


```
from google.colab import files
uploaded = files.upload()
```

 Choose Files 2 files

- **deliveries.csv**(text/csv) - 484206 bytes, last modified: 4/6/2025 - 100% done
- **matches.csv**(text/csv) - 2753 bytes, last modified: 4/6/2025 - 100% done

Saving deliveries.csv to deliveries.csv
Saving matches.csv to matches.csv

```
import pandas as pd

matches = pd.read_csv('matches.csv')
deliveries = pd.read_csv('deliveries.csv')

# Show first few rows
matches.head()
```

	match_id	date	venue	team1	team2	stage	toss_winner	toss_decision	first_ings_score	first_ings_wkts	second_ings_sco
0	1	March 22,2025	Eden Gardens, Kolkata	KKR	RCB	League	RCB	Bowl	174	8	-
1	2	March 23,2025	Rajiv Gandhi International Stadium, Hyderabad	SRH	RR	League	RR	Bowl	286	6	-
2	3	March 23,2025	MA Chidambaram Stadium, Chennai	CSK	MI	League	CSK	Bowl	155	9	-
3	4	March 24,2025	ACA-VDCA Cricket Stadium, Vishakhapatnam	DC	LSG	League	DC	Bowl	209	8	-
4	5	March 25,2025	Narendra Modi Stadium, Ahmedabad	GT	PBKS	League	GT	Bowl	243	5	-

Next steps:

[Generate code with matches](#)

[View recommended plots](#)

[New interactive sheet](#)

```
import numpy as np
import pandas as pd
deliveries = pd.read_csv('/content/deliveries.csv')
matches = pd.read_csv('/content/matches.csv')

# Display the first few rows
deliveries.head()
```

	i_no	date	stage	venue	batting_team	bowling_team	innings	over	striker	bowler	runs_of_bat	extras	wide	legbyes	byes
	1	Mar 22, 2025	League stage	Eden Gardens, Kolkata	KKR	RCB	1	0.1	de Kock	Hazlewood	0	0	0	0	
	1	Mar 22, 2025	League stage	Eden Gardens, Kolkata	KKR	RCB	1	0.2	de Kock	Hazlewood	4	0	0	0	
	1	Mar 22, 2025	League stage	Eden Gardens, Kolkata	KKR	RCB	1	0.3	de Kock	Hazlewood	0	0	0	0	
	1	Mar 22, 2025	League stage	Eden Gardens, Kolkata	KKR	RCB	1	0.4	de Kock	Hazlewood	0	0	0	0	
	1	Mar 22, 2025	League stage	Eden Gardens, Kolkata	KKR	RCB	1	0.5	de Kock	Hazlewood	0	0	0	0	

Next steps:

[Generate code with deliveries](#)[View recommended plots](#)[New interactive sheet](#)

```
# Creating a NumPy array of fixed type (integers)
arr = np.array([1, 2, 3, 4], dtype=np.int32)
print(arr)
print(arr.dtype)
```

```
[1 2 3 4]
int32
```

```
a = np.array([1, 2, 3])
b = np.zeros((2, 3))
c = np.ones((2, 2))
d = np.arange(10)
e = np.linspace(0, 1, 5)
```

```
print("Array a:", a)
print("Zeros:\n", b)
print("Ones:\n", c)
print("Arange:", d)
print("Linspace:", e)
```

```
Array a: [1 2 3]
Zeros:
[[0. 0. 0.]
 [0. 0. 0.]]
Ones:
[[1. 1.]
 [1. 1.]]
Arange: [0 1 2 3 4 5 6 7 8 9]
Linspace: [0. 0.25 0.5 0.75 1. ]
```

```
arr = np.arange(10)
print("Original:", arr)
print("First 5 elements:", arr[:5])
print("Every other element:", arr[::2])
```

```
Original: [0 1 2 3 4 5 6 7 8 9]
First 5 elements: [0 1 2 3 4]
Every other element: [0 2 4 6 8]
```

```
reshaped = np.arange(12).reshape((3, 4))
print("Reshaped Array:\n", reshaped)
```

```
Reshaped Array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
concat = np.concatenate([a, b])
split = np.split(concat, 2)
```

```
print("Concatenated:", concat)
print("Split:", split)
```

```
Concatenated: [1 2 3 4 5 6]
Split: [array([1, 2, 3]), array([4, 5, 6])]
```

```
arr = np.array([1, 2, 3, 4])
print("Square Root:", np.sqrt(arr))
print("Exponential:", np.exp(arr))
```

```
Square Root: [1.          1.41421356 1.73205081 2.          ]
Exponential: [ 2.71828183  7.3890561  20.08553692 54.59815003]
```

```
arr = np.random.randint(1, 100, size=(4, 5))
print("Array:\n", arr)
print("Sum:", arr.sum())
print("Max:", arr.max())
print("Mean (column-wise):", arr.mean(axis=0))
```

```
→ Array:
[[19 78 99 60 56]
 [23 94 20 58 37]
 [64 56 68 57 90]
 [63 10 23 24 26]]
Sum: 1025
Max: 99
Mean (column-wise): [42.25 59.5  52.5  49.75 52.25]
```

```
a = np.array([1, 2, 3])
b = 2
print("Broadcasting addition:", a + b)
```

```
→ Broadcasting addition: [3 4 5]
```

```
arr = np.array([10, 15, 20, 25])
mask = arr > 15
print("Boolean Mask:", mask)
print("Filtered:", arr[mask])
```

```
→ Boolean Mask: [False False  True  True]
Filtered: [20 25]
```

```
indices = [1, 3]
print("Fancy Indexing:", arr[indices])
```

```
→ Fancy Indexing: [15 25]
```

```
arr = np.array([42, 1, 15, 3])
print("Sorted:", np.sort(arr))
print("Indices for sorting:", np.argsort(arr))
```

```
→ Sorted: [ 1  3 15 42]
Indices for sorting: [1 3 2 0]
```

```
arr = np.array([10, 3, 15, 7, 8])
top3 = np.partition(arr, -3)[-3:]
print("Top 3 values (Unsorted):", top3)
```

```
→ Top 3 values (Unsorted): [ 8 10 15]
```

```
data = np.array([
    (1, 'Kohli', 50.0),
    (2, 'Dhoni', 45.2),
], dtype=[('id', 'i4'), ('name', 'U10'), ('avg', 'f4')])

print("Structured Array:\n", data)
print("Names only:", data['name'])
```

```
→ Structured Array:
[(1, 'Kohli', 50. ) (2, 'Dhoni', 45.2)]
Names only: ['Kohli' 'Dhoni']
```

```
record = data.view(np.recarray)
print(record.name)
```

```
→ ['Kohli' 'Dhoni']
```

```
runs_by_batsman = deliveries.groupby('striker')['runs_of_bat'].sum().sort_values(ascending=False)
print(runs_by_batsman.head(10))
```

```
↔ striker
Pooran                201
Sai Sudharsan         186
Mitchell Marsh        184
Suryakumar Yadav      171
Buttler               166
Shreyas Iyer          149
Head                  140
Angkrish Raghuvanshi  128
Klaasen               125
Aniket Verma          123
Name: runs_of_bat, dtype: int64
```