In [69]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\shaw3\anaconda3\lib\site-p
ackages (5.24.1)
Requirement already satisfied: packaging in c:\users\shaw3\anaconda3\lib\sit
e-packages (from plotly) (21.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shaw3\anaconda3\l
ib\site-packages (from plotly) (8.5.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shaw3\an
aconda3\lib\site-packages (from packaging->plotly) (3.0.9)
```

In [70]:
```
!pip install --upgrade plotly
!pip install --upgrade jupyter
```

```
Requirement already satisfied: plotly in c:\users\shaw3\anaconda3\lib\site-p
ackages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shaw3\anaconda3\l
ib\site-packages (from plotly) (8.5.0)
Requirement already satisfied: packaging in c:\users\shaw3\anaconda3\lib\sit
e-packages (from plotly) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shaw3\an
aconda3\lib\site-packages (from packaging->plotly) (3.0.9)
Requirement already satisfied: jupyter in c:\users\shaw3\anaconda3\lib\site-
packages (1.1.1)
Requirement already satisfied: jupyterlab in c:\users\shaw3\anaconda3\lib\si
te-packages (from jupyter) (3.4.4)
Requirement already satisfied: nbconvert in c:\users\shaw3\anaconda3\lib\sit
e-packages (from jupyter) (6.4.4)
Requirement already satisfied: jupyter-console in c:\users\shaw3\anaconda3\l
ib\site-packages (from jupyter) (6.4.3)
Requirement already satisfied: ipywidgets in c:\users\shaw3\anaconda3\lib\si
te-packages (from jupyter) (7.6.5)
Requirement already satisfied: notebook in c:\users\shaw3\anaconda3\lib\site
-packages (from jupyter) (6.4.12)
Requirement already satisfied: ipykernel in c:\users\shaw3\anaconda3\lib\sit
e-packages (from jupyter) (6.15.2)
Requirement already satisfied: debugpy>=1.0 in c:\users\shaw3\anaconda3\lib
\site-packages (from ipykernel->jupyter) (1.5.1)
Requirement already satisfied: ipython>=7.23.1 in c:\users\shaw3\anaconda3\l
ib\site-packages (from ipykernel->jupyter) (7.31.1)
Requirement already satisfied: traitlets>=5.1.0 in c:\users\shaw3\anaconda3
\lib\site-packages (from ipykernel->jupyter) (5.1.1)
Requirement already satisfied: matplotlib-inline>=0.1 in c:\users\shaw3\anac
onda3\lib\site-packages (from ipykernel->jupyter) (0.1.6)
Requirement already satisfied: pyzmq>=17 in c:\users\shaw3\anaconda3\lib\sit
e-packages (from ipykernel->jupyter) (23.2.0)
Requirement already satisfied: packaging in c:\users\shaw3\anaconda3\lib\sit
e-packages (from ipykernel->jupyter) (21.3)
Requirement already satisfied: psutil in c:\users\shaw3\anaconda3\lib\site-p
ackages (from ipykernel->jupyter) (5.9.0)
Requirement already satisfied: nest-asyncio in c:\users\shaw3\anaconda3\lib
\site-packages (from ipykernel->jupyter) (1.5.5)
Requirement already satisfied: tornado>=6.1 in c:\users\shaw3\anaconda3\lib
\site-packages (from ipykernel->jupyter) (6.1)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\shaw3\anac
onda3\lib\site-packages (from ipykernel->jupyter) (7.3.4)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\shaw3\a
naconda3\lib\site-packages (from ipywidgets->jupyter) (1.0.0)
Requirement already satisfied: nbformat>=4.2.0 in c:\users\shaw3\anaconda3\l
ib\site-packages (from ipywidgets->jupyter) (5.5.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\users\shaw3\ana
conda3\lib\site-packages (from ipywidgets->jupyter) (0.2.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in c:\users\shaw3\a
naconda3\lib\site-packages (from ipywidgets->jupyter) (3.5.2)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
in c:\users\shaw3\anaconda3\lib\site-packages (from jupyter-console->jupyte
r) (3.0.20)
Requirement already satisfied: pygments in c:\users\shaw3\anaconda3\lib\site
-packages (from jupyter-console->jupyter) (2.17.2)
Requirement already satisfied: jupyter-core in c:\users\shaw3\anaconda3\lib
```

```
\site-packages (from jupyterlab->jupyter) (4.11.1)
Requirement already satisfied: jinja2>=2.1 in c:\users\shaw3\anaconda3\lib\s
ite-packages (from jupyterlab->jupyter) (2.11.3)
Requirement already satisfied: jupyter-server~=1.16 in c:\users\shaw3\anacon
da3\lib\site-packages (from jupyterlab->jupyter) (1.18.1)
Requirement already satisfied: nbclassic in c:\users\shaw3\anaconda3\lib\sit
e-packages (from jupyterlab->jupyter) (0.3.5)
Requirement already satisfied: jupyterlab-server~=2.10 in c:\users\shaw3\ana
conda3\lib\site-packages (from jupyterlab->jupyter) (2.10.3)
Requirement already satisfied: terminado>=0.8.3 in c:\users\shaw3\anaconda3
\lib\site-packages (from notebook->jupyter) (0.13.1)
Requirement already satisfied: prometheus-client in c:\users\shaw3\anaconda3
\lib\site-packages (from notebook->jupyter) (0.14.1)
Requirement already satisfied: Send2Trash>=1.8.0 in c:\users\shaw3\anaconda3
\lib\site-packages (from notebook->jupyter) (1.8.0)
Requirement already satisfied: argon2-cffi in c:\users\shaw3\anaconda3\lib\s
ite-packages (from notebook->jupyter) (21.3.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\shaw3\anaconda
3\lib\site-packages (from nbconvert->jupyter) (0.4)
Requirement already satisfied: jupyterlab-pygments in c:\users\shaw3\anacond
a3\lib\site-packages (from nbconvert->jupyter) (0.1.2)
Requirement already satisfied: beautifulsoup4 in c:\users\shaw3\anaconda3\li
b\site-packages (from nbconvert->jupyter) (4.11.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\shaw3\anaconda3
\lib\site-packages (from nbconvert->jupyter) (0.8.4)
Requirement already satisfied: defusedxml in c:\users\shaw3\anaconda3\lib\si
te-packages (from nbconvert->jupyter) (0.7.1)
Requirement already satisfied: testpath in c:\users\shaw3\anaconda3\lib\site
-packages (from nbconvert->jupyter) (0.6.0)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\shaw3\anac
onda3\lib\site-packages (from nbconvert->jupyter) (0.5.13)
Requirement already satisfied: bleach in c:\users\shaw3\anaconda3\lib\site-p
ackages (from nbconvert->jupyter) (4.1.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\shaw3\anacon
da3\lib\site-packages (from nbconvert->jupyter) (1.5.0)
Requirement already satisfied: setuptools>=18.5 in c:\users\shaw3\anaconda3
\lib\site-packages (from ipython>=7.23.1->ipykernel->jupyter) (63.4.1)
Requirement already satisfied: decorator in c:\users\shaw3\anaconda3\lib\sit
e-packages (from ipython>=7.23.1->ipykernel->jupyter) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\shaw3\anaconda3\lib\si
te-packages (from ipython>=7.23.1->ipykernel->jupyter) (0.18.1)
Requirement already satisfied: pickleshare in c:\users\shaw3\anaconda3\lib\s
ite-packages (from ipython>=7.23.1->ipykernel->jupyter) (0.7.5)
Requirement already satisfied: backcall in c:\users\shaw3\anaconda3\lib\site
-packages (from ipython>=7.23.1->ipykernel->jupyter) (0.2.0)
Requirement already satisfied: colorama in c:\users\shaw3\anaconda3\lib\site
-packages (from ipython>=7.23.1->ipykernel->jupyter) (0.4.5)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\shaw3\anaconda3
\lib\site-packages (from jinja2>=2.1->jupyterlab->jupyter) (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\shaw3\anac
onda3\lib\site-packages (from jupyter-client>=6.1.12->ipykernel->jupyter)
(2.8.2)
Requirement already satisfied: pywin32>=1.0 in c:\users\shaw3\anaconda3\lib
\site-packages (from jupyter-core->jupyterlab->jupyter) (302)
Requirement already satisfied: anyio<4,>=3.1.0 in c:\users\shaw3\anaconda3\l
ib\site-packages (from jupyter-server~=1.16->jupyterlab->jupyter) (3.5.0)
```

```
Requirement already satisfied: pywinpty in c:\users\shaw3\anaconda3\lib\site
-packages (from jupyter-server~=1.16->jupyterlab->jupyter) (2.0.2)
Requirement already satisfied: websocket-client in c:\users\shaw3\anaconda3
\lib\site-packages (from jupyter-server~=1.16->jupyterlab->jupyter) (0.58.0)
Requirement already satisfied: jsonschema>=3.0.1 in c:\users\shaw3\anaconda3
\lib\site-packages (from jupyterlab-server~=2.10->jupyterlab->jupyter) (4.1
6.0)
Requirement already satisfied: requests in c:\users\shaw3\anaconda3\lib\site
-packages (from jupyterlab-server~=2.10->jupyterlab->jupyter) (2.28.1)
Requirement already satisfied: babel in c:\users\shaw3\anaconda3\lib\site-pa
ckages (from jupyterlab-server~=2.10->jupyterlab->jupyter) (2.9.1)
Requirement already satisfied: json5 in c:\users\shaw3\anaconda3\lib\site-pa
ckages (from jupyterlab-server~=2.10->jupyterlab->jupyter) (0.9.6)
Requirement already satisfied: fastjsonschema in c:\users\shaw3\anaconda3\li
b\site-packages (from nbformat>=4.2.0->ipywidgets->jupyter) (2.16.2)
Requirement already satisfied: wcwidth in c:\users\shaw3\anaconda3\lib\site-
packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->jupyter-console
->jupyter) (0.2.5)
Requirement already satisfied: argon2-cffi-bindings in c:\users\shaw3\anacon
da3\lib\site-packages (from argon2-cffi->notebook->jupyter) (21.2.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\shaw3\anaconda3\lib
\site-packages (from beautifulsoup4->nbconvert->jupyter) (2.3.1)
Requirement already satisfied: six>=1.9.0 in c:\users\shaw3\anaconda3\lib\si
te-packages (from bleach->nbconvert->jupyter) (1.16.0)
Requirement already satisfied: webencodings in c:\users\shaw3\anaconda3\lib
\site-packages (from bleach->nbconvert->jupyter) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shaw3\an
aconda3\lib\site-packages (from packaging->ipykernel->jupyter) (3.0.9)
Requirement already satisfied: idna>=2.8 in c:\users\shaw3\anaconda3\lib\sit
e-packages (from anyio<4,>=3.1.0->jupyter-server~=1.16->jupyterlab->jupyter)
(3.3)
Requirement already satisfied: sniffio>=1.1 in c:\users\shaw3\anaconda3\lib
\site-packages (from anyio<4,>=3.1.0->jupyter-server~=1.16->jupyterlab->jupy
ter) (1.2.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\shaw3\anacond
a3\lib\site-packages (from jedi>=0.16->ipython>=7.23.1->ipykernel->jupyter)
(0.8.3)
Requirement already satisfied: attrs>=17.4.0 in c:\users\shaw3\anaconda3\lib
\site-packages (from jsonschema>=3.0.1->jupyterlab-server~=2.10->jupyterlab-
>jupyter) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0
in c:\users\shaw3\anaconda3\lib\site-packages (from jsonschema>=3.0.1->jupyt
erlab-server~=2.10->jupyterlab->jupyter) (0.18.0)
Requirement already satisfied: cffi>=1.0.1 in c:\users\shaw3\anaconda3\lib\s
ite-packages (from argon2-cffi-bindings->argon2-cffi->notebook->jupyter) (1.
15.1)
Requirement already satisfied: pytz>=2015.7 in c:\users\shaw3\anaconda3\lib
\site-packages (from babel->jupyterlab-server~=2.10->jupyterlab->jupyter) (2
022.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shaw3\anaconda
3\lib\site-packages (from requests->jupyterlab-server~=2.10->jupyterlab->jup
yter) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\shaw3\an
aconda3\lib\site-packages (from requests->jupyterlab-server~=2.10->jupyterla
b->jupyter) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\shaw3\anaco
```

```
nda3\lib\site-packages (from requests->jupyterlab-server~=2.10->jupyterlab->
jupyter) (1.26.11)
Requirement already satisfied: pycparser in c:\users\shaw3\anaconda3\lib\sit
e-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook->j
upyter) (2.21)
```

In [97]:
```
import plotly.express as px
import plotly.graph_objects as go
```

In [98]:
```
# Initialize Plotly for Jupyter Notebooks
import plotly.offline as pyo
pyo.init_notebook_mode(connected=True)
```

In [96]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [74]:
```
matches= pd.read_csv("matches.csv")
delivery= pd.read_csv("deliveries.csv")
```

In [75]:
```
matches.head()
```

Out[75]:

	id	season	city	date	match_type	player_of_match	ven
0	335982	2007/08	Bangalore	2008-04-18	League	BB McCullum	Chinnaswa Stadi
1	335983	2007/08	Chandigarh	2008-04-19	League	MEK Hussey	Pun Cric Associat Stadiu Moh
2	335984	2007/08	Delhi	2008-04-19	League	MF Maharoof	Feroz Sh Kc
3	335985	2007/08	Mumbai	2008-04-20	League	MV Boucher	Wankhe Stadi
4	335986	2007/08	Kolkata	2008-04-20	League	DJ Hussey	Ec Garde

In [76]:
```
delivery.head()
```

Out[76]:

	match_id	inning	batting_team	bowling_team	over	ball	batter	bowle
0	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	1	SC Ganguly	I Kuma
1	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	2	BB McCullum	I Kuma
2	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	3	BB McCullum	I Kuma
3	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	4	BB McCullum	I Kuma
4	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	5	BB McCullum	I Kuma

In [77]:
```
matches.tail()
```

Out[77]:

	id	season	city	date	match_type	player_of_match	
1090	1426307	2024	Hyderabad	2024-05-19	League	Abhishek Sharma	Raji Inte : Upp
1091	1426309	2024	Ahmedabad	2024-05-21	Qualifier 1	MA Starc	N : Ahr
1092	1426310	2024	Ahmedabad	2024-05-22	Eliminator	R Ashwin	N : Ahr
1093	1426311	2024	Chennai	2024-05-24	Qualifier 2	Shahbaz Ahmed	Chida : C
1094	1426312	2024	Chennai	2024-05-26	Final	MA Starc	Chida : C

In [78]:
```
delivery.tail()
```

Out[78]:

	match_id	inning	batting_team	bowling_team	over	ball	batter	b
260915	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	9	5	SS Iyer	Ma
260916	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	9	6	VR Iyer	Ma
260917	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	10	1	VR Iyer	Sh A
260918	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	10	2	SS Iyer	Sh A
260919	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	10	3	VR Iyer	Sh A

In [79]:
```
ipl= delivery.merge(matches, left_on= 'match_id', right_on ='id')
ipl.head()
```

Out[79]:

	match_id	inning	batting_team	bowling_team	over	ball	batter	bowle
0	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	1	SC Ganguly	I Kuma
1	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	2	BB McCullum	I Kuma
2	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	3	BB McCullum	I Kuma
3	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	4	BB McCullum	I Kuma
4	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	5	BB McCullum	I Kuma

5 rows × 37 columns

avg vs sr graph for top 50 batsman(in trumns of run)

fetching a new dataframe with top 50 batsman

In [80]:
```
top50 = ipl.groupby('batter')['batsman_runs'].sum().sort_values(ascending=Fa
```

In [81]:
```
top50
```

Out[81]:
```
Index(['V Kohli', 'S Dhawan', 'RG Sharma', 'DA Warner', 'SK Raina', 'MS Dho
ni',
       'AB de Villiers', 'CH Gayle', 'RV Uthappa', 'KD Karthik', 'KL Rahu
l',
       'AM Rahane', 'F du Plessis', 'SV Samson', 'AT Rayudu', 'G Gambhir',
       'SR Watson', 'MK Pandey', 'SA Yadav', 'JC Buttler', 'KA Pollard',
       'RR Pant', 'YK Pathan', 'Shubman Gill', 'Q de Kock', 'SS Iyer',
       'RA Jadeja', 'WP Saha', 'DA Miller', 'BB McCullum', 'PA Patel',
       'GJ Maxwell', 'Yuvraj Singh', 'V Sehwag', 'MA Agarwal', 'Ishan Kisha
n',
       'N Rana', 'M Vijay', 'HH Pandya', 'SPD Smith', 'SE Marsh', 'AD Russe
ll',
       'JH Kallis', 'DR Smith', 'RD Gaikwad', 'SR Tendulkar', 'RA Tripath
i',
       'R Dravid', 'KS Williamson', 'AJ Finch'],
      dtype='object', name='batter')
```

In [82]:
```
new_ipl= ipl[ipl['batter'].isin(top50)]
```

In [83]:
```
new_ipl
```

Out[83]:

	match_id	inning	batting_team	bowling_team	over	ball	batter
1	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	2	BB McCullum
2	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	3	BB McCullum
3	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	4	BB McCullum
4	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	5	BB McCullum
5	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	6	BB McCullum
...	...	...	...	...	...	...	...
260763	1426312	1	Sunrisers Hyderabad	Kolkata Knight Riders	4	1	RA Tripathi
260764	1426312	1	Sunrisers Hyderabad	Kolkata Knight Riders	4	2	RA Tripathi
260910	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	8	6	SS Iyer
260915	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	9	5	SS Iyer
260918	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	10	2	SS Iyer

139298 rows × 37 columns

calculating SR

sr= (number of runs scored)/(number of balls played)*100

```
In [84]: runs= new_ipl.groupby('batter')['batsman_runs'].sum()
         balls= new_ipl.groupby('batter')['batsman_runs'].count()

         sr= (runs/balls)*100
         sr=sr.reset_index()
```

```
In [85]: sr
```

Out[85]:

	batter	batsman_runs
0	AB de Villiers	148.580442
1	AD Russell	164.224422
2	AJ Finch	123.349057
3	AM Rahane	120.321410
4	AT Rayudu	124.584527
5	BB McCullum	126.848592
6	CH Gayle	142.121729
7	DA Miller	134.684477
8	DA Warner	135.429986
9	DR Smith	132.279534
10	F du Plessis	133.071325
11	G Gambhir	119.665153
12	GJ Maxwell	150.488599
13	HH Pandya	139.691290
14	Ishan Kishan	132.797589
15	JC Buttler	142.238984
16	JH Kallis	105.936272
17	KA Pollard	140.457703
18	KD Karthik	131.353404
19	KL Rahul	131.050866
20	KS Williamson	122.952710
21	M Vijay	118.614130
22	MA Agarwal	128.255646
23	MK Pandey	117.366180
24	MS Dhoni	132.835065
25	N Rana	130.818859
26	PA Patel	116.625717
27	Q de Kock	131.120332
28	R Dravid	113.347237
29	RA Jadeja	124.432296
30	RA Tripathi	135.515152
31	RD Gaikwad	133.632791
32	RG Sharma	127.918194

	batter	batsman_runs
33	RR Pant	143.597561
34	RV Uthappa	126.152279
35	S Dhawan	123.454313
36	SA Yadav	142.505948
37	SE Marsh	130.109775
38	SK Raina	132.535312
39	SPD Smith	124.812406
40	SR Tendulkar	114.187867
41	SR Watson	134.163209
42	SS Iyer	123.025540
43	SV Samson	135.137615
44	Shubman Gill	132.236842
45	V Kohli	128.511867
46	V Sehwag	148.827059
47	WP Saha	123.902027
48	YK Pathan	138.046272
49	Yuvraj Singh	124.784776

calculated avg

avg= (total number of runs)/(total numbers of out)

In [86]:
```
out= ipl[ipl['player_dismissed'].isin(top50)]

nout= out['player_dismissed'].value_counts()

avg= runs/nout
avg=avg.reset_index()

avg.rename(columns={'index': 'batter', 0: 'avg'}, inplace= True)

avg=avg.merge(sr, on= 'batter')
```

In [87]:
```
avg
```

Out[87]:

	batter	avg	batsman_runs
0	AB de Villiers	39.853846	148.580442
1	AD Russell	28.930233	164.224422
2	AJ Finch	24.904762	123.349057
3	AM Rahane	30.142857	120.321410
4	AT Rayudu	28.051613	124.584527
5	BB McCullum	27.711538	126.848592
6	CH Gayle	39.658730	142.121729
7	DA Miller	35.658537	134.684477
8	DA Warner	40.042683	135.429986
9	DR Smith	28.392857	132.279534
10	F du Plessis	35.992126	133.071325
11	G Gambhir	31.007353	119.665153
12	GJ Maxwell	24.750000	150.488599
13	HH Pandya	28.471910	139.691290
14	Ishan Kishan	28.430108	132.797589
15	JC Buttler	37.715789	142.238984
16	JH Kallis	28.552941	105.936272
17	KA Pollard	28.404959	140.457703
18	KD Karthik	26.320652	131.353404
19	KL Rahul	44.657143	131.050866
20	KS Williamson	35.533333	122.952710
21	M Vijay	25.930693	118.614130
22	MA Agarwal	22.811966	128.255646
23	MK Pandey	29.015038	117.366180
24	MS Dhoni	39.126866	132.835065
25	N Rana	28.344086	130.818859
26	PA Patel	22.603175	116.625717
27	Q de Kock	30.980392	131.120332
28	R Dravid	28.233766	113.347237
29	RA Jadeja	27.398148	124.432296
30	RA Tripathi	26.939759	135.515152
31	RD Gaikwad	41.754386	133.632791
32	RG Sharma	29.730942	127.918194

	batter	avg	batsman_runs
33	RR Pant	35.451613	143.597561
34	RV Uthappa	27.522222	126.152279
35	S Dhawan	35.072539	123.454313
36	SA Yadav	31.805310	142.505948
37	SE Marsh	39.507937	130.109775
38	SK Raina	32.374269	132.535312
39	SPD Smith	34.652778	124.812406
40	SR Tendulkar	33.826087	114.187867
41	SR Watson	30.793651	134.163209
42	SS Iyer	31.948980	123.025540
43	SV Samson	30.687500	135.137615
44	Shubman Gill	37.835294	132.236842
45	V Kohli	38.714976	128.511867
46	V Sehwag	27.555556	148.827059
47	WP Saha	24.247934	123.902027
48	YK Pathan	29.290909	138.046272
49	Yuvraj Singh	24.810811	124.784776

```
In [88]: nout
```

```
Out[88]: RG Sharma        223
         V Kohli          207
         S Dhawan         193
         KD Karthik       184
         RV Uthappa       180
         SK Raina         171
         DA Warner        164
         AT Rayudu        155
         AM Rahane        154
         SV Samson        144
         G Gambhir        136
         MS Dhoni         134
         MK Pandey        133
         AB de Villiers   130
         F du Plessis     127
         PA Patel         126
         SR Watson        126
         CH Gayle         126
         WP Saha          121
         KA Pollard       121
         MA Agarwal       117
         SA Yadav         113
         GJ Maxwell       112
         Yuvraj Singh     111
         YK Pathan        110
         RA Jadeja        108
         KL Rahul         105
         BB McCullum      104
         Q de Kock        102
         M Vijay          101
         V Sehwag          99
         SS Iyer           98
         JC Buttler        95
         N Rana            93
         Ishan Kishan      93
         RR Pant           93
         HH Pandya         89
         AD Russell        86
         Shubman Gill      85
         JH Kallis         85
         AJ Finch          84
         DR Smith          84
         RA Tripathi       83
         DA Miller         82
         R Dravid          77
         SPD Smith         72
         SR Tendulkar      69
         SE Marsh          63
         KS Williamson     60
         RD Gaikwad        57
         Name: player_dismissed, dtype: int64
```

Scatter plot

In [89]:
```python
import plotly.graph_objs as go
import plotly.offline as pyo
```

In [90]:
```python
# Create scatter plot trace
trace_plot = go.Scatter(x=avg['avg'], y=avg['batsman_runs'], mode='markers')

# Data for the plot
data = [trace_plot]

# Define layout for the plot
layout = go.Layout(title='Batsman Avg vs SR',
                   xaxis={'title': 'Batsman Average'},
                   yaxis={'title': 'Strike Rate'})

# Create figure
fig = go.Figure(data=data, layout=layout)


pyo.plot(fig)
```

Out[90]:
```
'temp-plot.html'
```

In [91]:
```python
trace_plot = go.Scatter(
    x=avg['avg'],
    y=avg['batsman_runs'],
    mode='markers',
    marker=dict(
        size=12,                # Adjust size of markers
        color=avg['batsman_runs'],  # Color based on runs
        colorscale='Viridis',    # Apply color scale
        showscale=True           # Show color scale bar
    )
)
```

In [92]:
```python
layout = go.Layout(title='Batsman Avg vs SR',
                   xaxis={'title': 'Batsman Average'},
                   yaxis={'title': 'Strike Rate'})

# Create figure
fig = go.Figure(data=data, layout=layout)


pyo.plot(fig)
```

Out[92]:
```
'temp-plot.html'
```

In [93]:
```python
trace_plot = go.Scatter(
    x=avg['avg'],
    y=avg['batsman_runs'],
    mode='markers',
    marker=dict(size=12, color=avg['batsman_runs'], colorscale='Viridis', sh
    text=avg['batter'],  # Show player name on hover
    hoverinfo='text+x+y'      # Display both x and y values along with text
)
```

In [94]:
```
layout = go.Layout(title='Batsman Avg vs SR',
                   xaxis={'title': 'Batsman Average'},
                   yaxis={'title': 'Strike Rate'})

# Create figure
fig = go.Figure(data=data, layout=layout)


pyo.plot(fig)
```

Out[94]:
```
'temp-plot.html'
```

In [95]:
```
pyo.iplot(fig)
```

In [96]:
```
annotations = [
    dict(
        x=avg['avg'][idx],
        y=avg['batsman_runs'][idx],
        text=f"Top Player: {avg['batter'][idx]}",
        showarrow=True,
        arrowhead=2,
        ax=20,
        ay=-30
    ) for idx in [0, 1, 2]  # Highlight first three top players
]
layout = go.Layout(
    title='Batsman Avg vs SR',
    xaxis={'title': 'Batsman Average'},
    yaxis={'title': 'Strike Rate'},
    annotations=annotations   # Add annotations to layout
)
```

In [102…
```
# Create figure
fig = go.Figure(data=data, layout=layout)


pyo.plot(fig)
```

Out[102…
```
'temp-plot.html'
```

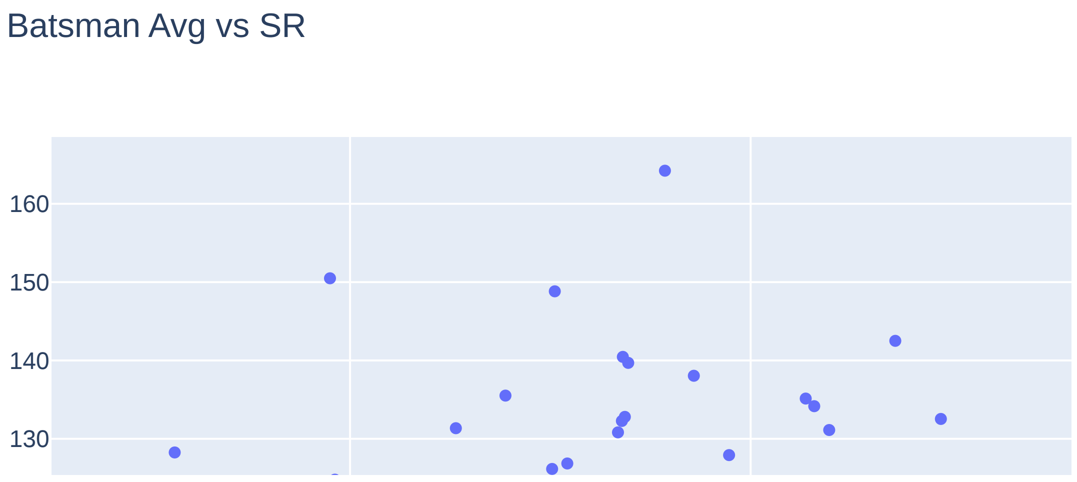
In [103…
```
pyo.iplot(fig)
```

In [99]:
```
layout = go.Layout(
    title='Batsman Avg vs SR',
    xaxis=dict(
        title='Batsman Average',
        rangeselector=dict(
            buttons=list([
                dict(count=1, label="1m", step="month", stepmode="backward")
                dict(count=6, label="6m", step="month", stepmode="backward")
                dict(step="all")
            ])
        ),
        rangeslider=dict(visible=True),  # Add a range slider
    )
)
```

In [100…
```
# Create figure
fig = go.Figure(data=data, layout=layout)


pyo.plot(fig)
```

Out[100…
```
'temp-plot.html'
```

In [101…
```
pyo.iplot(fig)
```



year by performance

In [102…
```
single= ipl[ipl['batter']== 'V Kohli']
performance= single.groupby('season')['batsman_runs'].sum().reset_index()
performance
```

Out[102…

	season	batsman_runs
0	2007/08	165
1	2009	246
2	2009/10	307
3	2011	557
4	2012	364
5	2013	639
6	2014	359
7	2015	505
8	2016	973
9	2017	308
10	2018	530
11	2019	464
12	2020/21	471
13	2021	405
14	2022	341
15	2023	639
16	2024	741

In [103…
```
single= ipl[ipl['batter']== 'F du Plessis']
performance1= single.groupby('season')['batsman_runs'].sum().reset_index()
performance1
```

Out[103…

	season	batsman_runs
0	2012	398
1	2014	303
2	2015	380
3	2016	206
4	2017	8
5	2018	162
6	2019	396
7	2020/21	449
8	2021	633
9	2022	468
10	2023	730
11	2024	438

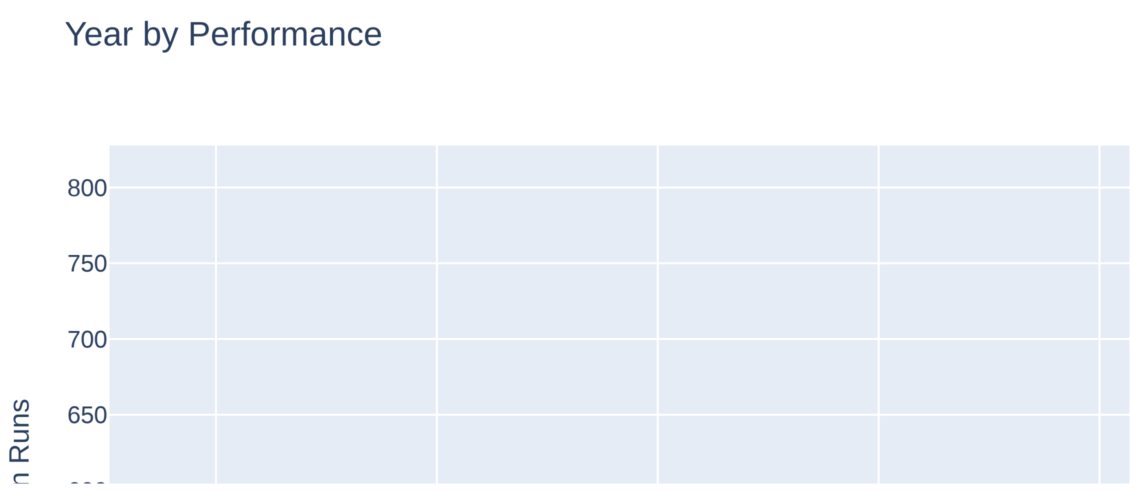
PLOT LINE CHART

In [104…

```
trace = go.Scatter(
    x=performance['season'],
    y=performance['batsman_runs'],
    mode='lines+markers',
    marker=dict(color='#00a65a')
)

data=[trace]

layout = go.Layout(
    title='Year by Performance',
    xaxis={'title': 'Season'},
    yaxis={'title': 'Batsman Runs'}
)

# Create figure
fig1 = go.Figure(data=data, layout=layout)
```

In [105…

```
# Plot the figure
pyo.iplot(fig1)  # Use this for Jupyter Notebooks


# or
pyo.plot(fig1)  # Use this for a standalone HTML file
```



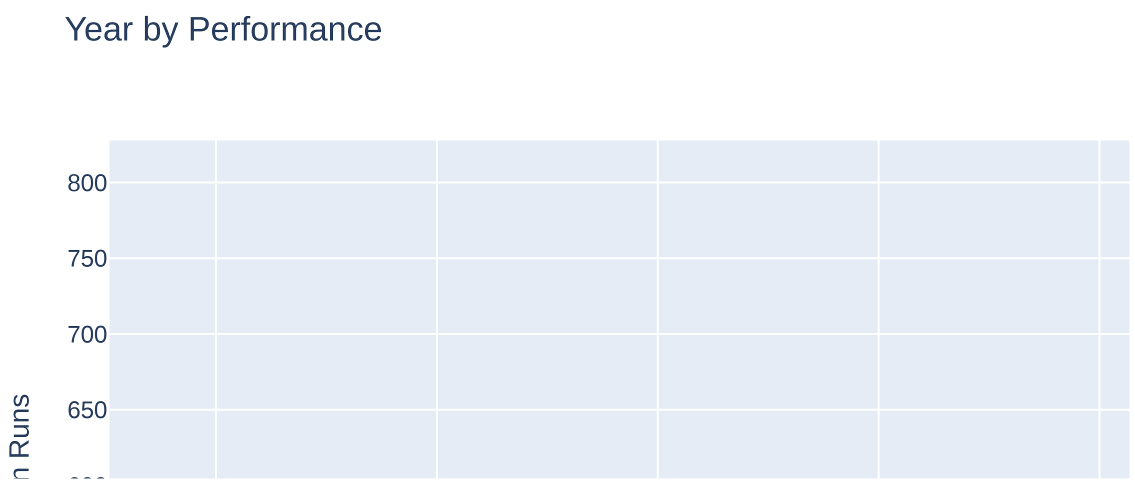
```
Out[105… 'temp-plot.html'
```

```
In [106…
```

```
performance = {
    'season': [2018, 2019, 2020, 2021, 2022],
    'batsman_runs': [500, 600, 400, 700, 800]
}
```

```
In [107…
```

```
# Sample performance data (replace this with your actual DataFrame)
performance = {
    'season': [2018, 2019, 2020, 2021, 2022],
    'batsman_runs': [500, 600, 400, 700, 800]
}

trace = go.Scatter(
    x=performance['season'],
    y=performance['batsman_runs'],
    mode='lines+markers',
    marker=dict(color='#00a65a')
)

data = [trace]

layout = go.Layout(
    title='Year by Performance',
    xaxis=dict(title='Season'),
    yaxis=dict(title='Batsman Runs')
)

# Create figure
fig = go.Figure(data=data, layout=layout)


# Plot the figure
pyo.iplot(fig)  # Use this for Jupyter Notebooks


# or
pyo.plot(fig)  # Use this for a standalone HTML file
```



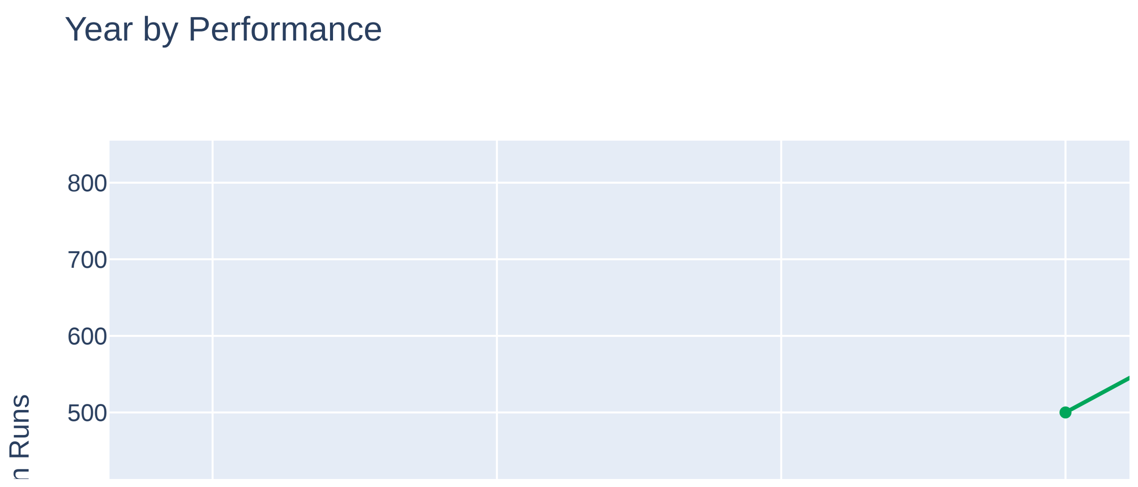
Out[107… 'temp-plot.html'

multi line chart

In [108…
```
trace = go.Scatter(
    x=performance['season'],
    y=performance['batsman_runs'],
    mode='lines+markers',
    marker=dict(color='#00a65a')
)

trace1 = go.Scatter(
    x=performance1['season'],
    y=performance1['batsman_runs'],
    mode='lines+markers'
)

# Put both traces in the data list
data = [trace, trace1]

layout = go.Layout(
    title='Year by Performance',
    xaxis={'title': 'Season'},
    yaxis={'title': 'Batsman Runs'}
)

# Create figure
fig2 = go.Figure(data=data, layout=layout)
```

In [109…

```
# Plot the figure
pyo.iplot(fig2)  # Use this for Jupyter Notebooks


# or
pyo.plot(fig2)  # Use this for a standalone HTML file
```



```
Out[109… 'temp-plot.html'
```

bar plot

```
In [110… top10 = ipl.groupby('batter')['batsman_runs'].sum().sort_values(ascending=Fa
         top10_df=ipl[ipl['batter'].isin(top10)]
```

```
In [111… top10_df
```

Out[111…

	match_id	inning	batting_team	bowling_team	over	ball	batter
1	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	2	BB McCullum
2	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	3	BB McCullum
3	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	4	BB McCullum
4	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	5	BB McCullum
5	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	0	6	BB McCullum
...	...	...	...	...	...	...	...
260763	1426312	1	Sunrisers Hyderabad	Kolkata Knight Riders	4	1	RA Tripathi
260764	1426312	1	Sunrisers Hyderabad	Kolkata Knight Riders	4	2	RA Tripathi
260910	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	8	6	SS Iyer
260915	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	9	5	SS Iyer
260918	1426312	2	Kolkata Knight Riders	Sunrisers Hyderabad	10	2	SS Iyer

139298 rows × 37 columns

In [113…

```python
import plotly.graph_objects as go

trace = go.Bar(
    x=top10_df['batter'],
    y=top10_df['batsman_runs']
)

data = [trace]

layout = go.Layout(
    title='Top 10 IPL Batsmen',
    xaxis={'title': 'Batsman Name'},
    yaxis={'title': 'Batsman Runs'}
)
```

```
fig3 = go.Figure(data=data, layout=layout)
```

In [ ]:
```
# Plot the figure
pyo.iplot(fig3)  # Use this for Jupyter Notebooks


# or
pyo.plot(fig3)
```

Add Colors Based on Values

In [114…
```
trace = go.Bar(
    x=top10_df['batter'],
    y=top10_df['batsman_runs'],
    marker=dict(
        color=top10_df['batsman_runs'],  # Color based on the number of runs
        colorscale='Viridis',
        showscale=True  # Display color scale bar
    )
)
```

Add Annotations to Highlight Key Performers

In [46]:
```
annotations = [
    dict(
        x='Virat Kohli',  # Input  Batsman's name
        y=973,  # Runs scored
        xref='x',
        yref='y',
        text='Highest Scorer',  # Annotation text
        showarrow=True,
        arrowhead=7,
        ax=0,
        ay=-40
    )
]

layout = go.Layout(
    title='Top 10 IPL Batsmen by Runs',
    xaxis={'title': 'Batsman Name'},
    yaxis={'title': 'Batsman Runs'},
    annotations=annotations  # Add the annotations
)
```

In [47]:
```
fig6 = go.Figure(data=data, layout=layout)
```

In [48]:
```
pyo.plot(fig6)
```

```
Out[48]: 'temp-plot.html'
```

Add a Range Slider to Zoom into Specific Data

```
In [98]: layout = go.Layout(
             title='Top 10 IPL Batsmen by Runs',
             xaxis=dict(
                 title='Batsman Name',
                 rangeslider=dict(visible=True),  # Add range slider for x-axis
                 type='category'  # Set type as category for non-numeric x-axis
             ),
             yaxis={'title': 'Batsman Runs'}
         )
```

Stacked Bar Chart to Show Multiple Metrics

```
In [99]: trace1 = go.Bar(
             x=top10_df['batter'],
             y=top10_df['batsman_runs'],
             name='Runs',
             marker=dict(color='blue')
         )

         trace2 = go.Bar(
             x=top10_df['batter'],
             y=top10_df['ball'],
             name='ball',
             marker=dict(color='green')
         )

         data = [trace1, trace2]

         layout = go.Layout(
             title='Top 10 IPL Batsmen: Runs vs Strike Rate',
             xaxis={'title': 'Batsman Name'},
             yaxis={'title': 'Metrics'},
             barmode='stack'  # Stack the bars
         )
```

```
In [100… fig7 = go.Figure(data=data, layout=layout)
```

```
In [101… pyo.plot(fig7)
```

```
Out[101… 'temp-plot.html'
```

Grouped Bar Chart for Comparison

In [102…
```
trace1 = go.Bar(
    x=top10_df['batter'],
    y=top10_df['batsman_runs'],
    name='2021 Season',
    marker=dict(color='blue')
)

trace2 = go.Bar(
    x=top10_df['batter'],
    y=top10_df['batsman_runs'],
    name='2022 Season',
    marker=dict(color='green')
)

data = [trace1, trace2]

layout = go.Layout(
    title='Top 10 IPL Batsmen by Season',
    xaxis={'title': 'Batsman Name'},
    yaxis={'title': 'Batsman Runs'},
    barmode='group'  # Grouped bars
)
```

In [103…
```
fig8 = go.Figure(data=data, layout=layout)
```

In [104…
```
pyo.plot(fig8)
```

Out[104…
```
'temp-plot.html'
```

BUBBLE CHART

In [105…
```
new_ipl= new_ipl[new_ipl['batsman_runs']== 6]
```

In [106…
```
new_ipl
```

Out[106…

	match_id	inning	batting_team	bowling_team	over	ball	batter
10	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	1	4	BB McCullum
20	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	3	2	BB McCullum
25	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	3	7	BB McCullum
60	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	9	5	BB McCullum
69	335982	1	Kolkata Knight Riders	Royal Challengers Bangalore	11	2	BB McCullum
...	...	...	...	...	...	...	...
260254	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	1	5	F du Plessis
260262	1426310	1	Royal Challengers Bengaluru	Rajasthan Royals	3	1	V Kohli
260402	1426310	2	Rajasthan Royals	Royal Challengers Bengaluru	6	1	SV Samson
260506	1426311	1	Sunrisers Hyderabad	Rajasthan Royals	3	4	RA Tripathi
260509	1426311	1	Sunrisers Hyderabad	Rajasthan Royals	4	1	RA Tripathi

7091 rows × 37 columns

In [107…
```
six= new_ipl.groupby('batter')['batsman_runs'].count().reset_index()
```

In [108…
```
six
```

Out[108…

	batter	batsman_runs
0	AB de Villiers	253
1	AD Russell	209
2	AJ Finch	78
3	AM Rahane	103
4	AT Rayudu	173
5	BB McCullum	130
6	CH Gayle	359
7	DA Miller	134
8	DA Warner	236
9	DR Smith	117
10	F du Plessis	166
11	G Gambhir	59
12	GJ Maxwell	160
13	HH Pandya	137
14	Ishan Kishan	119
15	JC Buttler	161
16	JH Kallis	44
17	KA Pollard	224
18	KD Karthik	161
19	KL Rahul	187
20	KS Williamson	64
21	M Vijay	91
22	MA Agarwal	98
23	MK Pandey	111
24	MS Dhoni	252
25	N Rana	132
26	PA Patel	49
27	Q de Kock	123
28	R Dravid	28
29	RA Jadeja	107
30	RA Tripathi	84
31	RD Gaikwad	91
32	RG Sharma	281

	batter	batsman_runs
33	RR Pant	154
34	RV Uthappa	182
35	S Dhawan	153
36	SA Yadav	130
37	SE Marsh	78
38	SK Raina	204
39	SPD Smith	60
40	SR Tendulkar	29
41	SR Watson	190
42	SS Iyer	113
43	SV Samson	206
44	Shubman Gill	95
45	V Kohli	273
46	V Sehwag	106
47	WP Saha	87
48	YK Pathan	161
49	Yuvraj Singh	149

In [109…

```
x=avg.merge(six, on= 'batter')
x
```

Out[109…

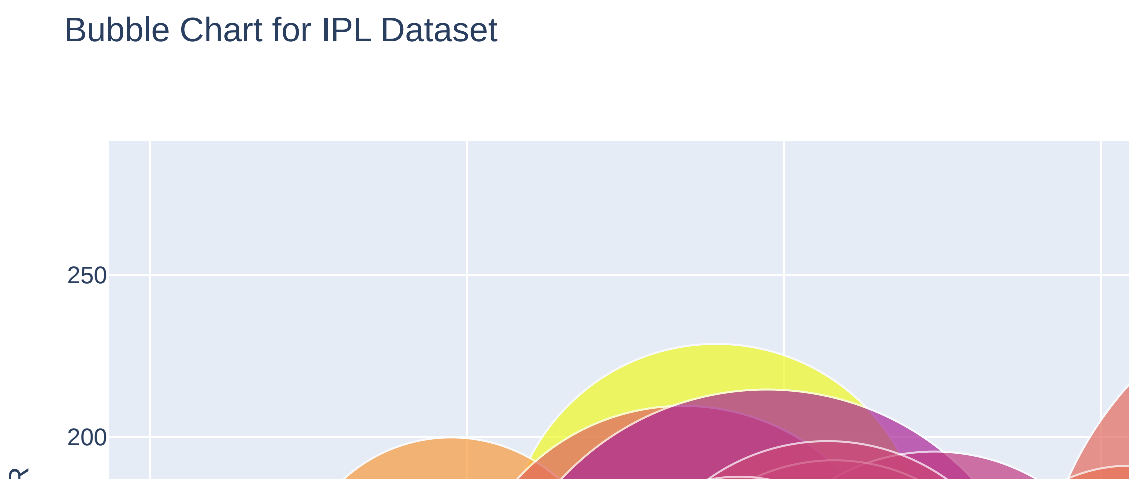
	batter	avg	batsman_runs_x	batsman_runs_y
0	AB de Villiers	39.853846	148.580442	253
1	AD Russell	28.930233	164.224422	209
2	AJ Finch	24.904762	123.349057	78
3	AM Rahane	30.142857	120.321410	103
4	AT Rayudu	28.051613	124.584527	173
5	BB McCullum	27.711538	126.848592	130
6	CH Gayle	39.658730	142.121729	359
7	DA Miller	35.658537	134.684477	134
8	DA Warner	40.042683	135.429986	236
9	DR Smith	28.392857	132.279534	117
10	F du Plessis	35.992126	133.071325	166
11	G Gambhir	31.007353	119.665153	59
12	GJ Maxwell	24.750000	150.488599	160
13	HH Pandya	28.471910	139.691290	137
14	Ishan Kishan	28.430108	132.797589	119
15	JC Buttler	37.715789	142.238984	161
16	JH Kallis	28.552941	105.936272	44
17	KA Pollard	28.404959	140.457703	224
18	KD Karthik	26.320652	131.353404	161
19	KL Rahul	44.657143	131.050866	187
20	KS Williamson	35.533333	122.952710	64
21	M Vijay	25.930693	118.614130	91
22	MA Agarwal	22.811966	128.255646	98
23	MK Pandey	29.015038	117.366180	111
24	MS Dhoni	39.126866	132.835065	252
25	N Rana	28.344086	130.818859	132
26	PA Patel	22.603175	116.625717	49
27	Q de Kock	30.980392	131.120332	123
28	R Dravid	28.233766	113.347237	28
29	RA Jadeja	27.398148	124.432296	107
30	RA Tripathi	26.939759	135.515152	84
31	RD Gaikwad	41.754386	133.632791	91
32	RG Sharma	29.730942	127.918194	281

	batter	avg	batsman_runs_x	batsman_runs_y
33	RR Pant	35.451613	143.597561	154
34	RV Uthappa	27.522222	126.152279	182
35	S Dhawan	35.072539	123.454313	153
36	SA Yadav	31.805310	142.505948	130
37	SE Marsh	39.507937	130.109775	78
38	SK Raina	32.374269	132.535312	204
39	SPD Smith	34.652778	124.812406	60
40	SR Tendulkar	33.826087	114.187867	29
41	SR Watson	30.793651	134.163209	190
42	SS Iyer	31.948980	123.025540	113
43	SV Samson	30.687500	135.137615	206
44	Shubman Gill	37.835294	132.236842	95
45	V Kohli	38.714976	128.511867	273
46	V Sehwag	27.555556	148.827059	106
47	WP Saha	24.247934	123.902027	87
48	YK Pathan	29.290909	138.046272	161
49	Yuvraj Singh	24.810811	124.784776	149

In [110…

```
import plotly.graph_objects as go
import plotly.offline as pyo
pyo.init_notebook_mode(connected=True)

# Create a bubble chart
trace = go.Scatter(
    x=x['avg'],
    y=x['batsman_runs_x'],
    mode='markers',  # Display markers
    marker=dict(
        size=x['batsman_runs_y'],  # Size of markers
        color=x['batsman_runs_x'],  # Color by another dimension (optional)
        showscale=True  # Show color scale
    )
)

# Create layout
layout = go.Layout(
    title='Bubble Chart for IPL Dataset',
    xaxis={'title': 'Average'},
    yaxis={'title': 'SR'}
)

# Create figure
fig14 = go.Figure(data=[trace], layout=layout)
```

```
In [111…  fig14.show()
```



```
In [112…  pyo.plot(fig14)
```

```
Out[112…  'temp-plot.html'
```

```
In [112…
import plotly.graph_objects as go
import plotly.offline as pyo
pyo.init_notebook_mode(connected=True)

# Create a bubble chart using the 'batter' names and 'batsman_runs'
trace = go.Scatter(
    x=top10_df['batter'],
    y=top10_df['batsman_runs'],
    mode='markers',
    marker=dict(
        size=top10_df['batsman_runs'] / 10,
        color=top10_df['batsman_runs'],
        showscale=True
    )
)

layout = go.Layout(
    title='Top 10 IPL Batsmen - Bubble Chart',
    xaxis={'title': 'Batsman Name'},
    yaxis={'title': 'Batsman Runs'},
    hovermode='closest'
)

fig11 = go.Figure(data=[trace], layout=layout)
```

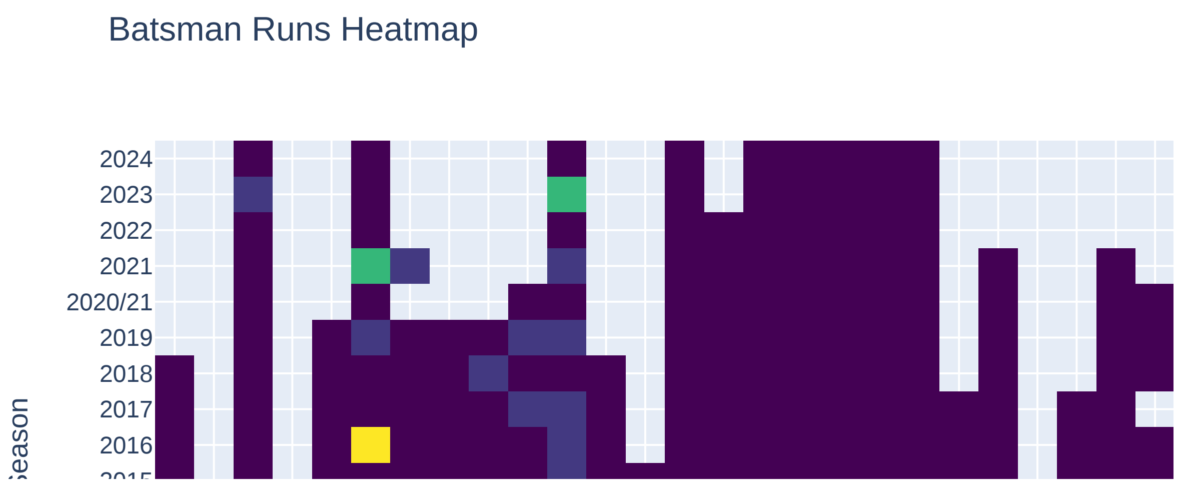
In [75]:
```
pyo.plot(fig11)
```

Out[75]:
```
'temp-plot.html'
```

HEATMAP

In [113…
```
import plotly.graph_objects as go

# Create a heatmap (assuming you have more data, like batsman names and mult
trace = go.Heatmap(
    z=top10_df['batsman_runs'],  # Data to color by
    x=top10_df['batter'],
    y=top10_df['season'],        # Seasons (if available in dataset)
    colorscale='Viridis'         # Choose a color scale
)

layout = go.Layout(
    title='Batsman Runs Heatmap',
    xaxis={'title': 'Batsman'},
    yaxis={'title': 'Season'},
)

fig12 = go.Figure(data=[trace], layout=layout)
```

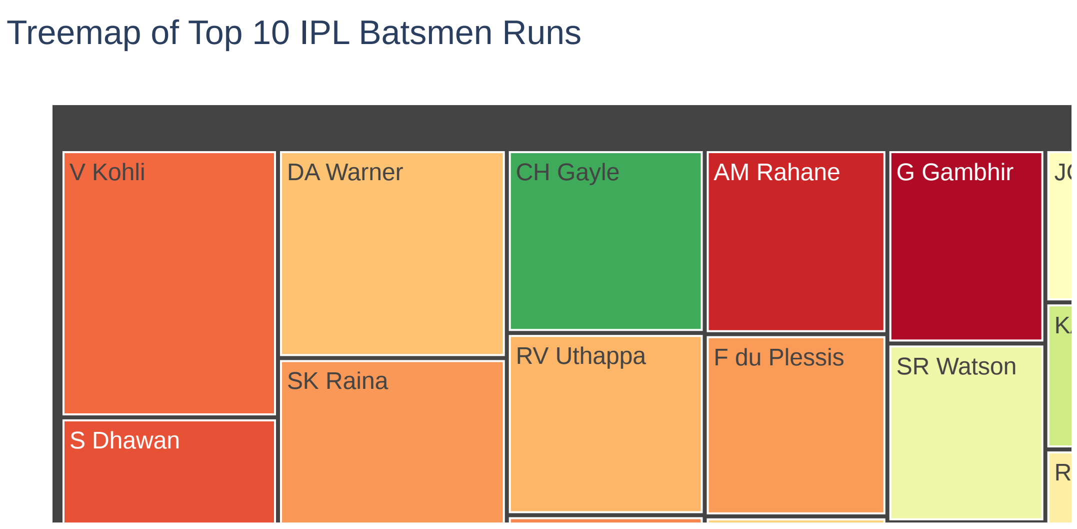
In [114…
```
fig12.show()
```



```
In [117…  pyo.plot(fig12)

Out[117…  'temp-plot.html'
```

TREEMAP

```
In [115…  import plotly.express as px

          # Create a treemap (assuming hierarchical data, use 'batter' and 'batsman_ru
          fig = px.treemap(
              top10_df,
              path=['batter'],           # Path for hierarchy (just batter here)
              values='batsman_runs',     # Values for sizing
              color='batsman_runs',      # Color by runs
              color_continuous_scale='RdYlGn'  # Color scale
          )

          fig.update_layout(title='Treemap of Top 10 IPL Batsmen Runs')
          fig.show()
```



BOX PLOT

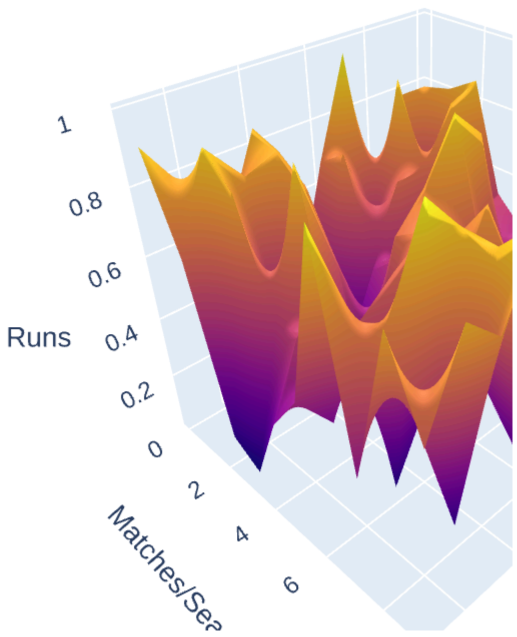
```
In [ ]: trace = go.Box(
            y=top10_df['batsman_runs'],
            boxpoints='all',  # Show all points
            jitter=0.3,       # Spread the points
            pointpos=-1.8,    # Offset points to the left
            marker_color='blue'
        )

        layout = go.Layout(
            title='Box Plot of Batsman Runs',
            yaxis={'title': 'Batsman Runs'}
        )

        fig = go.Figure(data=[trace], layout=layout)
        pyo.plot(fig)
```

3D Surface Plot

In [116…

```python
# Example with hypothetical data
z_data = np.random.rand(10, 10)

trace = go.Surface(z=z_data) # creates a 3D surface plot
layout = go.Layout(
    title='3D Surface Plot of IPL Data',
    scene = dict(                           #defines the 3D plot’s axes:
        xaxis_title='Batter',
        yaxis_title='Matches/Seasons',
        zaxis_title='Runs'
    )
)

fig = go.Figure(data=[trace], layout=layout)
pyo.iplot(fig)
```

3D Surface Plot of IPL Data



z_data is a 10x10 array of random numbers between 0 and 1. This data represents the z values (the height values in the 3D plot)

Funnel Chart

In [ ]:
```
import plotly.express as px

# Create a funnel chart to show progressive runs by batsmen
fig = px.funnel(
    top10_df,
    x='batsman_runs',  # Values for the funnel stages
    y='batter'         # Names for the funnel stages
)

fig.update_layout(title='Funnel Chart of Batsman Runs')
fig.show()
```

In [ ]:
```
pyo.plot(fig)
```

Waterfall Chart

A waterfall chart shows the cumulative effect of sequentially introduced positive or negative values

In [86]:
```
import plotly.graph_objects as go

trace = go.Waterfall(
    x=top10_df['batter'],
    y=top10_df['batsman_runs'],
    connector={"line":{"color":"rgb(63, 63, 63)"}}
)

layout = go.Layout(
    title='Waterfall Chart of Batsman Runs',
    xaxis={'title': 'Batter'},
    yaxis={'title': 'Cumulative Runs'}
)

fig = go.Figure(data=[trace], layout=layout)
```

In [87]:
```
pyo.plot(fig)
```

Out[87]:
```
'temp-plot.html'
```

In [ ]:
```
fig.show()
```

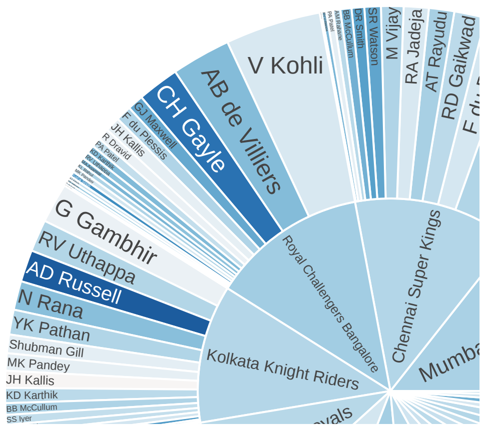
Sunburst Chart

Sunburst charts are used to represent hierarchical data.

In [117…

```
# Create a sunburst chart with the correct column names
fig = px.sunburst(
    top10_df,
    path=['batting_team', 'batter'],  # Create hierarchy (team -> batter)
    values='batsman_runs',  # Values to be represented in the chart
    color='batsman_runs',  # Color by number of runs
    color_continuous_scale='RdBu'
)

fig.update_layout(title='Sunburst Chart of Batsman Runs by Team')
fig.show()
```

Sunburst Chart of Batsman Runs by Team



In [81]:

```
pyo.plot(fig)
```

```
Out[81]: 'temp-plot.html'
```

Violin Plot

A violin plot is useful to visualize the distribution of the data and its probability density.

```
In [ ]: import plotly.express as px

        # Create a violin plot for batsman runs distribution
        fig = px.violin(
            top10_df,
            y='batsman_runs',
            box=True,  # Add a box plot inside the violin
            points='all'  # Show all points
        )

        fig.update_layout(title='Violin Plot of Batsman Runs')
        fig.show()
```

```
In [ ]: pyo.plot(fig)
```

update_layout: This method allows you to update the layout of the figure, including aspects such as the title, axis labels, colors, and more.

Polar Chart

Polar charts are useful for visualizing data in circular layouts

```
In [82]: import plotly.express as px

         # Create a polar chart
         fig = px.line_polar(
             top10_df,
             r='batsman_runs',  # Radial axis (runs)
             theta='batter',    # Angular axis (batsmen)
```

```
    line_close=True,   # Close the line to form a loop
    template="plotly_dark"
)

fig.update_layout(title='Polar Chart of Top 10 IPL Batsmen by Runs')
fig.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_28536\864864616.py in <module>
      2 
      3 # Create a polar chart
----> 4 fig = px.line_polar(
      5     top10_df,
      6     r='batsman_runs',  # Radial axis (runs)

~\anaconda3\lib\site-packages\plotly\express\_chart_types.py in line_polar(d
ata_frame, r, theta, color, line_dash, hover_name, hover_data, custom_data,
line_group, text, symbol, animation_frame, animation_group, category_orders,
labels, color_discrete_sequence, color_discrete_map, line_dash_sequence, lin
e_dash_map, symbol_sequence, symbol_map, markers, direction, start_angle, li
ne_close, line_shape, render_mode, range_r, range_theta, log_r, title, templ
ate, width, height)
    996     of a polyline mark in polar coordinates.
    997     """
--> 998     return make_figure(args=locals(), constructor=go.Scatterpolar)
    999 
   1000 

~\anaconda3\lib\site-packages\plotly\express\_core.py in make_figure(args, c
onstructor, trace_patch, layout_patch)
   2298                     args, trace_spec, group, mapping_labels.copy(), size
ref
   2299                 )
-> 2300                 trace.update(patch)
   2301                 if fit_results is not None:
   2302                     trendline_rows.append(mapping_labels.copy())

~\anaconda3\lib\site-packages\plotly\basedatatypes.py in update(self, dict1,
overwrite, **kwargs)
   5131                     BaseFigure._perform_update(self, kwargs, overwrite=o
verwrite)
   5132             else:
-> 5133                 BaseFigure._perform_update(self, dict1, overwrite=overwr
ite)
   5134                 BaseFigure._perform_update(self, kwargs, overwrite=overw
rite)
   5135 

~\anaconda3\lib\site-packages\plotly\basedatatypes.py in _perform_update(plo
tly_obj, update_obj, overwrite)
   3911                         # Update compound objects recursively
   3912                         # plotly_obj[key].update(val)
-> 3913                         BaseFigure._perform_update(plotly_obj[key], val)
   3914                     elif isinstance(validator, CompoundArrayValidator):
   3915                         if plotly_obj[key]:

~\anaconda3\lib\site-packages\plotly\basedatatypes.py in _perform_update(plo
tly_obj, update_obj, overwrite)
   3888                     err = _check_path_in_prop_tree(plotly_obj, key, erro
r_cast=ValueError)
   3889                     if err is not None:
-> 3890                         raise err
```

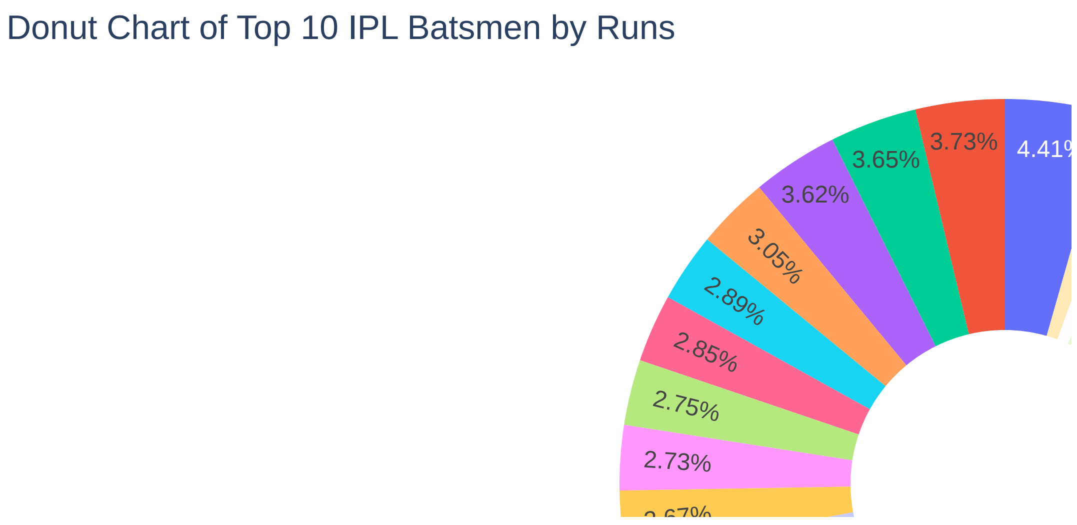
```
   3891
   3892             # Convert update_obj to dict

ValueError: Invalid property specified for object of type plotly.graph_objs.
scatterpolargl.Line: 'shape'

Did you mean "dash"?

    Valid properties:
        color
            Sets the line color.
        dash
            Sets the style of the lines.
        width
            Sets the line width (in px).

Did you mean "dash"?

Bad property path:
shape
^^^^^
```

Donut Chart (Pie Chart Variation)

Donut charts are variations of pie charts where the center is hollow, allowing we can add an additional layer of information.

In [118…

```
# Create a donut chart to represent the percentage of total runs by each bat
fig = px.pie(
    top10_df,
    names='batter',
    values='batsman_runs',
    hole=0.4  # Hollow center to create a donut chart
)

fig.update_layout(title='Donut Chart of Top 10 IPL Batsmen by Runs')
```

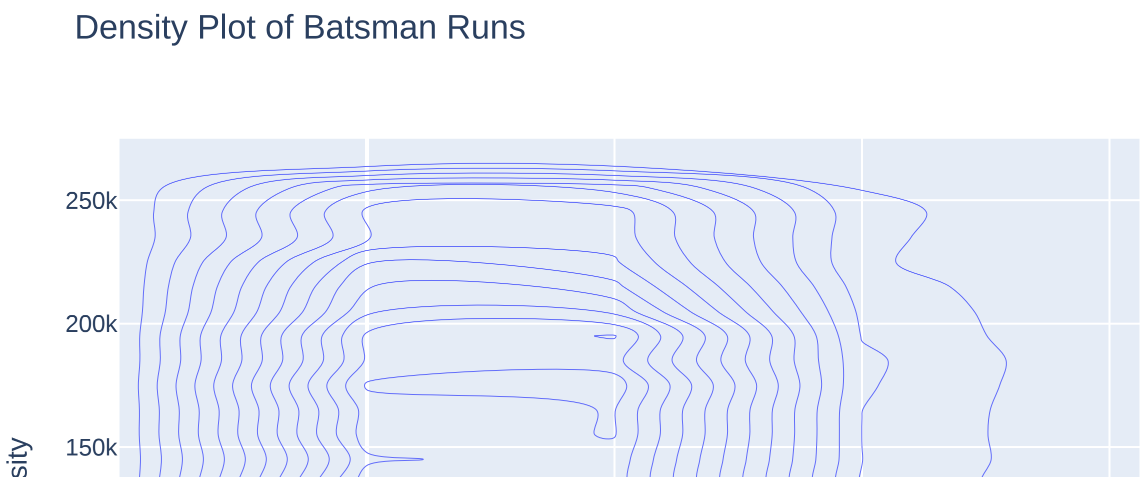


```
In [ ]: pyo.plot(fig)
```

Density Plot

A density plot (or Kernel Density Estimate) provides a smoothed estimate of the distribution.

```
In [119…
# Create a density plot
fig = px.density_contour(
    top10_df,
    x='batsman_runs',  # Variable for the x-axis
    title='Density Plot of Batsman Runs'
)

fig.update_layout(
    xaxis_title='Batsman Runs',
    yaxis_title='Density'
)

fig.show()
```



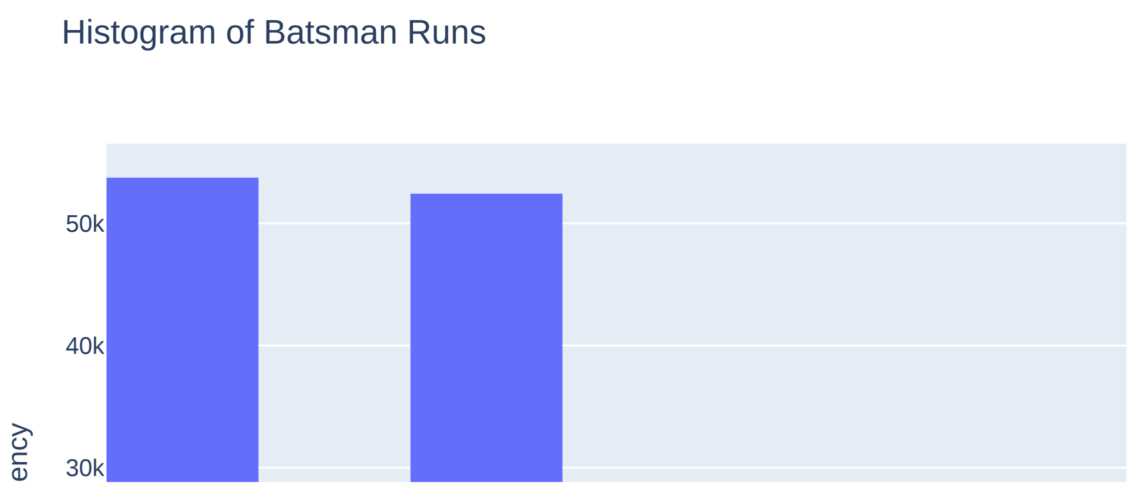
In [85]:
```
pyo.plot(fig)
```

Out[85]:
```
'temp-plot.html'
```

HISTOGRAM

In [120…
```
# Create a histogram
fig = px.histogram(
    top10_df,
    x='batsman_runs',  # Variable for the x-axis
    nbins=20,          # Number of bins
    title='Histogram of Batsman Runs'
)

fig.update_layout(
    xaxis_title='Batsman Runs',
    yaxis_title='Frequency'
)
```

```
fig.show()
```



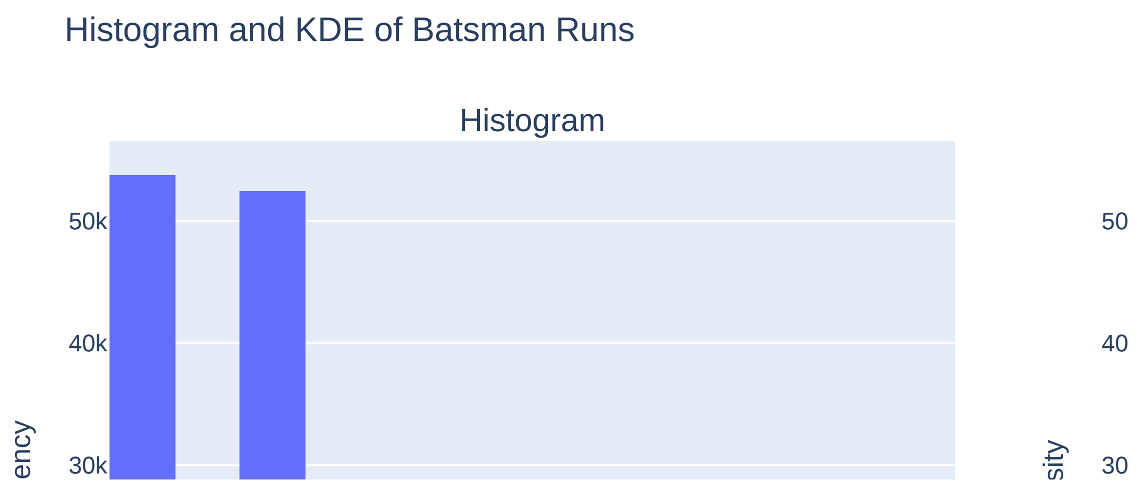
In [71]:
```
pyo.plot(fig)
```

Out[71]:
```
'temp-plot.html'
```

Combined Histogram and KDE Plot

In [123…
```
import plotly.graph_objects as go
import plotly.subplots as sp

# Create subplots
fig = sp.make_subplots(rows=1, cols=2, subplot_titles=('Histogram', 'KDE'))

# Add Histogram
histogram = go.Histogram(
    x=top10_df['batsman_runs'],
    nbinsx=20,
    name='Histogram'
)
fig.add_trace(histogram, row=1, col=1)
```

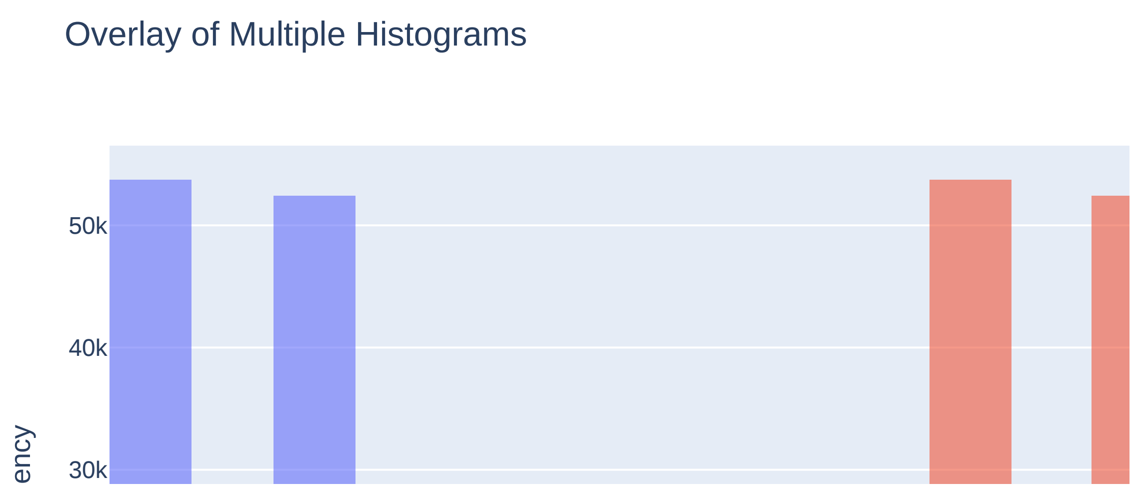
```
# Add KDE
kde = go.Histogram(
    x=top10_df['batsman_runs'],
    histnorm='density',
    name='KDE'
)
fig.add_trace(kde, row=1, col=2)

fig.update_layout(
    title='Histogram and KDE of Batsman Runs',
    xaxis_title='Batsman Runs',
    yaxis_title='Frequency',
    xaxis2_title='Batsman Runs',
    yaxis2_title='Density'
)

fig.show()
```



Overlay Multiple Histograms

In [125…

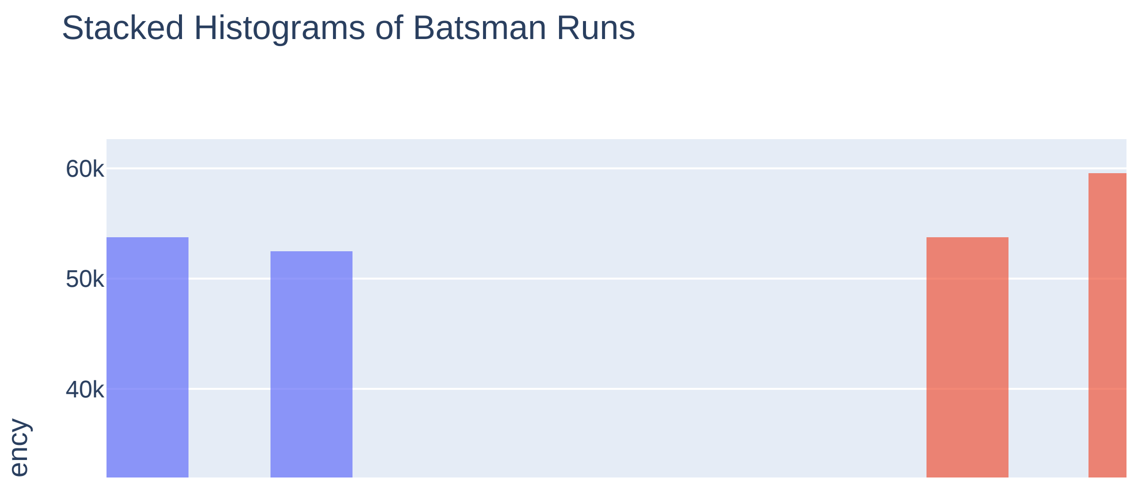
```
# Sample data for overlay
top10_df2 = top10_df.copy()
top10_df2['batsman_runs'] = top10_df2['batsman_runs'] + 5  # Modify data for

fig = go.Figure()

fig.add_trace(go.Histogram(
    x=top10_df['batsman_runs'],
    nbinsx=30,
    opacity=0.6,
    name='Dataset 1'
))

fig.add_trace(go.Histogram(
    x=top10_df2['batsman_runs'],
    nbinsx=30,
    opacity=0.6,
    name='Dataset 2'
))

fig.update_layout(
    title='Overlay of Multiple Histograms',
    xaxis_title='Batsman Runs',
    yaxis_title='Frequency',
    barmode='overlay'  # Overlay bars
)

fig.show()
```



STACKED HISTOGRAM

In [126…

```
# Sample data for stacked histograms
fig = go.Figure()

fig.add_trace(go.Histogram(
    x=top10_df['batsman_runs'],
    nbinsx=30,
    name='Dataset 1',
    opacity=0.7
))

fig.add_trace(go.Histogram(
    x=top10_df2['batsman_runs'],
    nbinsx=30,
    name='Dataset 2',
    opacity=0.7
))

fig.update_layout(
    title='Stacked Histograms of Batsman Runs',
    xaxis_title='Batsman Runs',
    yaxis_title='Frequency',
    barmode='stack'  # Stack bars on top of each other
)

fig.show()
```

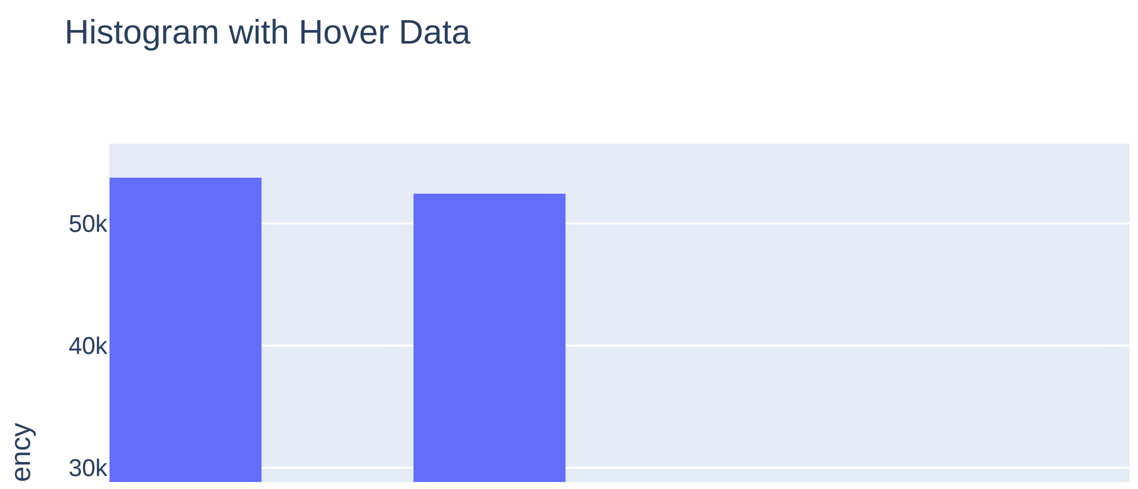


Histogram with Hover Data

In [127…

```
fig = go.Figure()

fig.add_trace(go.Histogram(
    x=top10_df['batsman_runs'],
    nbinsx=30,
    name='Runs Distribution',
    hoverinfo='x+y'  # Show x and y values on hover
))

fig.update_layout(
    title='Histogram with Hover Data',
    xaxis_title='Batsman Runs',
    yaxis_title='Frequency'
)
```

```
fig.show()
```

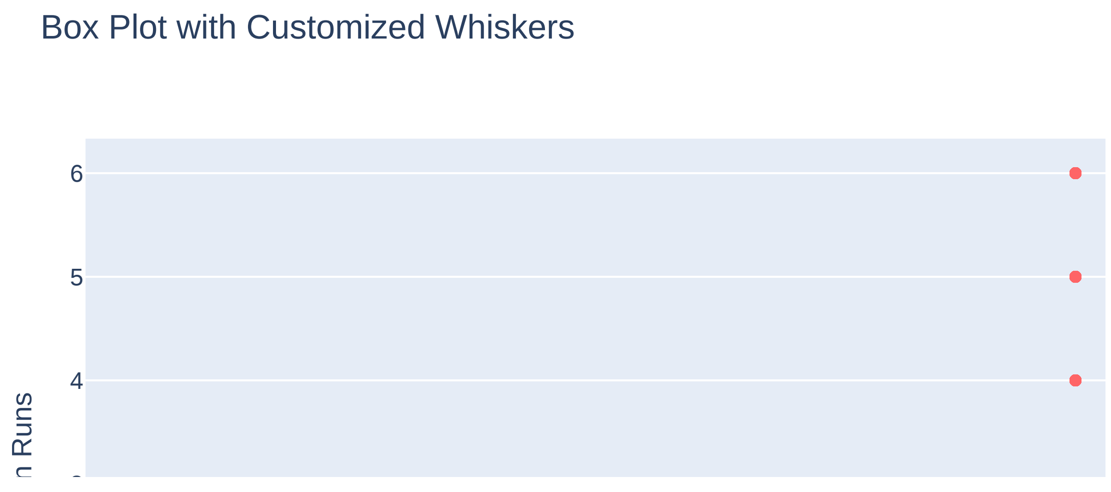


BOX PLOT WITH CUSTOMIZED WHISKERS

In [129…

```
# Box Plot with customized whiskers
fig = go.Figure()

fig.add_trace(go.Box(
    y=top10_df['batsman_runs'],
    name='Batsman Runs',
    boxmean='sd',
    whiskerwidth=0.4,  # Width of the whiskers
    line_color='rgba(255, 100, 102, 0.8)'
))

fig.update_layout(
    title='Box Plot with Customized Whiskers',
    yaxis_title='Batsman Runs'
)
```

```
fig.show()
```



```
In [ ]:
```