876	0.147955	0.306675	-0.065898	0.156502	0.462478	-0.093418	-0.020783	-0.016659	-0.020221	-0.009856
shutout	0.471805	-0.103274	-0.197321	-0.145559	0.057998	-0.041396	-0.019119	0.426004	0.237721	-0.106563	-0.636862	-0.630192	-0.630833
save	0.666530	-0.096380	-0.106367	-0.130371	0.171576	-0.142370	-0.028540	0.099445	0.126297	-0.183418	-0.616224	-0.589663	-0.607005
error	-0.089485	-0.023262	0.316743	-0.033173	0.105754	0.126678	-0.207597	-0.075685	0.155133	0.079149	0.198996	0.136921	0.113137

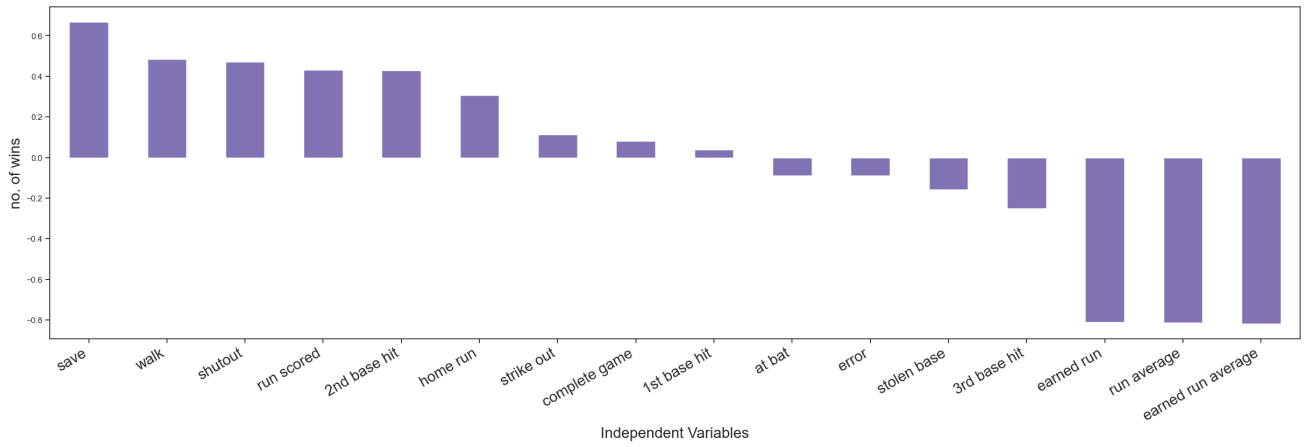
In [176]:
```
plt.figure (figsize = (12,6), facecolor = "white")
sns.heatmap(df.corr(),linewidth=0.1,fmt="0.1g",linecolor="black",annot=True,cmap="Blues_r")
plt.yticks(rotation=0);
plt.show()


#  Here in the following heat map , it is very much clear the 'run average', 'earned run' & 'earned run average' all thre
# ....highly correleted with each other.
# the 'run average', 'earned run' & 'earned run average' all three are STRONGLY NEGATIVE CORRELETED with NO. OF WINS...
# .....with the value of (-0.8)
```



In [177]:
```
cor['no. of wins'].sort_values(ascending=False)

#  here in the following table :
# 1) highly positive correleted column with 'No. of wins' is - 'save'
#  2) highly negative correleted column with 'No. of wins' is - 'earned run', 'earned run average', & 'run average'
```

Out[177]:
```
no. of wins           1.000000
save                  0.666530
walk                  0.484342
shutout               0.471805
run scored            0.430751
2nd base hit          0.427797
home run              0.307407
strike out            0.111850
complete game         0.080533
1st base hit          0.037612
at bat               -0.087947
error                -0.089485
stolen base          -0.157234
3rd base hit         -0.251118
earned run           -0.809435
run average          -0.812952
earned run average   -0.819600
Name: no. of wins, dtype: float64
```

In [178]:
```
plt.figure(figsize=(30,8))
df.corr()['no. of wins'].sort_values(ascending=False).drop(['no. of wins']).plot(kind='bar',color="m")
plt.xlabel('Independent Variables',fontsize=20)
plt.xticks(rotation=30,ha='right',fontsize=20)
plt.ylabel('no. of wins',fontsize =20)
plt.title=("Correlation with No. of wins")
plt.show()

# As above we also mentioned that 'earned run' 'run average' 'earned run average',these 3 columns are most NEGATIVELY...
# ...CORRELATED with the TARGET COLUMN
#
```



In [ ]:
```

```

========== CHECKING FOR OUTLIERS

==========

In [179]:
```
df.describe()
# here as we can see in the above table, we see a huge difference between 75% & Max of some columns like :-
# "run scored","3rd base hit", "strike out", "stolen base", "complete game", "shutout", " save" & "error"
#  due to which we can assume that there may presence of outliers, so we have to check this with "BOXPLOT METHOD"

# Here we can also observed that there is huge difference between 75 Prcentile & MAX in above mentioned columns,
#  but if outliers are present then STNDARD DEVIATION should also be HIGH, but we can see that the standard deviation is
#  ...also high in some of the columns.
#  so before move ahead we can sure about the presence of outliers by using BOXPLOT METHEOD.
```

Out[179]:

	no. of wins	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.00000	30.000000	30.000000	30.00(
mean	80.966667	688.233333	5516.266667	1403.533333	274.733333	31.300000	163.633333	469.100000	1248.20000	83.500000	688.233333	635.83
std	10.453455	58.761754	70.467372	57.140923	18.095405	10.452355	31.823309	57.053725	103.75947	22.815225	72.108005	70.14(
min	63.000000	573.000000	5385.000000	1324.000000	236.000000	13.000000	100.000000	375.000000	973.00000	44.000000	525.000000	478.00(
25%	74.000000	651.250000	5464.000000	1363.000000	262.250000	23.000000	140.250000	428.250000	1157.50000	69.000000	636.250000	587.25(
50%	81.000000	689.000000	5510.000000	1382.500000	275.500000	31.000000	158.500000	473.000000	1261.50000	83.500000	695.500000	644.50(
75%	87.750000	718.250000	5570.000000	1451.500000	288.750000	39.000000	177.000000	501.250000	1311.50000	96.500000	732.500000	679.25(
max	100.000000	891.000000	5649.000000	1515.000000	308.000000	49.000000	232.000000	570.000000	1518.00000	134.000000	844.000000	799.00(

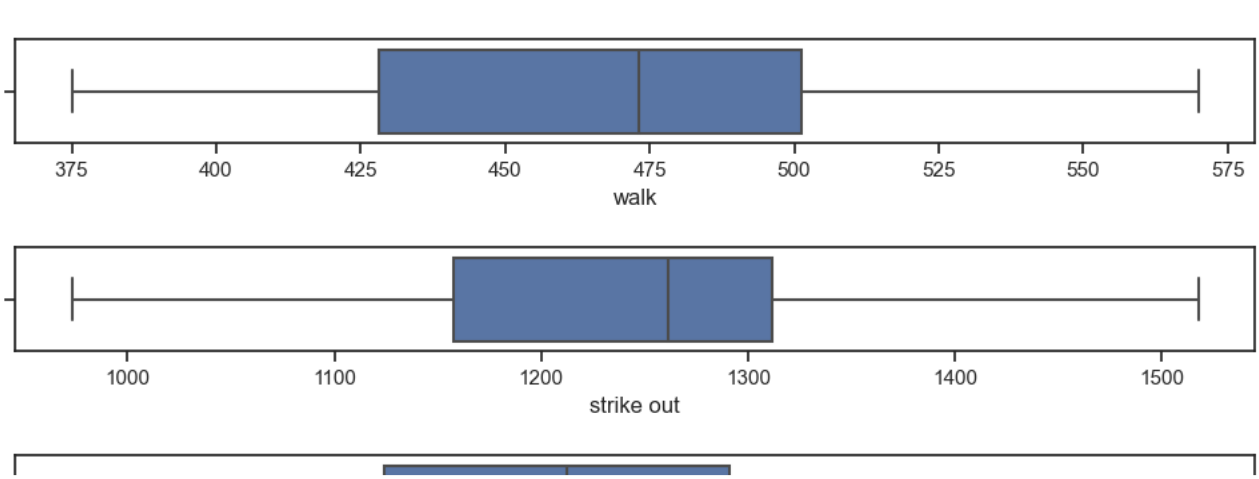
In [180]:
```
df.columns
```

Out[180]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

In [181]:
```
df.columns.nunique()
# there are 17 columns are present in our dataset, and we have to check each of them.
```

Out[181]:
```
17
```

In [182]:
```
for i in df.columns[0:17]:
    plt.figure (figsize = (12,1), facecolor = "white")
    sns.boxplot(x=i,data=df)
    plt.show()

#     here below we can find the outliers for all the cloumns by using boxplot.
#  and we are found outliers in :
#  run scored, earned run average, shut out, save & error.
# so out of 17 columns we found OUTLIERS IN 5 COLUMNS , now we have to remove those outliers from out dataset.
```



In [ ]:
```
```

========== REMOVING OF OUTLIERS BY USING Z-SCORE METHOD ==========

In [183]:
```
#  we can not remove outliers from out TARGET COLUMN, so first we have to seprate target column first.
# For this first we need to identify the ZSCORE VALUES, for which we have to import some libraries.
```

In [184]:
```
from scipy.stats import zscore
```

In [185]:
```
z = np.abs(zscore(df))
z.head(5)

# by applying 'abs' (absolute method), we are getting all the entries whose z-score value is positive side
#  Ideally we can call the OUTLIERS whos ZSCORE VALUE is LESS THEN 3 AND MORE THEN 3
# so we have to remove all the data whose ZSCORE >3 & <3
#  below here we apllying "abs" i.e absolute method it returns us the all zscore values greater then 3
#  so we just need to remove lesserr then 3 zscore values.
```

Out[185]:

	no. of wins	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned run	earned run average	complete game	shut
0	1.365409	0.619078	0.847731	1.663685	1.420173	1.041193	0.787299	1.534902	2.697630	0.913883	0.666234	0.505110	0.506955	0.539806	0.814
1	0.197838	0.134432	0.711094	0.970681	0.127403	1.235809	0.243967	0.536592	0.154878	0.601826	0.165971	0.248930	0.254598	0.539806	0.172
2	0.003243	0.332906	1.115233	0.151891	1.588795	0.223808	0.723377	1.139144	0.893982	0.111449	0.680339	0.751623	0.641347	2.772641	0.320
3	0.483244	1.146419	0.241522	0.401088	0.828122	0.418423	0.883181	1.160536	0.168602	0.690985	0.180076	0.103922	0.053010	1.300442	0.567
4	0.677839	0.013270	1.280738	1.984081	0.801892	1.722347	0.403770	0.251360	0.105866	0.022290	1.618803	1.597501	1.531318	1.300442	0.172

In [186]:
```
threshold = 3
print(np.where(z>3))
```

```
(array([5], dtype=int64), array([1], dtype=int64))
```

In [187]:
```
# here above we found only 1 outliers,  whose z-score is more then > 3
# i.e means we are having 1 outlier still present in our dataset, and we have to remove those outliers
```

In [188]:
```
df_new = df[(z<3).all(axis=1)]
df_new.shape
df_new
```

Out[188]:

	no. of wins	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned run	earned run average	complete game	shutout	save	error
0	95	724	5575	1497	300	42	139	383	973	104	641	601	3.73	2	8	56	88
1	83	696	5467	1349	277	44	156	439	1264	70	700	653	4.07	2	12	45	86
2	81	669	5439	1395	303	29	141	533	1157	86	640	584	3.67	11	10	38	79
3	76	622	5533	1381	260	27	136	404	1231	68	701	643	3.98	7	9	37	101
4	74	689	5605	1515	289	49	151	455	1259	83	803	746	4.64	7	12	35	86
6	87	764	5567	1397	272	19	212	554	1227	63	698	652	4.03	3	4	48	93
7	81	713	5485	1370	246	20	217	418	1331	44	693	646	4.05	0	10	43	77
8	80	644	5485	1383	278	32	167	436	1310	87	642	604	3.74	1	12	60	95
9	78	748	5640	1495	294	33	161	478	1148	71	753	694	4.31	3	10	40	97
10	88	751	5511	1419	279	32	172	503	1233	101	733	680	4.24	5	9	45	119
11	86	729	5459	1363	278	26	230	486	1392	121	618	572	3.57	5	13	39	85
12	85	661	5417	1331	243	21	176	435	1150	52	675	630	3.94	2	12	46	93
13	76	656	5544	1379	262	22	198	478	1336	69	726	677	4.16	6	12	45	94
14	68	694	5600	1405	277	46	146	475	1119	78	729	664	4.14	5	15	28	126
15	100	647	5484	1386	288	39	137	506	1267	69	525	478	2.94	1	15	62	96
16	98	697	5631	1462	292	27	140	461	1322	98	596	532	3.21	0	13	54	122
17	97	689	5491	1341	272	30	171	567	1518	95	608	546	3.36	6	21	48	111
18	68	655	5480	1378	274	34	145	412	1299	84	737	682	4.28	1	7	40	116
19	64	640	5571	1382	257	27	167	496	1255	134	754	700	4.33	2	8	35	90
20	90	683	5527	1351	295	17	177	488	1290	51	613	557	3.43	1	14	50	88
21	83	703	5428	1363	265	13	177	539	1344	57	635	577	3.62	4	13	41	90
22	71	613	5463	1420	236	40	120	375	1150	112	678	638	4.02	0	12	35	77
23	67	573	5420	1361	251	18	100	471	1107	69	760	698	4.41	3	10	44	90
24	63	626	5529	1374	272	37	130	387	1274	88	809	749	4.69	1	7	35	117
25	92	667	5385	1346	263	26	187	563	1258	59	595	553	3.44	6	21	47	75
26	84	696	5565	1486	288	39	136	457	1159	93	627	597	3.72	7	18	41	78
27	79	720	5649	1494	289	48	154	490	1312	132	713	659	4.04	1	12	44	86
28	74	650	5457	1324	260	36	148	426	1327	82	731	655	4.09	1	6	41	92
29	68	737	5572	1479	274	49	186	388	1283	97	844	799	5.04	4	4	36	95

In [189]:
```
df.shape
```

Out[189]: (30, 17)

In [190]:
```
df_new.shape
```

Out[190]: (29, 17)

In [191]:
```
# here you can see our rows are reduced from 30 to 29, that means 1 Outlier is removed from our dataset.
```

In [ ]:
```

```

==========CHECKING REMOVAL OF OUTLIERS BY BOXPLOT (COMPARING 'df' & 'df_new') ==========

In [192]:
```
df_new.columns
```

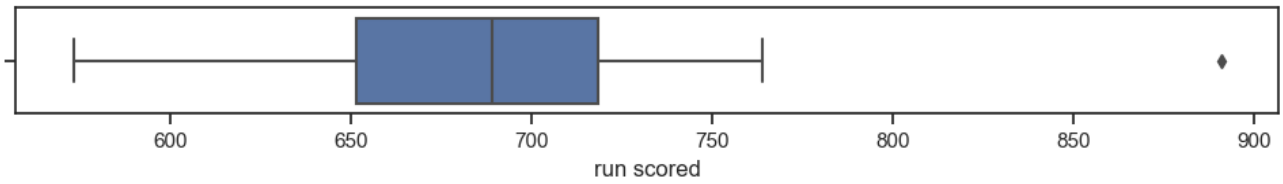
Out[192]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

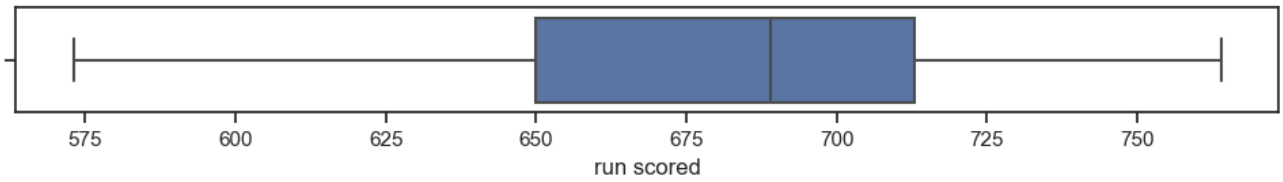
In [193]:
```
df_new.columns.nunique()
# here also same columns of earlier 'df' dataset is present
```

Out[193]: 17

In [194]:
```
# now in the earlier findings we found that there are may presence of outliers in 5 columns....
# ....i.e (run scored, earned run average, shut out, save & error)
#  now we have to check out of these columns which outlier hasbeen removed from the dataset.
```

In [195]:
```
plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='run scored',data=df)
plt.show()

# it is the EARLIER (df dataset) PRESENCE OF OUTLIERS

plt.figure (figsize = (12,1), facecolor = "white")
sns.boxplot(x='run scored',data=df_new)
plt.show()

# outliers are succesfully removed.
# it is the Newer (df_new dataset)  OUTLIERS ARE REMOVED.
# So as we can see , the 1 outlier which is removed above by Z-SCORE METHOD is from 'RUN SCORED' column.
```





In [ ]:

CHECKING SKEWNESS

==========>>>

In [196]:
```
# the skewness shows the distribution of data, if the data is widely skewed that means it is not good for our model.
#  ideal range of skewness is ( -0.5 to +0.5)
# We can't remove skewness from our Target Column
```

In [197]:
```
df_new.skew()
#  here we can see the slightly skewness in the column- 'complete game' & 'error'.
#  we are having very limited data in our dataset, & these columns are very slightly skewed, which is acceptable...
# ....so we ignore this minute skewness and can say out dataset is not skewed.
```

Out[197]:
```
no. of wins           0.119013
run scored           -0.215364
at bat                0.169573
1st base hit          0.783772
2nd base hit         -0.335304
3rd base hit          0.090124
home run              0.450862
walk                  0.151193
strike out           -0.233815
stolen base           0.494966
run average           0.018155
earned run            0.018461
earned run average    0.016693
complete game         0.854980
shutout               0.526943
save                  0.627480
error                 0.840271
dtype: float64
```

In [ ]:

DIVIDING DATA INTO INDEPENDENT & TARGET VARIABLE

==========

In [198]:
```
df_new.columns
```

Out[198]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

In [199]:
```
x = df_new[['run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error']]
```

In [200]:
```
y = df_new[['no. of wins']]
```

In [201]:
```
x.shape
```

Out[201]:
```
(29, 16)
```

In [202]:
```
y.shape
```

Out[202]:
```
(29, 1)
```

In [203]:
```
# here above we seprates independent and dependent columns successfully.
```

In [ ]:

APPLYING SCALING TECHNIQUES

==========

In [204]:
```
# here we need to apply scaling techniques on our dataset,by scaling techniques we normalise the values.
#  we can't apply SCALING TECHNIQUES on TARGET VARIABLE
#  to aplly scaling techinuque we need to import some libraries first.
```

In [205]:
```
from sklearn.preprocessing import StandardScaler
```

In [206]:
```
st = StandardScaler()
```

In [207]:
```
x = st.fit_transform(x)
x
```

Out[207]:
```
array([[ 0.95939835,  0.83008422,  1.73830631,  1.55653766,  1.01084549,
        -0.76586314, -1.53635899, -2.72762331,  0.9059529 , -0.66451353,
        -0.51114812, -0.51138828, -0.50148589, -0.81326172,  1.64315663,
        -0.47109143],
       [ 0.33114717, -0.70283074, -0.93869788,  0.20117132,  1.20891656,
        -0.18138864, -0.49502051,  0.12189552, -0.58531515,  0.15463823,
         0.23216082,  0.23895226, -0.50148589,  0.15929869,  0.21100645,
        -0.61489829],
       [-0.27466646, -1.10025314, -0.10665604,  1.73332457, -0.2766165 ,
        -0.69710144,  1.25294051, -0.9258657 ,  0.11645805, -0.67839746,
        -0.75415297, -0.64380131,  2.85461197, -0.32698152, -0.70036184,
        -1.1182223 ],
       [-1.32923093,  0.23395062, -0.35988616, -0.8006212 , -0.47468758,
        -0.8690057 , -1.14585706, -0.20124579, -0.6730368 ,  0.16852216,
         0.08921679,  0.04033271,  1.36301292, -0.57012162, -0.83055731,
         0.46365315],
       [ 0.17408438,  1.25589393,  2.0638879 ,  0.90831897,  1.70409425,
        -0.35329291, -0.19749523,  0.07293472, -0.01512442,  1.58468284,
         1.56154027,  1.4968761 ,  1.36301292,  0.15929869, -1.09094826,
```

In [208]:
```
xf = pd.DataFrame(data=x)
print(xf)

# here we get our dataset (xf) after applying SCALING TECHING (STANDARD SCALER)
```

	0	1	2	3	4	5	6
0	0.959398	0.830084	1.738306	1.556538	1.010845	-0.765863	-1.536359
1	0.331147	-0.702831	-0.938698	0.201171	1.208917	-0.181389	-0.495021
2	-0.274666	-1.100253	-0.106656	1.733325	-0.276617	-0.697101	1.252941
3	-1.329231	0.233951	-0.359886	-0.800621	-0.474688	-0.869006	-1.145857
4	0.174084	1.255894	2.063888	0.908319	1.704094	-0.353293	-0.197495
5	1.856900	0.716535	-0.070480	-0.093474	-1.266972	1.743939	1.643442
6	0.712585	-0.447345	-0.558853	-1.625627	-1.167936	1.915843	-0.885522
7	-0.835605	-0.447345	-0.323710	0.260100	0.020490	0.196801	-0.550807
8	1.497899	1.752672	1.702131	1.202964	0.119526	-0.009484	0.230197
9	1.565212	-0.078310	0.327453	0.319029	0.020490	0.368705	0.695081
10	1.071586	-0.816380	-0.685468	0.260100	-0.573723	2.362794	0.378960
11	-0.454167	-1.412514	-1.264279	-1.802414	-1.068901	0.506228	-0.569402
12	-0.566355	0.390081	-0.396062	-0.682763	-0.969865	1.262607	0.230197
13	0.286272	1.184926	0.074223	0.201171	1.406988	-0.525197	0.174411
14	-0.768292	-0.461539	-0.269447	0.849390	0.713739	-0.834625	0.750867
15	0.353585	1.624929	1.105231	1.085106	-0.474688	-0.731482	-0.085923
16	0.174084	-0.362183	-1.083401	-0.093474	-0.177581	0.334324	1.885182
17	-0.588792	-0.518313	-0.414150	0.024384	0.218561	-0.559578	-0.997094
18	-0.925355	0.773310	-0.341798	-0.977408	-0.474688	0.196801	0.564913
19	0.039459	0.148789	-0.902522	1.261893	-1.465043	0.540609	0.416151
20	0.488210	-1.256383	-0.685468	-0.505976	-1.861185	0.540609	1.364512
21	-1.531169	-0.759605	0.345541	-2.214917	0.812774	-1.419099	-1.685122
22	-2.428670	-1.369933	-0.721643	-1.330982	-1.366007	-2.106716	0.100030
23	-1.239481	0.177176	-0.486501	-0.093474	0.515668	-1.075291	-1.461978
24	-0.319542	-1.866711	-0.992961	-0.623834	-0.573723	0.884418	1.810800
25	0.331147	0.688148	1.539340	0.849390	0.713739	-0.869006	-0.160305
26	0.869648	1.880415	1.684043	0.908319	1.605059	-0.250150	0.453341
27	-0.700980	-0.844767	-1.390895	-0.800621	0.416632	-0.456435	-0.736760
28	1.251086	0.787503	1.412725	0.024384	1.704094	0.850037	-1.443382

	7	8	9	10	11	12	13
0	-2.727623	0.905953	-0.664514	-0.511148	-0.511388	-0.501486	-0.813262
1	0.121896	-0.585315	0.154638	0.232161	0.238952	-0.501486	0.159299
2	-0.925866	0.116458	-0.678397	-0.754153	-0.643801	2.854612	-0.326982
3	-0.201246	-0.673037	0.168522	0.089217	0.040333	1.363013	-0.570122
4	0.072935	-0.015124	1.584683	1.561540	1.496876	1.363013	0.159299
5	-0.240414	-0.892341	0.126870	0.217866	0.150677	-0.128586	-1.785822
6	0.777970	-1.725697	0.057451	0.132100	0.194815	-1.247285	-0.326982
7	0.572335	0.160319	-0.650630	-0.468265	-0.489319	-0.874386	0.159299
8	-1.013995	-0.541454	0.890486	0.818231	0.768604	-0.128586	-0.326982
9	-0.181661	0.774370	0.612808	0.618110	0.614123	0.617213	-0.570122
10	1.375292	1.651587	-0.983844	-0.925686	-0.864490	0.617213	0.402439
11	-0.994411	-1.374810	-0.192460	-0.096610	-0.047943	-0.501486	0.159299
12	0.826931	-0.629176	0.515620	0.575226	0.437572	0.990113	0.159299
13	-1.297968	-0.234429	0.557272	0.389399	0.393434	0.617213	0.888719
14	0.151272	-0.629176	-2.275049	-2.269360	-2.254827	-0.874386	0.888719
15	0.689841	0.642788	-1.289290	-1.497462	-1.658968	-1.247285	0.402439
16	2.609104	0.511205	-1.122683	-1.297340	-1.327935	0.990113	2.347560
17	0.464621	0.028736	0.668344	0.646698	0.702398	-0.874386	-1.056402
18	0.033766	2.221778	0.904370	0.903998	0.812742	-0.501486	-0.813262
19	0.376492	-1.418671	-1.053264	-1.140102	-1.173453	-0.874386	0.645579
20	0.905268	-1.155506	-0.747817	-0.854214	-0.754146	0.244314	0.402439
21	-0.994411	1.256840	-0.150808	0.017745	0.128608	-1.247285	0.159299
22	-1.415474	-0.629176	0.987674	0.875409	0.989293	-0.128586	-0.326982
23	0.219817	0.204180	1.667986	1.604423	1.607220	-0.874386	-1.056402
24	0.063143	-1.067784	-1.303174	-1.197279	-1.151385	0.990113	2.347560
25	-0.906281	0.423484	-0.858889	-0.568326	-0.533457	1.363013	1.618139
26	0.591919	2.134056	0.335129	0.317927	0.172746	-0.874386	0.159299
27	0.738802	-0.058985	0.585040	0.260750	0.283090	-0.874386	-1.299542
28	0.307947	0.598927	2.153924	2.319144	2.379630	0.244314	-1.785822

	14	15
0	1.643157	-0.471091
1	0.211006	-0.614898
2	-0.700362	-1.118222
3	-0.830557	0.463653
4	-1.090948	-0.614898
5	0.601593	-0.111574
6	-0.049384	-1.262029
7	2.163939	0.032233
8	-0.439971	0.176039
9	0.211006	1.757915
10	-0.570166	-0.686802
11	0.341202	-0.111574
12	0.211006	-0.039671
13	-2.002317	2.261239
14	2.424329	0.104136
15	1.382766	1.973625
16	0.601593	1.182687
17	-0.439971	1.542205

```
18 -1.090948 -0.327285
19  0.861984 -0.471091
20 -0.309775 -0.327285
21 -1.090948 -1.262029
22  0.080811 -0.327285
23 -1.090948  1.614108
24  0.471397 -1.405836
25 -0.309775 -1.190126
26  0.080811 -0.614898
27 -0.309775 -0.183478
28 -0.960753  0.032233
```

In [209]:
```
xf.columns
```

Out[209]:
```
RangeIndex(start=0, stop=16, step=1)
```

In [210]:
```
df_new.columns
```

Out[210]:
```
Index(['no. of wins', 'run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error'],
      dtype='object')
```

In [211]:
```
column = ['run scored', 'at bat', '1st base hit', '2nd base hit',
       '3rd base hit', 'home run', 'walk', 'strike out', 'stolen base',
       'run average', 'earned run', 'earned run average', 'complete game',
       'shutout', 'save', 'error']
```

In [212]:
```
xf.columns = column
# succesfully assigning all the column names again to the new 'xf' dataset
```

In [213]:
```
xf.head(3)
```

Out[213]:

	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	run average	earned run	earned run average	complete game	shutou
0	0.959398	0.830084	1.738306	1.556538	1.010845	-0.765863	-1.536359	-2.727623	0.905953	-0.664514	-0.511148	-0.511388	-0.501486	-0.81326
1	0.331147	-0.702831	-0.938698	0.201171	1.208917	-0.181389	-0.495021	0.121896	-0.585315	0.154638	0.232161	0.238952	-0.501486	0.15929
2	-0.274666	-1.100253	-0.106656	1.733325	-0.276617	-0.697101	1.252941	-0.925866	0.116458	-0.678397	-0.754153	-0.643801	2.854612	-0.32698

In [214]:
```
yf=y
```

In [215]:
```
yf.head(2)
```

Out[215]:

	no. of wins
0	95
1	83

In [ ]:
```

```

FINDING MULTICOLINEARITY

==========

In [216]:
```
# We have to find the multicollinearity between the features and to remove it we can use VIF (VARIANCE INFLATION FACTOR)
# we can not apply VIF on the TARGET COLUMN
# for apllyin VIF we have to import some libraries as follows
```

In [217]:
```
import statsmodels.api as sm
from scipy import stats
from statsmodels .stats.outliers_influence import variance_inflation_factor
```

In [218]:
```
# here we are making "def function" for calculating VIF
def calc_vif(xf):
    vif = pd.DataFrame()
    vif["FETURES"] = xf.columns
    vif["VIF FACTOR"] = [variance_inflation_factor(xf.values,i) for i in range (xf.shape[1])]
    return (vif)
```

In [219]:
```
xf.shape
```

Out[219]: (29, 16)

In [220]:
```
yf.shape
```

Out[220]: (29, 1)

In [221]:
```
calc_vif(xf)
# here we find that VIF VALUE of 'EARNED RUN ' & EARNED RUN AVERAGE is very high.
#  they are Highly MULTICOLLINERED WITH EACH OTHER. we have to drop one of the column between them.
```

Out[221]:

	FETURES	VIF FACTOR
0	run scored	7.229564
1	at bat	18.333377
2	1st base hit	10.558458
3	2nd base hit	3.743423
4	3rd base hit	3.278636
5	home run	8.426403
6	walk	3.431971
7	strike out	2.720640
8	stolen base	2.110804
9	run average	197.315706
10	earned run	1825.871631
11	earned run average	1438.816208
12	complete game	3.074115
13	shutout	3.673885
14	save	6.236487
15	error	2.180914

In [222]:
```
cor['no. of wins'].sort_values(ascending=False)
#  here as earlier also we can see that the EARNED RUN, EARNED RUN AVERAGE & RUN AVERAGE all three are MOST NEAGTIVELY .
# ..... RELATED WITH THR TARGET COLUMN. so we can drop any one of them...
```

Out[222]:
```
no. of wins           1.000000
save                  0.666530
walk                  0.484342
shutout               0.471805
run scored            0.430751
2nd base hit          0.427797
home run              0.307407
strike out            0.111850
complete game         0.080533
1st base hit          0.037612
at bat               -0.087947
error                -0.089485
stolen base          -0.157234
3rd base hit         -0.251118
earned run           -0.809435
run average          -0.812952
earned run average   -0.819600
Name: no. of wins, dtype: float64
```

In [223]:
```
#  Now we are dropping EARNED RUN column.
```

In [224]:
```
xf.drop(['earned run'],axis=1,inplace=True)
```

In [225]:
```
calc_vif(xf)
```

Out[225]:

	FETURES	VIF FACTOR
0	run scored	6.789990
1	at bat	7.689961
2	1st base hit	9.703669
3	2nd base hit	3.423326
4	3rd base hit	3.268301
5	home run	7.683683
6	walk	3.430050
7	strike out	2.713114
8	stolen base	1.907595
9	run average	144.425392
10	earned run average	139.967064
11	complete game	2.649316
12	shutout	3.498198
13	save	3.174949
14	error	2.092189

In [226]:
```
xf.drop(['run average'],axis=1,inplace=True)
```

In [227]:
```
calc_vif(xf)
# Now here after dropping EARNED RUN & RUN AVERAGE , our VIF is under controlled
#  and there is no such Huge Correlation between the columns.
```

Out[227]:

	FETURES	VIF FACTOR
0	run scored	6.787762
1	at bat	4.981743
2	1st base hit	8.569581
3	2nd base hit	3.416116
4	3rd base hit	3.239565
5	home run	6.590925
6	walk	3.081560
7	strike out	2.106522
8	stolen base	1.842528
9	earned run average	4.931066
10	complete game	2.649063
11	shutout	3.231029
12	save	3.075865
13	error	2.039641

========== UPTO HERE EDA AND OTHER TECHINIQUES ARE COMPLETED ==========

========== NOW WE NEED TO APPLY ML MODELS
==========

In [228]:
```
# NOW HERE WE CAN SEE THAT OUR TARGET/LABEL COLUMN IN NOT A CATEGORICAL DATA, IT IS HAVING NUMERICAL DATA,
# AND WHEN WE ARE HAVING "Y" (TARGET) IN NUMERICAL FORM THEN WE CAN APPLY "REGRESSION MODEL",
# SO HERE WE CAN APPLY REGRESSION MODEL ON OUR DATASET TO PREDICT, "NO. OF WINS".
```

In [229]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [230]:
```
lr = LinearRegression()
```

In [231]:
```
x_train,x_test,y_train,y_test = train_test_split(xf,yf,test_size=0.20,random_state=42)
```

In [232]:
```
lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
y_test.head(),y_pred[0:5]
```

Out[232]:
```
(    no. of wins
 28           74
 17           97
 13           76
 23           67
 9            78,
 array([[73.78843323],
        [88.99499525],
        [77.19737234],
        [66.72086528],
        [80.26818369]]))
```

In [233]:
```
#  here above we can see the similarity between "actual values" and "predicted values"
```

In [234]:
```
mean_squared_error (y_test,y_pred)
```

Out[234]:
```
19.68591744945243
```

In [235]:
```
r2_score(y_test,y_pred)
```

Out[235]:
```
0.7956479157496288
```

In [236]:
```
#  here above we can see that our R2 Score is aprrox 80.
```

In [237]:
```
xf.shape
```

Out[237]:
```
(29, 14)
```

In [238]:
```
def pred_func(w):
    w= w.reshape(1,14)
    win = lr.predict(w)
    print(win)

# making 'def' function to predict NO. OF WINS of any given value
```

In [239]:
```
#  TESTING SAMPLE -1
```

In [240]:
```
w= np.array([0.959398,0.830084,1.738306,1.556538,1.010845,-0.765863,-1.536359,-2.727623,0.905953,-0.511388,-0.501486,-0.
pred_func(w)

# here we are giving values to the model, and the model is predicting the NUMBER OF WINS .
# here the ACTUAL WINNING MATCHES for the values is = 95,  & OUR MODEL PREDICTING = 94
```

```
[[94.06678383]]
```

In [241]:
```
#  TESTING SAMPLE -2
```

In [242]:
```
w= np.array([-1.329231,0.233951,-0.359886,-0.800621,-0.474688,-0.869006,-1.145857,-0.201246,-0.673037,0.040333,1.363013,
pred_func(w)

# here the ACTUAL WINNING MATCHES for the values is = 76,  & OUR MODEL PREDICTING = 75
```

```
[[75.51673049]]
```

In [243]:
```
#  TESTING SAMPLE - 3
```

In [244]:
```
w= np.array([-0.274666,-1.100253,-0.106656,1.733325,-0.276617,-0.697101,1.252941,-0.925866,0.116458,-0.643801,2.854612,-
pred_func(w)

# here the ACTUAL WINNING MATCHES for the values is = 81,  & OUR MODEL PREDICTING = 81.94
```

```
[[81.94360098]]
```

In [245]:
```
yf.head(5)
```

Out[245]:

	no. of wins
0	95
1	83
2	81
3	76
4	74

In [246]:
```
xf.head(5)
```

Out[246]:

	run scored	at bat	1st base hit	2nd base hit	3rd base hit	home run	walk	strike out	stolen base	earned run average	complete game	shutout	save	err
0	0.959398	0.830084	1.738306	1.556538	1.010845	-0.765863	-1.536359	-2.727623	0.905953	-0.511388	-0.501486	-0.813262	1.643157	-0.4710
1	0.331147	-0.702831	-0.938698	0.201171	1.208917	-0.181389	-0.495021	0.121896	-0.585315	0.238952	-0.501486	0.159299	0.211006	-0.6148
2	-0.274666	-1.100253	-0.106656	1.733325	-0.276617	-0.697101	1.252941	-0.925866	0.116458	-0.643801	2.854612	-0.326982	-0.700362	-1.1182
3	-1.329231	0.233951	-0.359886	-0.800621	-0.474688	-0.869006	-1.145857	-0.201246	-0.673037	0.040333	1.363013	-0.570122	-0.830557	0.4636
4	0.174084	1.255894	2.063888	0.908319	1.704094	-0.353293	-0.197495	0.072935	-0.015124	1.496876	1.363013	0.159299	-1.090948	-0.6148

In [ ]:
```

```

==========HERE FROM THE ABOVE ALL 3 TESTING SAMPELS , WE CAN SAY THAT THE MODEL IS WORKING GOOD. AND THE ACCURACY OF PREDICTION IS 80% ============

In [ ]:
```

```

SAVING THE MODEL
==========

In [247]:
```
import pickle
```

In [248]:
```
file_name = 'BaseBall MatchWin Prediction.pkl'
pickle.dump(lr,open(file_name,'wb'))
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

========== COMPLETED
==========