Data: Import and export through csv, Excel, JSON, HTML, and SQL.

Pandas' several useful PARAMETERS

In [2]:
```
import pandas as pd

df = pd.read_csv("vk.csv")
df.sample()
```

Out[2]:

	match_id	batsman_runs
116	550	34

Opening a CSV file from an URL

In [3]:
```
import requests
from io import StringIO

url = "https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"

headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.
           Gecko/20100101 Firefox/66.0"}

req = requests.get(url, headers=headers)

data = StringIO(req.text)

pd.read_csv(data)
```

Out[3]:

	Country	Region
0	Algeria	AFRICA
1	Angola	AFRICA
2	Benin	AFRICA
3	Botswana	AFRICA
4	Burkina	AFRICA
...	...	...
189	Paraguay	SOUTH AMERICA
190	Peru	SOUTH AMERICA
191	Suriname	SOUTH AMERICA
192	Uruguay	SOUTH AMERICA
193	Venezuela	SOUTH AMERICA

194 rows × 2 columns

Sep Parameter

In [4]:
```
df =pd.read_csv("movie_titles_metadata.tsv")
df.head(3)
```

Out[4]:

	m0\t10 things i hate about you\t1999\t6.90\t62847\t['comedy' 'romance']
0	m1\t1492: conquest of paradise\t1992\t6.20\t10...
1	m2\t15 minutes\t2001\t6.10\t25854\t['action' '...
2	m3\t2001: a space odyssey\t1968\t8.40\t163227\...

it is in the "tsv" format which is Tab seperated Values

But we have used Here "read csv" . for this kind of these problem. Use **sep** Parameter

In [5]:
```
df = pd.read_csv("movie_titles_metadata.tsv",sep='\t')
df.head(3)
```

Out[5]:

	m0	10 things i hate about you	1999	6.90	62847	['comedy' 'romance']
0	m1	1492: conquest of paradise	1992	6.2	10421.0	['adventure' 'biography' 'drama' 'history']
1	m2	15 minutes	2001	6.1	25854.0	['action' 'crime' 'drama' 'thriller']
2	m3	2001: a space odyssey	1968	8.4	163227.0	['adventure' 'mystery' 'sci-fi']

Naming the Column

When data automatically takes first row as the column , we can use **names** paramter

In [6]:
```
df =pd.read_csv("movie_titles_metadata.tsv",sep='\t',
                names=['sno','name','release_year','rating','votes','genres'])
df.head(3)
```

Out[6]:

	sno	name	release_year	rating	votes	genres
0	m0	10 things i hate about you	1999	6.9	62847.0	['comedy' 'romance']
1	m1	1492: conquest of paradise	1992	6.2	10421.0	['adventure' 'biography' 'drama' 'history']
2	m2	15 minutes	2001	6.1	25854.0	['action' 'crime' 'drama' 'thriller']

Index_col parameter

In [7]:
```
df =pd.read_csv("aug_train.csv")
df.head(3)
```

Out[7]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course

replacing Default index column to desired column as index column using **index_col**

In [8]:
```
df =pd.read_csv("aug_train.csv",index_col='enrollee_id')
df.head(3)
```

Out[8]:

enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	e
8949	city_103	0.920	Male	Has relevent experience	no_enrollment	
29725	city_40	0.776	Male	No relevent experience	no_enrollment	
11561	city_21	0.624	NaN	No relevent experience	Full time course	

Header Parameter

In [9]:
```
df =pd.read_csv("test.csv")
df.head(3)
```

Out[9]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	
0	0	enrollee_id	city	city_development_index	gender	relevent_experience	enroll
1	1	29725	city_40	0.776	Male	No relevent experience	n
2	2	11561	city_21	0.624	NaN	No relevent experience	Ful

Here Problem is First Row Has index number as 0 so, Automatically it took first row as column names , for that we have to use **header** Parameter

In [10]:
```
df =pd.read_csv("test.csv", header=1)
df.head(3)
```

Out[10]:

	0	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_univers
0	1	29725	city_40	0.776	Male	No relevent experience	no_enrollm
1	2	11561	city_21	0.624	NaN	No relevent experience	Full time cou
2	3	33241	city_115	0.789	NaN	No relevent experience	N

use_cols parameter

In [11]:
```
df =pd.read_csv("aug_train.csv")
df.head(3)
```

Out[11]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course

usecols paramters helps us in chossig our desired columns from total data

In [12]:
```
df =pd.read_csv("aug_train.csv",usecols=['enrollee_id','gender','education_lev
df
```

Out[12]:

	enrollee_id	gender	education_level
0	8949	Male	Graduate
1	29725	Male	Graduate
2	11561	NaN	Graduate
3	33241	NaN	Graduate
4	666	Male	Masters
...	...	...	...
19153	7386	Male	Graduate
19154	31398	Male	Graduate
19155	24576	Male	Graduate
19156	5756	Male	High School
19157	23834	NaN	Primary School

19158 rows × 3 columns

Squeeze parameters

In [13]:
```
df =pd.read_csv("aug_train.csv")
df.head(3)
```

Out[13]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course

convert desired columns from dataframe into Series Object

In [14]:
```
df =pd.read_csv("aug_train.csv",usecols=['gender'],squeeze=True)
df.head()
```

Out[14]:
```
0    Male
1    Male
2     NaN
3     NaN
4    Male
Name: gender, dtype: object
```

Skiprows Parameter

In [15]:
```
df = pd.read_csv("aug_train.csv")
df.head()
```

Out[15]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course
3	33241	city_115	0.789	NaN	No relevent experience	NaN
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment

Here we can skip rows , as per our choice

In [16]:
```
df = pd.read_csv("aug_train.csv", skiprows=[0,1])
df.head()
```

Out[16]:

	29725	city_40	0.7759999999999999	Male	No relevent experience	no_enrollment	Graduate	STEM	'
0	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	
1	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<
2	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>2
3	21651	city_176	0.764	NaN	Has relevent experience	Part time course	Graduate	STEM	'
4	28806	city_160	0.920	Male	Has relevent experience	no_enrollment	High School	NaN	

nrows Paramter

In [17]:
```
df = pd.read_csv("aug_train.csv")
df
```

Out[17]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_unive
0	8949	city_103	0.920	Male	Has relevent experience	no_enrolli
1	29725	city_40	0.776	Male	No relevent experience	no_enrolli
2	11561	city_21	0.624	NaN	No relevent experience	Full time cc
3	33241	city_115	0.789	NaN	No relevent experience	
4	666	city_162	0.767	Male	Has relevent experience	no_enrolli
...	...	...	...	...	...	
19153	7386	city_173	0.878	Male	No relevent experience	no_enrolli
19154	31398	city_103	0.920	Male	Has relevent experience	no_enrolli
19155	24576	city_103	0.920	Male	Has relevent experience	no_enrolli
19156	5756	city_65	0.802	Male	Has relevent experience	no_enrolli
19157	23834	city_67	0.855	NaN	No relevent experience	no_enrolli

19158 rows × 14 columns

Here ,We Have Nearly 19158 rows . if we want only 100 rows , we can do that using **nrows**

In [18]:
```
df = pd.read_csv("aug_train.csv", nrows=100)
df
```

Out[18]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_universit
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollmer
1	29725	city_40	0.776	Male	No relevent experience	no_enrollmer
2	11561	city_21	0.624	NaN	No relevent experience	Full time cours
3	33241	city_115	0.789	NaN	No relevent experience	Nal
4	666	city_162	0.767	Male	Has relevent experience	no_enrollmer
...	...	...	...	...	...	.
95	12081	city_65	0.802	Male	Has relevent experience	Full time cours
96	7364	city_160	0.920	NaN	No relevent experience	Full time cours
97	11184	city_74	0.579	NaN	No relevent experience	Full time cours
98	7016	city_65	0.802	Male	Has relevent experience	no_enrollmer
99	8695	city_11	0.550	Male	Has relevent experience	no_enrollmer

100 rows × 14 columns

Encoding parameter

In [19]:
```
df = pd.read_csv("zomato.csv")
df.head()
```

Gives Error :UnicodeDecodeError: 'utf-8' codec can't decode byte 0xed in position 7044: invalid continuation byte

In [21]:
```
df = pd.read_csv("zomato.csv",encoding='latin-1')
df.head(3)
```

Skip bad lines

In [22]:
```
pd.read_csv('BX-Books.csv', sep=';', encoding="latin-1")
```

it gives parser error

In [23]:
```
pd.read_csv('BX-Books.csv', sep=';', encoding="latin-1",error_bad_lines=False)
```

dtypes parameter

In [24]:
```
df = pd.read_csv("aug_train.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19158 non-null  int64  
 1   city                    19158 non-null  object 
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object 
 4   relevent_experience     19158 non-null  object 
 5   enrolled_university     18772 non-null  object 
 6   education_level         18698 non-null  object 
 7   major_discipline        16345 non-null  object 
 8   experience              19093 non-null  object 
 9   company_size            13220 non-null  object 
 10  company_type            13018 non-null  object 
 11  last_new_job            18735 non-null  object 
 12  training_hours          19158 non-null  int64  
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

Here if we want to chnge the data type value in to someyhong , we can chage it.for example: here target column is float64, i want to chnge it to int32 , by using **dtypes** paramter

In [25]:
```
pd.read_csv('aug_train.csv',dtype={'target':int}).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19158 non-null  int64  
 1   city                    19158 non-null  object 
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object 
 4   relevent_experience     19158 non-null  object 
 5   enrolled_university     18772 non-null  object 
 6   education_level         18698 non-null  object 
 7   major_discipline        16345 non-null  object 
 8   experience              19093 non-null  object 
 9   company_size            13220 non-null  object 
 10  company_type            13018 non-null  object 
 11  last_new_job            18735 non-null  object 
 12  training_hours          19158 non-null  int64  
 13  target                  19158 non-null  int32  
dtypes: float64(1), int32(1), int64(2), object(10)
memory usage: 2.0+ MB
```

Handling Dates

In [26]:
```
pd.read_csv("IPL Matches 2008-2020.csv").info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               816 non-null    int64  
 1   city             803 non-null    object 
 2   date             816 non-null    object 
 3   player_of_match  812 non-null    object 
 4   venue            816 non-null    object 
 5   neutral_venue    816 non-null    int64  
 6   team1            816 non-null    object 
 7   team2            816 non-null    object 
 8   toss_winner      816 non-null    object 
 9   toss_decision    816 non-null    object 
 10  winner           812 non-null    object 
 11  result           812 non-null    object 
 12  result_margin    799 non-null    float64
 13  eliminator       812 non-null    object 
 14  method           19 non-null     object 
 15  umpire1          816 non-null    object 
 16  umpire2          816 non-null    object 
dtypes: float64(1), int64(2), object(14)
memory usage: 108.5+ KB
```

Here we have date column consisting of object='String' we have to convert date column into date and time using. Day and time paramter

In [27]:
```
pd.read_csv("IPL Matches 2008-2020.csv",parse_dates=['date']).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 816 entries, 0 to 815
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   id               816 non-null    int64         
 1   city             803 non-null    object        
 2   date             816 non-null    datetime64[ns]
 3   player_of_match  812 non-null    object        
 4   venue            816 non-null    object        
 5   neutral_venue    816 non-null    int64         
 6   team1            816 non-null    object        
 7   team2            816 non-null    object        
 8   toss_winner      816 non-null    object        
 9   toss_decision    816 non-null    object        
 10  winner           812 non-null    object        
 11  result           812 non-null    object        
 12  result_margin    799 non-null    float64       
 13  eliminator       812 non-null    object        
 14  method           19 non-null     object        
 15  umpire1          816 non-null    object        
 16  umpire2          816 non-null    object        
dtypes: datetime64[ns](1), float64(1), int64(2), object(13)
memory usage: 108.5+ KB
```

If we have multiple data consisting of day, Month and year. You have to combine the Three columns into one column and. We have to apply. Date and time. By using **parse data** paramter

- list of lists. e.g. If [[1, 3]] -> combine columns 1 and 3 and parse as a single date column
- dict, e.g. {‘foo’ : [1, 3]} -> parse columns 1, 3 as date and call result ‘foo’ If a column or index cannot be represented as an array of datetimes, say because of an unparsable value or a mixture of timezones, the column or index will be returned unaltered as an object data type. For non-standard datetime parsing, use pd.to_datetime after pd.read_csv.

Covertors

In [28]:
```
pd.read_csv("IPL Matches 2008-2020.csv").head(3)
```

Out[28]:

	id	city	date	player_of_match	venue	neutral_venue	team1	tear
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolka Knig Ride
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chenr Sup Kin
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils	Rajasth Roya

Here here we have another column, which is team1. It's consisting of multiple IPL Teams example Royal Challengers Bengalore. We have to rename that to RCB by using **converter** parameters.

In [29]:
```
def rename(name):
    if name == 'Royal Challengers Bangalore':
        return 'RCB'
    elif name == 'Delhi Daredevils':
            return 'DD'
    else:
        return name
```

In [30]:
```
rename("Royal Challengers Bangalore")
```

Out[30]:
```
'RCB'
```

In [31]:
```
rename('Delhi Daredevils')
```

Out[31]:
```
'DD'
```

In [32]:
```
pd.read_csv('IPL Matches 2008-2020.csv',converters={'team1':rename}).head(3)
```

Out[32]:

	id	city	date	player_of_match	venue	neutral_venue	team1	team2	t
0	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	RCB	Kolkata Knight Riders	
1	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	
2	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	DD	Rajasthan Royals	

na_values parameter

In [33]:
```
pd.read_csv('aug_train.csv').head()
```

Out[33]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course
3	33241	city_115	0.789	NaN	No relevent experience	NaN
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment

If the column consisting of null values, but it if it is represented as (-) in the data use **na_Values** Parameter.

- For example we have change 'Male' to Nan values , we can do it

In [34]:
```
pd.read_csv('aug_train.csv',na_values=['Male']).head(3)
```

Out[34]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university
0	8949	city_103	0.920	NaN	Has relevent experience	no_enrollment
1	29725	city_40	0.776	NaN	No relevent experience	no_enrollment
2	11561	city_21	0.624	NaN	No relevent experience	Full time course

Loading a huge datset in chunks

In [35]:
```
pd.read_csv('aug_train.csv').shape
```

Out[35]:
```
(19158, 14)
```

Here we have nearly 19158 rows and 14 columns, which is a large data set .for analysis purpose, We have to divide the data into smaller chunks for easy analysis by using **Chunk size**

In [36]:
```
dfs =pd.read_csv('aug_train.csv',chunksize=5000)
```

In [37]:
```
for chunksize in dfs:
    print(chunksize.shape)
```

```
(5000, 14)
(5000, 14)
(5000, 14)
(4158, 14)
```

In [38]:
```
dfs
```

Out[38]:
```
<pandas.io.parsers.readers.TextFileReader at 0x2296154be20>
```

Pandas Import

Working with Excel

In [39]:
```
pd.read_excel('bollywood.xlsx').head()
```

Out[39]:

	movie	lead
0	Uri: The Surgical Strike	Vicky Kaushal
1	Battalion 609	Vicky Ahuja
2	The Accidental Prime Minister (film)	Anupam Kher
3	Why Cheat India	Emraan Hashmi
4	Evening Shadows	Mona Ambegaonkar

In [ ]:
```
# For acessing by sheet_name

pd.read_excel('output.xlsx',sheet_name='sheet_name_2')
```

In [ ]:
```
# Reading Text files ( for reading text file, we have to use read_CSV  )

pd.read_csv('question_answer_pairs.txt',sep='\t')
```

Working with JSON

In [41]:
```
pd.read_json("train.json").head()
```

Out[41]:

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...

- JSON From URL :https://api.exchangerate-api.com/v4/latest/INR (https://api.exchangerate-api.com/v4/latest/INR)

In [42]:
```
pd.read_json('https://api.exchangerate-api.com/v4/latest/INR')
```

Out[42]:

	provider	WARNING_UPGRADE_TO_V6	terms	base	dat
AED	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
AFN	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
ALL	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
AMD	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
ANG	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
...	...	...	...	...	.
XPF	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
YER	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
ZAR	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
ZMW	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3
ZWL	https://www.exchangerate-api.com	https://www.exchangerate-api.com/docs/free	https://www.exchangerate-api.com/terms	INR	2023 05-3

162 rows × 7 columns

Working with SQL

- SQL file :[https://www.kaggle.com/datasets/busielmorley/worldcities-pop-lang-rank-sql-create-tbls?resource=download (https://www.kaggle.com/datasets/busielmorley/worldcities-pop-lang-rank-sql-create-tbls?resource=download)](https://www.kaggle.com/datasets/busielmorley/worldcities-pop-lang-rank-sql-create-tbls?resource=download)

In [ ]:
```
!pip install mysql.connector
```

In [43]:
```
import mysql.connector
```

In [48]:
```
db = mysql.connector.connect(host='localhost',user='root',
                             password='',database='world')
```

In [50]:
```
pd.read_sql_query('SELECT * FROM city' , db)
```

Out[50]:

	ID	Name	CountryCode	District	Population
0	1	Kabul	AFG	Kabol	1780000
1	2	Qandahar	AFG	Qandahar	237500
2	3	Herat	AFG	Herat	186800
3	4	Mazar-e-Sharif	AFG	Balkh	127800
4	5	Amsterdam	NLD	Noord-Holland	731200
...	...	...	...	...	...
4074	4075	Khan Yunis	PSE	Khan Yunis	123175
4075	4076	Hebron	PSE	Hebron	119401
4076	4077	Jabaliya	PSE	North Gaza	113901
4077	4078	Nablus	PSE	Nablus	100231
4078	4079	Rafah	PSE	Rafah	92020

4079 rows × 5 columns

In [52]:
```
pd.read_sql_query("SELECT * FROM countrylanguage",db)
```

Out[52]:

	CountryCode	Language	IsOfficial	Percentage
0	ABW	Dutch	T	5.3
1	ABW	English	F	9.5
2	ABW	Papiamento	F	76.7
3	ABW	Spanish	F	7.4
4	AFG	Balochi	F	0.9
...	...	...	...	...
979	ZMB	Tongan	F	11.0
980	ZWE	English	T	2.2
981	ZWE	Ndebele	F	16.2
982	ZWE	Nyanja	F	2.2
983	ZWE	Shona	F	72.1

984 rows × 4 columns

In [53]:
```
# Saving as DataFrame

df = pd.read_sql_query("SELECT * FROM countrylanguage",db)
```

In [54]:
```
df
```

Out[54]:

	CountryCode	Language	IsOfficial	Percentage
0	ABW	Dutch	T	5.3
1	ABW	English	F	9.5
2	ABW	Papiamento	F	76.7
3	ABW	Spanish	F	7.4
4	AFG	Balochi	F	0.9
...	...	...	...	...
979	ZMB	Tongan	F	11.0
980	ZWE	English	T	2.2
981	ZWE	Ndebele	F	16.2
982	ZWE	Nyanja	F	2.2
983	ZWE	Shona	F	72.1

984 rows × 4 columns

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   CountryCode  984 non-null    object 
 1   Language     984 non-null    object 
 2   IsOfficial   984 non-null    object 
 3   Percentage   984 non-null    float64
dtypes: float64(1), object(3)
memory usage: 30.9+ KB
```

Pandas Export

- to csv
- to excel
- to html
- to json
- to sql

to_csv

In [56]:
```
df = pd.read_csv("deliveries.csv")
```

In [58]:
```
df.head()
```

Out[58]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_s
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	

5 rows × 21 columns

In [60]:
```
# We want batsman and batsman runs

df.groupby('batsman')['batsman_runs'].sum().reset_index()
```

Out[60]:

	batsman	batsman_runs
0	A Ashish Reddy	280
1	A Chandila	4
2	A Chopra	53
3	A Choudhary	25
4	A Dananjaya	4
...	...	...
511	YV Takawale	192
512	Yashpal Singh	47
513	Younis Khan	3
514	Yuvraj Singh	2765
515	Z Khan	117

516 rows × 2 columns

In [61]:
```
# Here above content can be saved as csv

temp_df =df.groupby('batsman')['batsman_runs'].sum().reset_index()
```

In [63]:
```
temp_df.to_csv('batsman_runs',index=False) # For Saving to CSV
```

to_excel

In [64]:
```
temp_df
```

Out[64]:

	batsman	batsman_runs
0	A Ashish Reddy	280
1	A Chandila	4
2	A Chopra	53
3	A Choudhary	25
4	A Dananjaya	4
...	...	...
511	YV Takawale	192
512	Yashpal Singh	47
513	Younis Khan	3
514	Yuvraj Singh	2765
515	Z Khan	117

516 rows × 2 columns

In [67]:
```
temp_df2 = df.pivot_table(index='batsman',columns='bowling_team',
                          values ='batsman_runs',aggfunc='sum')
```

In [69]:
```
temp_df2.head(1) # Second Dataframe
```

Out[69]:

bowling_team	Chennai Super Kings	Deccan Chargers	Delhi Capitals	Delhi Daredevils	Gujarat Lions	Kings XI Punjab	Kochi Tuskers Kerala	Kolkata Knight Riders	Mum Indi
batsman									
A Ashish Reddy	45.0	NaN	NaN	36.0	NaN	37.0	NaN	17.0	2

In [65]:
```
temp_df.to_excel('batsman_runs.xlsx')
```

In [70]:
```
# for Acessing Multple Exel file use : ExcelWriter

with pd.ExcelWriter('output.xlsx') as writer:
    temp_df.to_excel(writer, sheet_name='Sheet_name_1')
    temp_df2.to_excel(writer, sheet_name='Sheet_name_1')
```

to_html

In [71]:
```
df
```

Out[71]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills
...	...	...	...	...	...	...	...	...	...
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga

179078 rows × 21 columns

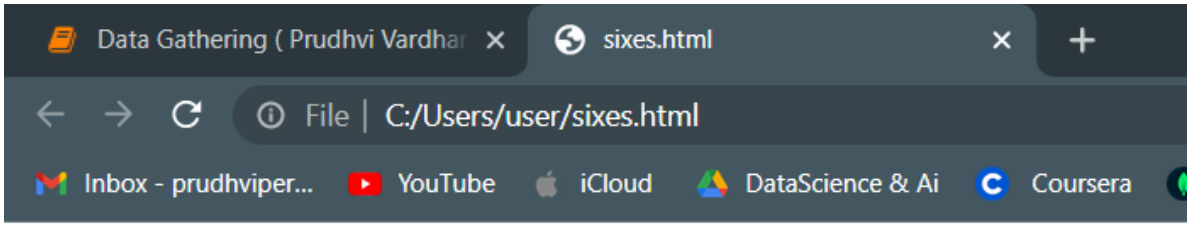
In [74]:
```
df.query('batsman_runs==6').pivot_table(index='over',
                                        columns='ball',
                                        values='batsman_runs',
                                        aggfunc='count')
```

Out[74]:

ball	1	2	3	4	5	6	7	8	9
over									
1	7.0	12.0	27.0	31.0	24.0	20.0	12.0	1.0	NaN
2	26.0	30.0	35.0	43.0	45.0	42.0	10.0	5.0	NaN
3	63.0	46.0	57.0	52.0	48.0	58.0	8.0	2.0	1.0
4	49.0	61.0	51.0	81.0	54.0	53.0	11.0	1.0	NaN
5	54.0	56.0	82.0	64.0	62.0	60.0	10.0	2.0	NaN
6	61.0	82.0	44.0	65.0	56.0	66.0	11.0	1.0	NaN
7	27.0	45.0	34.0	44.0	51.0	28.0	3.0	3.0	NaN
8	44.0	47.0	55.0	49.0	53.0	39.0	7.0	NaN	NaN
9	70.0	56.0	56.0	58.0	52.0	35.0	11.0	NaN	1.0
10	43.0	38.0	60.0	43.0	52.0	54.0	9.0	1.0	NaN
11	59.0	70.0	62.0	54.0	61.0	56.0	8.0	NaN	NaN
12	70.0	58.0	72.0	63.0	65.0	52.0	11.0	1.0	NaN
13	67.0	87.0	58.0	80.0	75.0	65.0	10.0	2.0	NaN
14	83.0	85.0	79.0	74.0	65.0	71.0	7.0	3.0	NaN
15	68.0	93.0	83.0	74.0	64.0	86.0	11.0	1.0	NaN
16	77.0	96.0	95.0	91.0	91.0	78.0	15.0	2.0	NaN
17	84.0	88.0	86.0	87.0	103.0	78.0	20.0	3.0	NaN
18	91.0	111.0	112.0	115.0	94.0	101.0	17.0	1.0	1.0
19	82.0	90.0	114.0	103.0	111.0	98.0	31.0	6.0	3.0
20	99.0	95.0	106.0	113.0	88.0	102.0	24.0	5.0	NaN

In [75]:
```
# Saving as HTML file

df.query('batsman_runs==6').pivot_table(index='over',
                                        columns='ball',
                                        values='batsman_runs',
                                        aggfunc='count').to_html('sixes.html')
```



ball	1	2	3	4	5	6	7	8	9
over									
1	7.0	12.0	27.0	31.0	24.0	20.0	12.0	1.0	NaN
2	26.0	30.0	35.0	43.0	45.0	42.0	10.0	5.0	NaN
3	63.0	46.0	57.0	52.0	48.0	58.0	8.0	2.0	1.0
4	49.0	61.0	51.0	81.0	54.0	53.0	11.0	1.0	NaN
5	54.0	56.0	82.0	64.0	62.0	60.0	10.0	2.0	NaN
6	61.0	82.0	44.0	65.0	56.0	66.0	11.0	1.0	NaN
7	27.0	45.0	34.0	44.0	51.0	28.0	3.0	3.0	NaN
8	44.0	47.0	55.0	49.0	53.0	39.0	7.0	NaN	NaN
9	70.0	56.0	56.0	58.0	52.0	35.0	11.0	NaN	1.0
10	43.0	38.0	60.0	43.0	52.0	54.0	9.0	1.0	NaN
11	59.0	70.0	62.0	54.0	61.0	56.0	8.0	NaN	NaN
12	70.0	58.0	72.0	63.0	65.0	52.0	11.0	1.0	NaN
13	67.0	87.0	58.0	80.0	75.0	65.0	10.0	2.0	NaN
14	83.0	85.0	79.0	74.0	65.0	71.0	7.0	3.0	NaN
15	68.0	93.0	83.0	74.0	64.0	86.0	11.0	1.0	NaN
16	77.0	96.0	95.0	91.0	91.0	78.0	15.0	2.0	NaN
17	84.0	88.0	86.0	87.0	103.0	78.0	20.0	3.0	NaN
18	91.0	111.0	112.0	115.0	94.0	101.0	17.0	1.0	1.0
19	82.0	90.0	114.0	103.0	111.0	98.0	31.0	6.0	3.0
20	99.0	95.0	106.0	113.0	88.0	102.0	24.0	5.0	NaN

to_json

In [76]:
```
df
```

Out[76]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills
...	...	...	...	...	...	...	...	...	...
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga

179078 rows × 21 columns

In [77]:
```
df.groupby(['batting_team','batsman'])['batsman_runs'].sum()
```

Out[77]:
```
batting_team          batsman
Chennai Super Kings   A Flintoff               62
                      A Mukund                  0
                      A Nehra                   1
                      AS Rajpoot                2
                      AT Rayudu               910
                                             ...
Sunrisers Hyderabad   WP Saha                 223
                      X Thalaivan Sargunam     10
                      Y Venugopal Rao          71
                      YK Pathan               319
                      Yuvraj Singh            488
Name: batsman_runs, Length: 935, dtype: int64
```

In [78]:
```
df.groupby(['batting_team','batsman'])['batsman_runs'].sum().unstack()
```

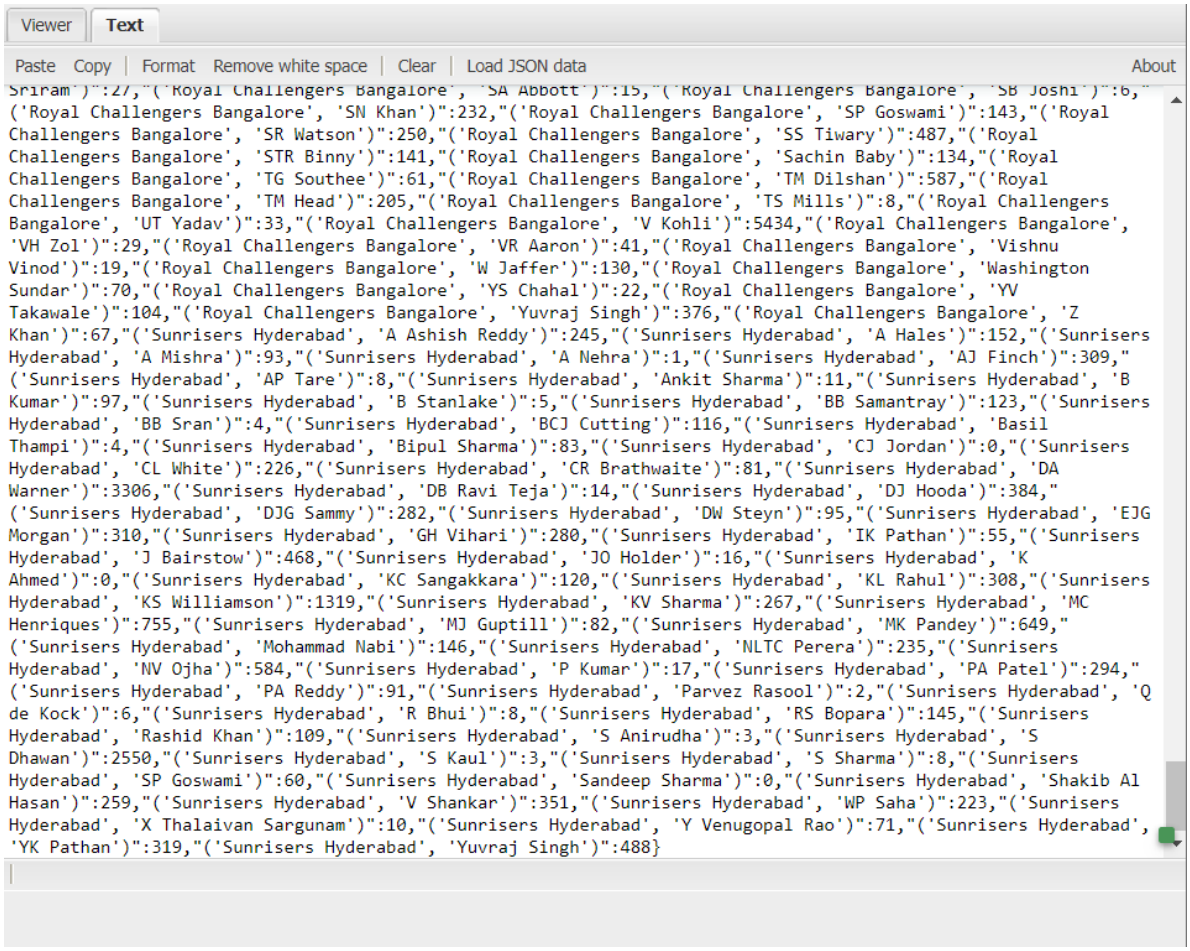
Out[78]:

batsman	A Ashish Reddy	A Chandila	A Chopra	A Choudhary	A Dananjaya	A Flintoff	A Hales	A Joseph	A Kumbl
batting_team									
Chennai Super Kings	NaN	NaN	NaN	NaN	NaN	62.0	NaN	NaN	NaN
Deccan Chargers	35.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Delhi Capitals	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Delhi Daredevils	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gujarat Lions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kings XI Punjab	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kochi Tuskers Kerala	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kolkata Knight Riders	NaN	NaN	53.0	NaN	NaN	NaN	NaN	NaN	NaN
Mumbai Indians	NaN	NaN	NaN	NaN	4.0	NaN	NaN	15.0	NaN
Pune Warriors	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rajasthan Royals	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rising Pune Supergiant	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rising Pune Supergiants	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Royal Challengers Bangalore	NaN	NaN	NaN	25.0	NaN	NaN	NaN	NaN	35.(
Sunrisers Hyderabad	245.0	NaN	NaN	NaN	NaN	NaN	152.0	NaN	NaN

15 rows × 516 columns

In [80]:
```
## Saving TO JSON

df.groupby(['batting_team','batsman'])['batsman_runs'].sum().unstack().to_jso
```




```
Sriram')":27,"('Royal Challengers Bangalore', 'SA Abbott')":15,"('Royal Challengers Bangalore', 'SB Joshi')":6,"
('Royal Challengers Bangalore', 'SN Khan')":232,"('Royal Challengers Bangalore', 'SP Goswami')":143,"('Royal
Challengers Bangalore', 'SR Watson')":250,"('Royal Challengers Bangalore', 'SS Tiwary')":487,"('Royal
Challengers Bangalore', 'STR Binny')":141,"('Royal Challengers Bangalore', 'Sachin Baby')":134,"('Royal
Challengers Bangalore', 'TG Southee')":61,"('Royal Challengers Bangalore', 'TM Dilshan')":587,"('Royal
Challengers Bangalore', 'TM Head')":205,"('Royal Challengers Bangalore', 'TS Mills')":8,"('Royal Challengers
Bangalore', 'UT Yadav')":33,"('Royal Challengers Bangalore', 'V Kohli')":5434,"('Royal Challengers Bangalore',
'VH Zol')":29,"('Royal Challengers Bangalore', 'VR Aaron')":41,"('Royal Challengers Bangalore', 'Vishnu
Vinod')":19,"('Royal Challengers Bangalore', 'W Jaffer')":130,"('Royal Challengers Bangalore', 'Washington
Sundar')":70,"('Royal Challengers Bangalore', 'YS Chahal')":22,"('Royal Challengers Bangalore', 'YV
Takawale')":104,"('Royal Challengers Bangalore', 'Yuvraj Singh')":376,"('Royal Challengers Bangalore', 'Z
Khan')":67,"('Sunrisers Hyderabad', 'A Ashish Reddy')":245,"('Sunrisers Hyderabad', 'A Hales')":152,"('Sunrisers
Hyderabad', 'A Mishra')":93,"('Sunrisers Hyderabad', 'A Nehra')":1,"('Sunrisers Hyderabad', 'AJ Finch')":309,"
('Sunrisers Hyderabad', 'AP Tare')":8,"('Sunrisers Hyderabad', 'Ankit Sharma')":11,"('Sunrisers Hyderabad', 'B
Kumar')":97,"('Sunrisers Hyderabad', 'B Stanlake')":5,"('Sunrisers Hyderabad', 'BB Samantray')":123,"('Sunrisers
Hyderabad', 'BB Sran')":4,"('Sunrisers Hyderabad', 'BCJ Cutting')":116,"('Sunrisers Hyderabad', 'Basil
Thampi')":4,"('Sunrisers Hyderabad', 'Bipul Sharma')":83,"('Sunrisers Hyderabad', 'CJ Jordan')":0,"('Sunrisers
Hyderabad', 'CL White')":226,"('Sunrisers Hyderabad', 'CR Brathwaite')":81,"('Sunrisers Hyderabad', 'DA
Warner')":3306,"('Sunrisers Hyderabad', 'DB Ravi Teja')":14,"('Sunrisers Hyderabad', 'DJ Hooda')":384,"
('Sunrisers Hyderabad', 'DJG Sammy')":282,"('Sunrisers Hyderabad', 'DW Steyn')":95,"('Sunrisers Hyderabad', 'EJG
Morgan')":310,"('Sunrisers Hyderabad', 'GH Vihari')":280,"('Sunrisers Hyderabad', 'IK Pathan')":55,"('Sunrisers
Hyderabad', 'J Bairstow')":468,"('Sunrisers Hyderabad', 'JO Holder')":16,"('Sunrisers Hyderabad', 'K
Ahmed')":0,"('Sunrisers Hyderabad', 'KC Sangakkara')":120,"('Sunrisers Hyderabad', 'KL Rahul')":308,"('Sunrisers
Hyderabad', 'KS Williamson')":1319,"('Sunrisers Hyderabad', 'KV Sharma')":267,"('Sunrisers Hyderabad', 'MC
Henriques')":755,"('Sunrisers Hyderabad', 'MJ Guptill')":82,"('Sunrisers Hyderabad', 'MK Pandey')":649,"
('Sunrisers Hyderabad', 'Mohammad Nabi')":146,"('Sunrisers Hyderabad', 'NLTC Perera')":235,"('Sunrisers
Hyderabad', 'NV Ojha')":584,"('Sunrisers Hyderabad', 'P Kumar')":17,"('Sunrisers Hyderabad', 'PA Patel')":294,"
('Sunrisers Hyderabad', 'PA Reddy')":91,"('Sunrisers Hyderabad', 'Parvez Rasool')":2,"('Sunrisers Hyderabad', 'Q
de Kock')":6,"('Sunrisers Hyderabad', 'R Bhui')":8,"('Sunrisers Hyderabad', 'RS Bopara')":145,"('Sunrisers
Hyderabad', 'Rashid Khan')":109,"('Sunrisers Hyderabad', 'S Anirudha')":3,"('Sunrisers Hyderabad', 'S
Dhawan')":2550,"('Sunrisers Hyderabad', 'S Kaul')":3,"('Sunrisers Hyderabad', 'S Sharma')":8,"('Sunrisers
Hyderabad', 'SP Goswami')":60,"('Sunrisers Hyderabad', 'Sandeep Sharma')":0,"('Sunrisers Hyderabad', 'Shakib Al
Hasan')":259,"('Sunrisers Hyderabad', 'V Shankar')":351,"('Sunrisers Hyderabad', 'WP Saha')":223,"('Sunrisers
Hyderabad', 'X Thalaivan Sargunam')":10,"('Sunrisers Hyderabad', 'Y Venugopal Rao')":71,"('Sunrisers Hyderabad',
'YK Pathan')":319,"('Sunrisers Hyderabad', 'Yuvraj Singh')":488}
```

to_sql

In [81]:
```
df
```

Out[81]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills
...	...	...	...	...	...	...	...	...	...
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga

179078 rows × 21 columns

In [84]:
```
!pip install pymysql
```

```
Collecting pymysql
  Downloading PyMySQL-1.0.3-py3-none-any.whl (43 kB)
Installing collected packages: pymysql
Successfully installed pymysql-1.0.3
```

In [85]:
```
import pymysql

from sqlalchemy import create_engine
```

In [87]:
```
engine = create_engine("mysql+pymysql://root:@localhost/ipl")

# (root): (password)@(url)/(database) df.to_sql('ipl_delivery', con engine, if

df.to_sql('ipl_delivery', con=engine, if_exists='append')
```

In [88]:
```
# Second Table

temp_df
```

Out[88]:

	batsman	batsman_runs
0	A Ashish Reddy	280
1	A Chandila	4
2	A Chopra	53
3	A Choudhary	25
4	A Dananjaya	4
...	...	...
511	YV Takawale	192
512	Yashpal Singh	47
513	Younis Khan	3
514	Yuvraj Singh	2765
515	Z Khan	117

516 rows × 2 columns

In [89]:
```
#Export to SQL server

df.to_sql('batsman_runs', con=engine, if_exists='append')
```

In [91]:
```
sixes =df.groupby(['batting_team','batsman'])['batsman_runs'].sum().unstack()
```

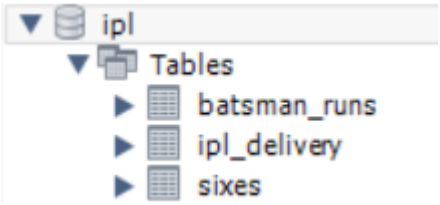
In [92]:
```
sixes
```

Out[92]:

batsman	A Ashish Reddy	A Chandila	A Chopra	A Choudhary	A Dananjaya	A Flintoff	A Hales	A Joseph	A Kumbl
batting_team									
Chennai Super Kings	NaN	NaN	NaN	NaN	NaN	62.0	NaN	NaN	NaN
Deccan Chargers	35.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Delhi Capitals	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Delhi Daredevils	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gujarat Lions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kings XI Punjab	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kochi Tuskers Kerala	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Kolkata Knight Riders	NaN	NaN	53.0	NaN	NaN	NaN	NaN	NaN	NaN
Mumbai Indians	NaN	NaN	NaN	NaN	4.0	NaN	NaN	15.0	NaN
Pune Warriors	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rajasthan Royals	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rising Pune Supergiant	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Rising Pune Supergiants	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Royal Challengers Bangalore	NaN	NaN	NaN	25.0	NaN	NaN	NaN	NaN	35.(
Sunrisers Hyderabad	245.0	NaN	NaN	NaN	NaN	NaN	152.0	NaN	NaN

15 rows × 516 columns

In [93]:
```
df.to_sql('sixes', con=engine, if_exists='append')
```



In [ ]: