VIRAT KOHLI ODI HUNDREDS

Exploratory Data Analysis (EDA)and Pre processing Report

1)Introduction:

Virat Kohli is one of the most successful cricketers in the history of One Day Internationals. He made his ODI debut in 2008 and quickly established himself as one of the world's top batsmen. His ability to chase down totals, accumulate runs, and maintain a high average makes him an invaluable player for India in ODIs.

A typical dataset for Virat Kohli's ODI career might contain the following columns:

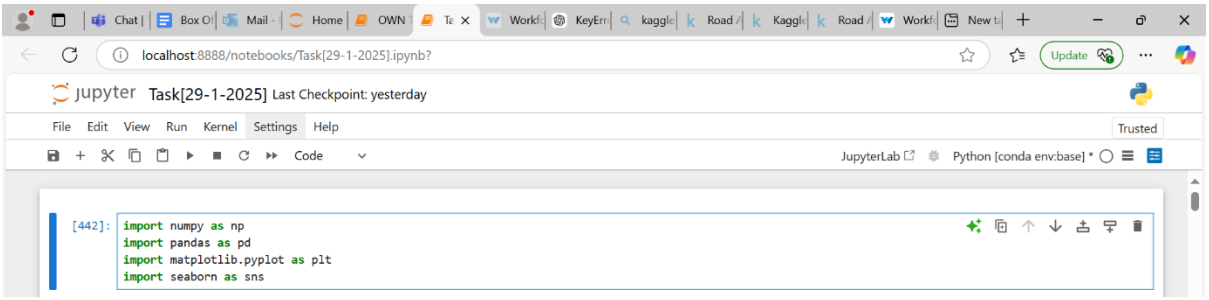
Data Analysis Process:

1. **Match Date**: The date on which the match was played.
2. **Opponent**: The team that India was playing against.
3. **Runs Scored**: The number of runs Kohli scored in that match.
4. **Fours**: The number of boundaries (fours) hit by Kohli.
5. **Sixes**: The number of sixes hit by Kohli.
6. **Strike Rate**: The number of runs scored per 100 balls.
7. **Dismissal**: The total runs scored divided by the number of innings played.
8. **Centuries**: Number of centuries (100+ runs) scored by Virat Kohli in ODIs.
9. **Ground/Location**: Where the match was played.
10. **Venue**: The specific venue where the match was held (e.g., Wankhede Stadium, MCG).
11. **Batting Position**: Where Kohli batted in the lineup (e.g., number 3, number 4).
12. **Balls Faced**: The number of balls faced by Virat Kohli in that match.

2)Import library

The code snippet imports essential libraries commonly used in data analysis and visualization with Python. numpy (imported as np) is a library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. pandas (imported as pd) is a powerful library for data manipulation and analysis, particularly for working with structured data like tables (DataFrames). matplotlib.pyplot (imported as plt) is a plotting library used for

creating static, interactive, and animated visualizations, like graphs and charts. seaborn (imported as sns) is built on top of matplotlib and provides a high-level interface for creating more attractive and informative statistical graphics. These libraries are frequently used together in data analysis workflows to clean, analyze, and visualize data.



3)Import Dataset

To import a dataset in Python, you typically use the pandas library, as it is one of the most efficient and convenient tools for handling datasets. Depending on the format of your dataset (CSV, Excel, SQL, etc.), the method of importing will vary.

```
[444]: dataset=pd.read_csv("Virat_Kohli_ODI.csv")
[446]: dataset.head(10)
```

[446]:

	Runs	Mins	BF	4s	6s	SR	Pos	Dismissal	Inns	Opposition	Ground	Start Date
0	12	33.0	22	1	0	54.54	2	lbw	1	v Sri Lanka	Dambulla	18-Aug-08
1	37	82.0	67	6	0	55.22	2	caught	2	v Sri Lanka	Dambulla	20-Aug-08
2	25	40.0	38	4	0	65.78	1	run out	1	v Sri Lanka	Colombo (RPS)	24-Aug-08
3	54	87.0	66	7	0	81.81	1	bowled	1	v Sri Lanka	Colombo (RPS)	27-Aug-08
4	31	45.0	46	3	1	67.39	1	lbw	2	v Sri Lanka	Colombo (RPS)	29-Aug-08
5	2	6.0	2	0	0	100	7	not out	1	v Sri Lanka	Colombo (RPS)	14-Sep-09
6	30	53.0	41	3	0	73.17	4	caught	2	v Australia	Vadodara	25-Oct-09
7	10	32.0	16	1	0	62.5	3	caught	2	v Australia	Mohali	2-Nov-09
8	27	20.0	19	4	0	142.1	7	bowled	1	v Sri Lanka	Rajkot	15-Dec-09
9	54	93.0	65	7	0	83.07	4	lbw	1	v Sri Lanka	Nagpur	18-Dec-09

4) Dataset.info

The info() method provides a concise summary of your DataFrame, including the umber of entries, column names, non-null values, and data types of each column

```
[24]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Runs         132 non-null    object
 1   Mins         126 non-null    float64
 2   BF           132 non-null    int64
 3   4s           132 non-null    int64
 4   6s           132 non-null    int64
 5   SR           132 non-null    object
 6   Pos          132 non-null    int64
 7   Dismissal    132 non-null    object
 8   Inns         132 non-null    int64
 9   Opposition   132 non-null    object
 10  Ground       132 non-null    object
 11  Start Date   132 non-null    object
dtypes: float64(1), int64(5), object(6)
memory usage: 12.5+ KB
```

5) Dataset.describe()

The describe() method generates summary statistics for the numeric columns in your dataset, such as count, mean, standard deviation, minimum, maximum, and percentiles.

```
[450]: dataset.describe()
```

[450]:

	Mins	BF	4s	6s	Pos	Inns
count	126.000000	132.000000	132.000000	132.000000	132.000000	132.000000
mean	70.492063	50.871212	4.371212	0.545455	3.303030	1.575758
std	57.270131	38.729716	4.404032	1.086795	0.873174	0.496110
min	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000
25%	18.250000	17.750000	1.000000	0.000000	3.000000	1.000000
50%	56.000000	42.500000	3.000000	0.000000	3.000000	2.000000
75%	120.750000	82.250000	7.000000	1.000000	4.000000	2.000000
max	202.000000	140.000000	18.000000	7.000000	7.000000	2.000000

6) Dataset.dtype()

The dtype attribute in pandas is used to check the data type of each column in a DataFrame. This is useful when you want to confirm whether the data in your columns are of the expected type (e.g., integers, floats, strings, etc.).

```
[464]: dataset.dtypes

[464]: Runs          object
       Mins         float64
       BF             int64
       4s             int64
       6s             int64
       SR            object
       Pos            int64
       Dismissal     object
       Inns           int64
       Opposition    object
       Ground        object
       Start Date    object
       dtype: object
```

7)Dataset.shape

The `shape` attribute in pandas provides the dimensions of a DataFrame, showing the number of rows and columns it contains. It returns a tuple where the first value is the number of rows, and the second value is the number of columns.

```
[466]: dataset.shape

[466]: (132, 12)

[468]: dataset=pd.read_csv("Virat_Kohli_ODI.csv")

[495]: sales_by_category_region = pd.pivot_table(
           dataset,
           values='6s',
```

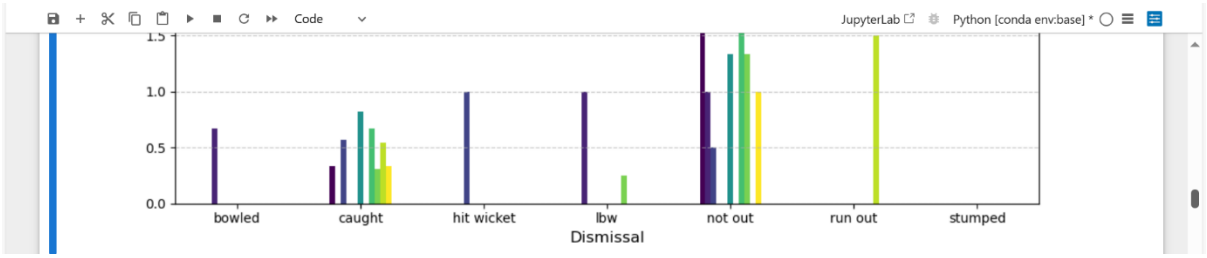
8) **Missing datas**

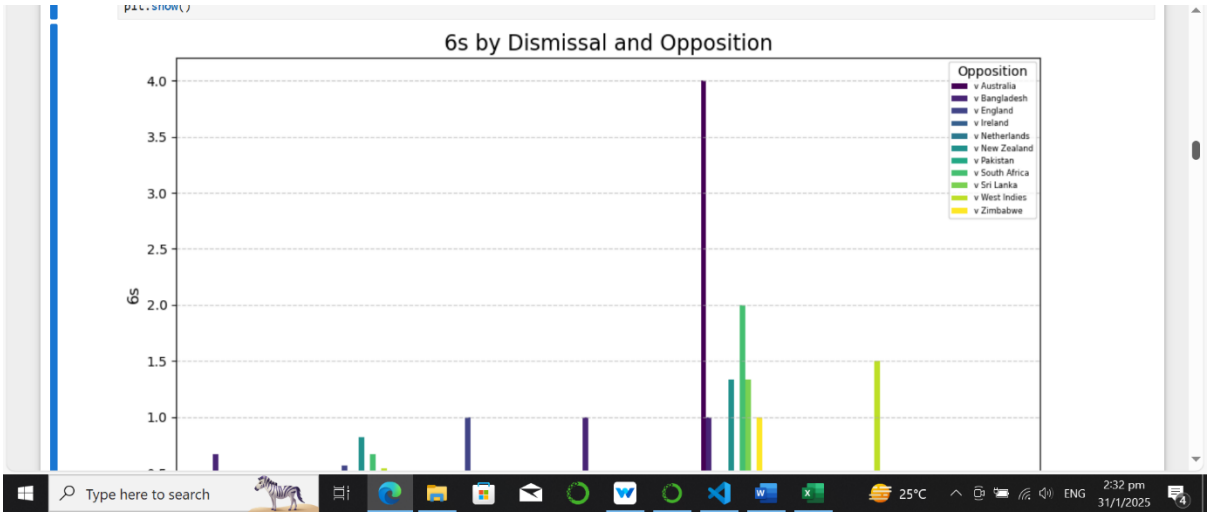
Handling missing data is crucial for performing accurate analysis, as it can affect the results and models you build. In pandas, you can identify, count, and handle missing data using several methods.

```
[458]: missing = dataset.isnull().sum()

       missing[missing > 0].sort_values(ascending=False).head()

[458]: Mins    6
       dtype: int64

[460]: dataset_columns = ["Pos","Ground","4s"]
       for i in dataset_columns:
           print(f"{i}: {dataset[i].dtype}")
```

9)**Bar graph**

Bar graph shows the number of sixes hit by kohli at the same time it will shows the Dismissel type and it will also shows opposition team by various colors, The image will shows the classification.

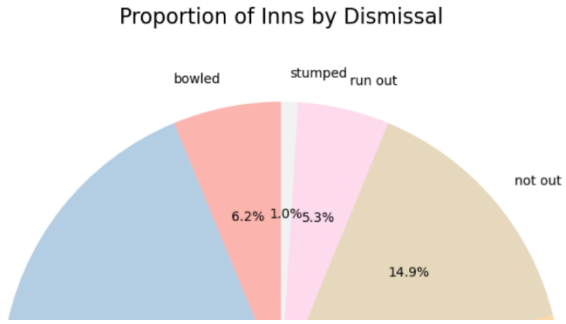


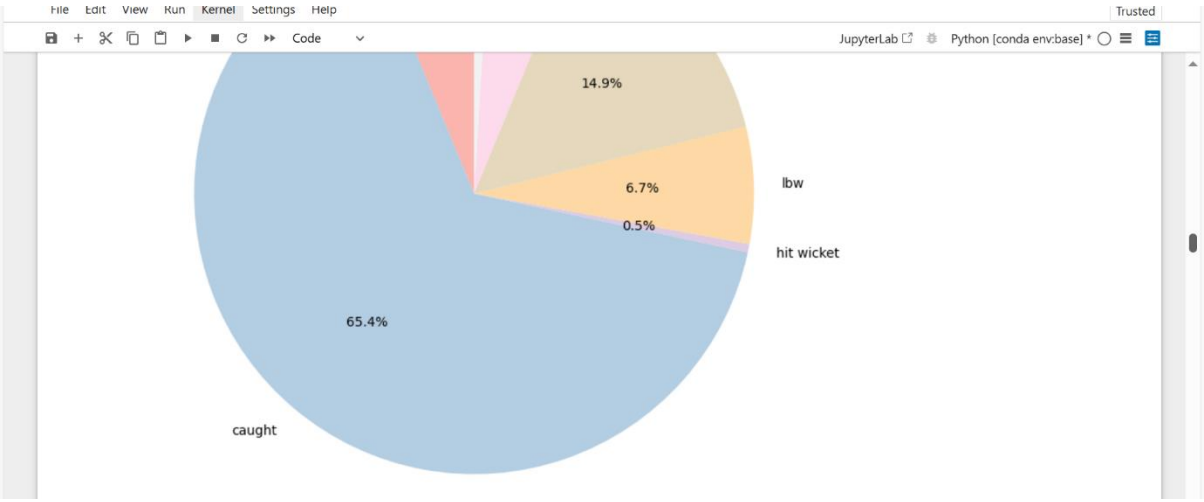


10) Pie chart

The Pie chart will shows the Dismissal by every innings with percentage by the player

```
[472]: sales_proportion = dataset.groupby('Dismissal')['Inns'].sum() / dataset['Inns'].sum()   #piechart

sales_proportion.plot(kind='pie', autopct='%1.1f%%', startangle=90, colormap='Pastel1', figsize=(8, 8))
plt.title('Proportion of Inns by Dismissal', fontsize=16)
plt.ylabel('')
plt.tight_layout()
plt.show()
```

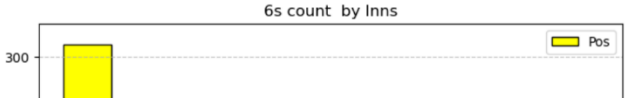


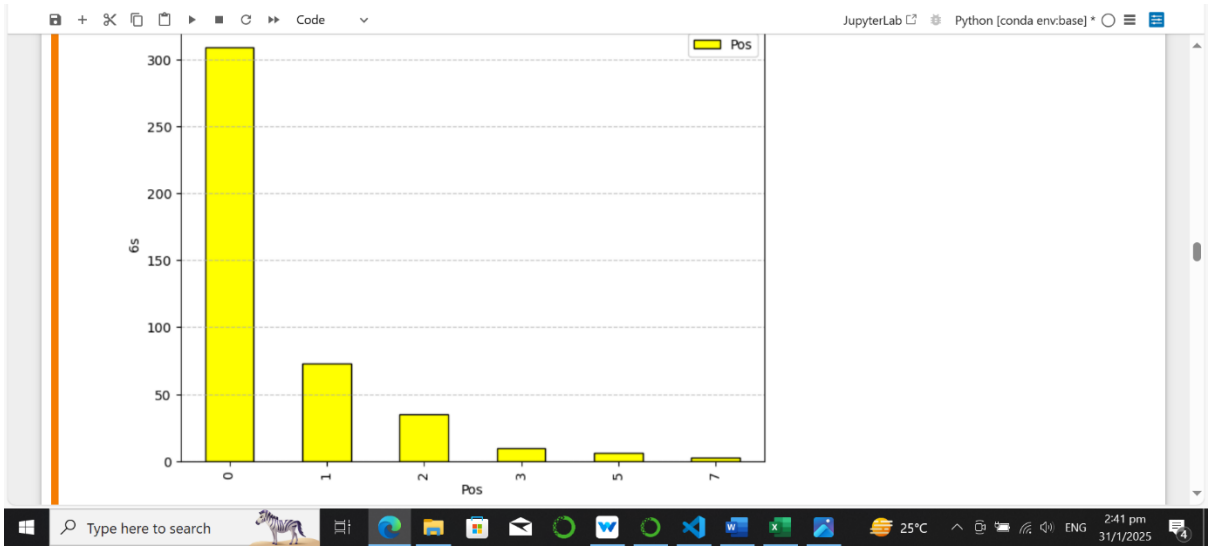


11)Bar graph

Bar graph will shows the hitting 6s in which postion by the player,with code.

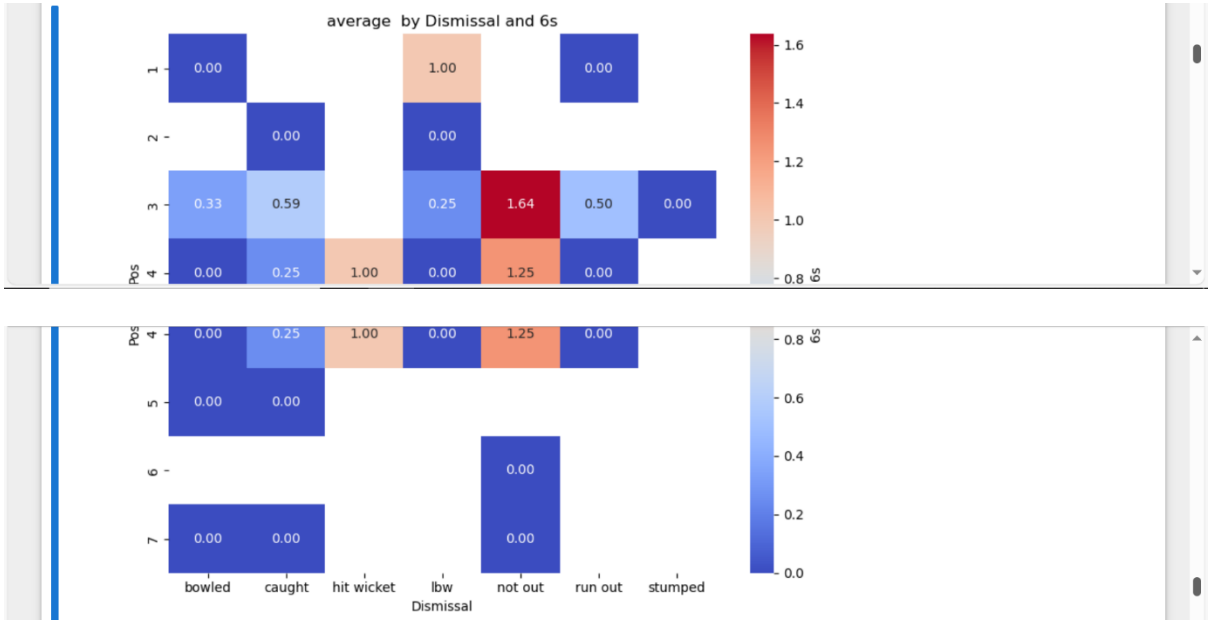
```
•[36]: pivot_bar2 = dataset.pivot_table(values="Pos", index="6s", aggfunc="sum")
pivot_bar2.plot(kind="bar", color=["yellow", "green"], edgecolor="black", figsize=(8, 6))
plt.title("6s count  by Pos")
plt.ylabel("6s")
plt.xlabel("Pos")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```





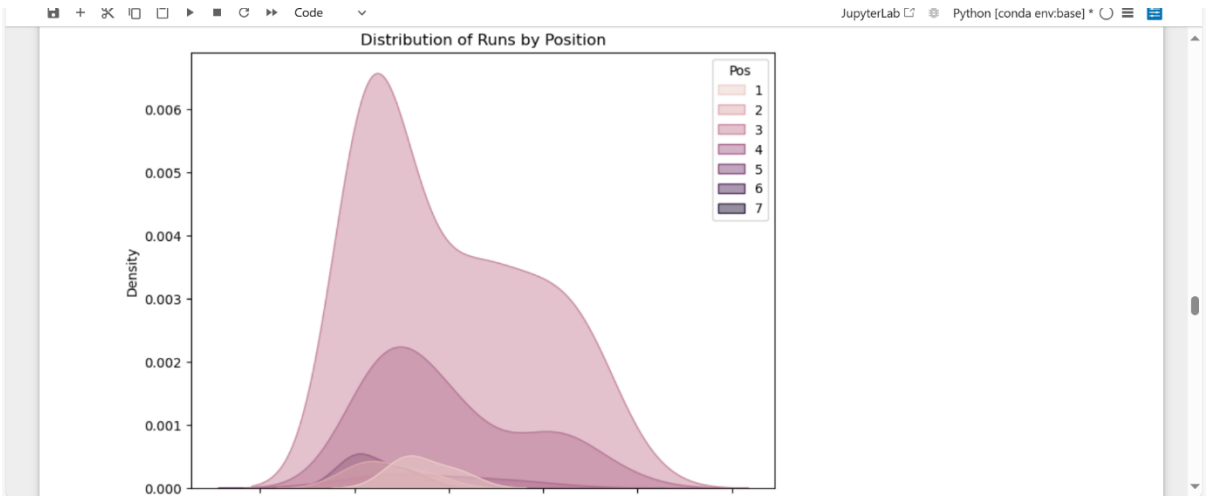
12)Heat Map

The heat map will shows the over all percentage of hitting 6s, postion and type dismissel by the player.



13)Density Plot

The density plot will shows the distribution of runs by the postions by the player.



14)Total hundred by player

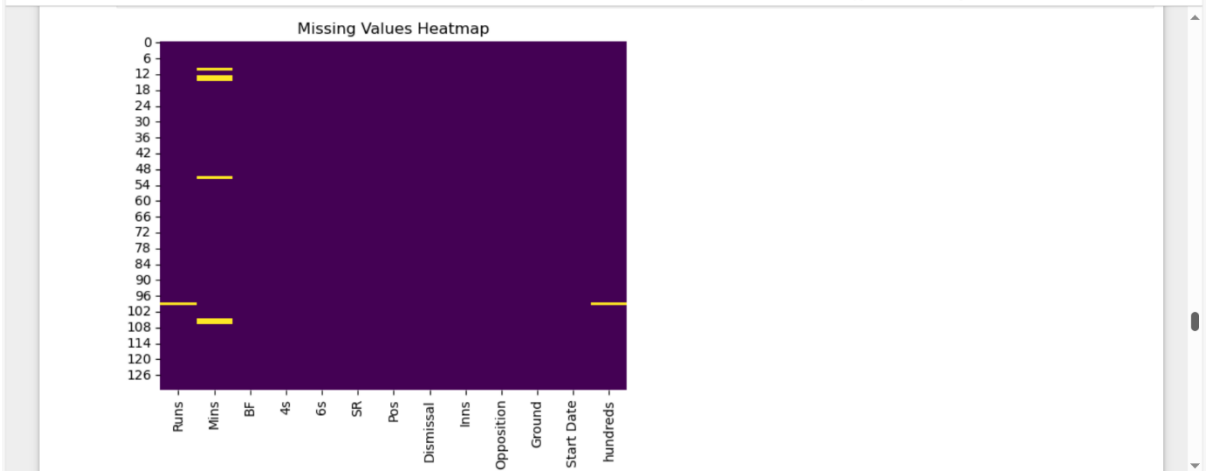
```
[480]:
def count_in_hundreds(df, column):

    df['hundreds'] = (df[column] // 100) * 100


    count_per_hundred = df['hundreds'].value_counts().sort_index()

    return count_per_hundred


count_per_hundred = count_in_hundreds(dataset, 'Runs')
print(count_per_hundred)
```

```
hundreds
0.0      107
100.0     24
Name: count, dtype: int64
```

The code will shows the over all century by player in the dataset.

15) Heat Map for Missing values

This heatmap will shows the missing values in the dataset.



16) Missing values filled by mean, media, mode

In the dataset missed values is filled by mean mode using python code.

```
[486]: # Mean Imputation for runs
mean_Runs = dataset['Runs'].mean()

dataset['Runs'] = dataset['Runs'].fillna(mean_Runs)

print(f"Filled missing values in 'Runs' with mean: {mean_Runs}")
```

```
Filled missing values in 'Runs' with mean: 47.56488549618321
```

```
[488]: # Mean Imputation for hundreds
mean_hundreds = dataset['hundreds'].mean()

dataset['hundreds'] = dataset['hundreds'].fillna(mean_hundreds)

print(f"Filled missing values in 'hundreds' with mean: {mean_hundreds}")
```

```
Filled missing values in 'hundreds' with mean: 18.3206106870229
```

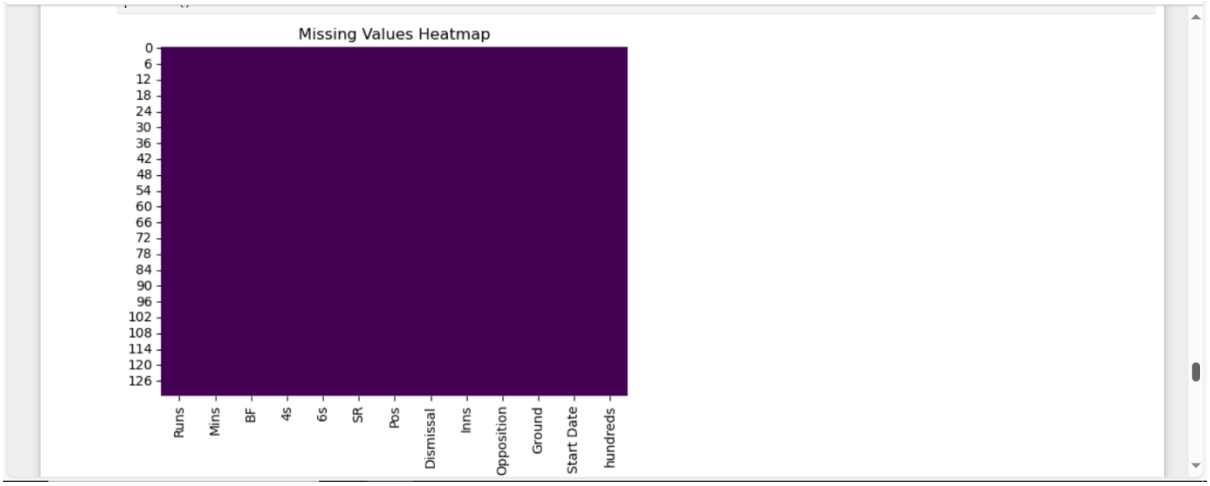
```
[490]: # Mean Imputation for Mins
mean_Mins = dataset['Mins'].mean()

dataset['Mins'] = dataset['Mins'].fillna(mean_Mins)

print(f"Filled missing values in 'mins' with mean: {mean_Mins}")
```

```
Filled missing values in 'mins' with mean: 70.4920634920635
```

```
[492]: sns.heatmap(dataset.isnull(), cbar=False, cmap='viridis')
```

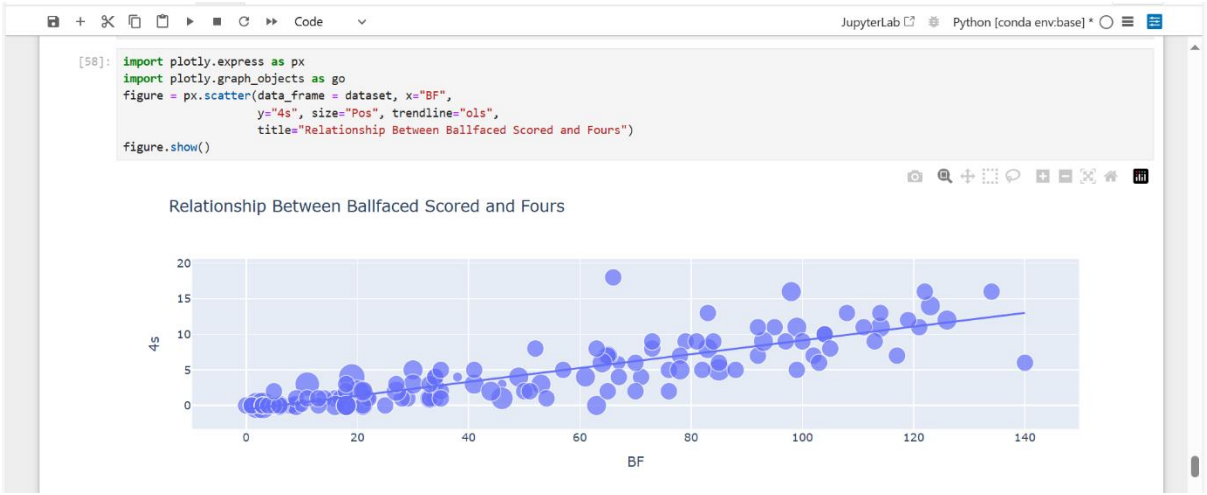
17) Second Heatmap

The heat Map will shows the filled values in the dataset.



18) Scatter Plot

The scatter plot will shows the Relationship Between Ballfaced Scored and Fours.



19) Box plot

The box plot will shows the runs scoring level which different Teams .