

```

1 #mount google drive
2
3 from google.colab import drive
4 drive.mount('/content/drive')
5

```

↗ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

Use the following code to upload the file from your local machine:

```

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

```

After uploading, read the file into a Pandas DataFrame:

```

import pandas as pd

dataframe = pd.read_csv('student.csv')

dataframe``

1 #import pandas
2 import pandas as pd
3

1 import pandas as pd
2
3 # Replace with the actual path to your file
4 file_path = '/content/drive/MyDrive/JustIT_Python/pandas/Resources/Copy of student.csv'
5 df = pd.read_csv(file_path)
6 print(df)
7

```

↗

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
6	7	My John Rob	Fifth	78	male
7	8	Asruid	Five	85	male
8	9	Tes Qry	Six	78	NaN
9	10	Big John	Four	55	female
10	11	Ronald	Six	89	female
11	12	Recky	Six	94	female
12	13	Kty	Seven	88	female
13	14	Bigy	Seven	88	female
14	15	Tade Row	NaN	88	male
15	16	Gimmy	Four	88	male
16	17	Tumyu	Six	54	male
17	18	Honny	Five	75	male
18	19	Tinny	Nine	18	male
19	20	Jackly	Nine	65	female
20	21	Babby John	Four	69	female
21	22	Reggid	Seven	55	female
22	23	Herod	Eight	79	male
23	24	Tiddy Now	Seven	78	male
24	25	Giff Tow	Seven	88	male
25	26	Crelea	Seven	79	male
26	27	NaN	Three	81	NaN
27	28	Rojj Base	Seven	86	female
28	29	Tess Played	Seven	55	male
29	30	Reppy Red	Six	79	female
30	31	Marry Toeey	Four	88	male
31	32	Binn Rott	Seven	90	female
32	33	Kenn Rein	Six	96	female
33	34	Gain Toe	Seven	69	male
34	35	Rows Noump	Six	88	female

✓ Inspecting Data

Lets try and see what methods are available to us to inspect datasets

```
1 #head() by default shows the top five rows of the dataset
2 print(df.head())
```

```
↵ id      name  class  mark  gender
0   1   John Deo   Four   75   female
1   2   Max Ruin  Three   85   male
2   3   Arnold   Three   55   male
3   4   Krish Star Four   60   female
4   5   John Mike  Four   60   female
```

```
1 df.head(5)
```

```
↵ id      name  class  mark  gender
0   1   John Deo   Four   75   female
1   2   Max Ruin  Three   85   male
2   3   Arnold   Three   55   male
3   4   Krish Star Four   60   female
4   5   John Mike  Four   60   female
```

```
1 #you can view amount of rows if you insert number into brackets
2 df.head(10)
```

```
↵ id      name  class  mark  gender
0   1   John Deo   Four   75   female
1   2   Max Ruin  Three   85   male
2   3   Arnold   Three   55   male
3   4   Krish Star Four   60   female
4   5   John Mike  Four   60   female
5   6   Alex John  Four   55   male
6   7   My John Rob Fifth   78   male
7   8   Asruid     Five   85   male
8   9   Tes Qry    Six    78   NaN
9  10   Big John   Four   55   female
```

```
1 #if there is a head there is a tail
2 #by default it shows the last five rows
3 df.tail()
```

```
↵ id      name  class  mark  gender
30  31  Marry Toeey   Four   88   male
31  32   Binn Rott   Seven   90   female
32  33   Kenn Rein    Six   96   female
33  34   Gain Toe    Seven   69   male
34  35  Rows Noump    Six   88   female
```

```
1 #checking for basic information using info()
2 df.info()
```

```
↵ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
#   Column  Non-Null Count  Dtype
---  -
0    id     35 non-null     int64
1   name    34 non-null     object
2   class   34 non-null     object
3   mark    35 non-null     int64
4   gender  33 non-null     object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
```

```
1 # inspecting summary statistics
2 print(df.describe())
3
4 df.describe()
```



	id	mark
count	35.000000	35.000000
mean	18.000000	74.657143
std	10.246951	16.401117
min	1.000000	18.000000
25%	9.500000	62.500000
50%	18.000000	79.000000
75%	26.500000	88.000000
max	35.000000	96.000000

	id	mark
count	35.000000	35.000000
mean	18.000000	74.657143
std	10.246951	16.401117
min	1.000000	18.000000
25%	9.500000	62.500000
50%	18.000000	79.000000
75%	26.500000	88.000000
max	35.000000	96.000000

```
1 #checking for missing values one by one
2 df.isnull()
```



	id	name	class	mark	gender
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	True
9	False	False	False	False	False
10	False	False	False	False	False
11	False	False	False	False	False
12	False	False	False	False	False
13	False	False	False	False	False
14	False	False	True	False	False
15	False	False	False	False	False
16	False	False	False	False	False
17	False	False	False	False	False
18	False	False	False	False	False
19	False	False	False	False	False
20	False	False	False	False	False
21	False	False	False	False	False
22	False	False	False	False	False
23	False	False	False	False	False
24	False	False	False	False	False
25	False	False	False	False	False
26	False	True	False	False	True
27	False	False	False	False	False
28	False	False	False	False	False
29	False	False	False	False	False
30	False	False	False	False	False

```
1 #checking for missing values
2 df.isnull().sum()
```

```
34 False 0 False False False False
id      0
name    1
class    1
mark     0
gender   2
dtype: int64
```

```
1 #checking for summary of the missing values
2 df.isnull().sum().sum()
```

```
4
```

```
1 #inspecting the rows v column
2 df.shape
```

```
(35, 5)
```

```
1 #checking for unique values in a column
2 df['gender'].unique()
```

```
array(['female', 'male', nan], dtype=object)
```

```
1 #finding duplicates in the data
2 df.duplicated().sum() #this checks for duplicated rows
```

```
0
```

```
1 #sorting for the top marks achieved by students
2 df.sort_values(by='mark', ascending=False) #by default its ascending (which is when its not included)
3 #descending does not exist so it is written ascending=False
```



	id	name	class	mark	gender
32	33	Kenn Rein	Six	96	female
11	12	Recky	Six	94	female
31	32	Binn Rott	Seven	90	female
10	11	Ronald	Six	89	female
24	25	Giff Tow	Seven	88	male
15	16	Gimmy	Four	88	male
14	15	Tade Row	NaN	88	male
13	14	Bigy	Seven	88	female
12	13	Kty	Seven	88	female
34	35	Rows Noup	Six	88	female
30	31	Marry Toeey	Four	88	male
27	28	Rojj Base	Seven	86	female
7	8	Asruid	Five	85	male
1	2	Max Ruin	Three	85	male
26	27	NaN	Three	81	NaN
22	23	Herod	Eight	79	male
29	30	Reppy Red	Six	79	female
25	26	Crelea	Seven	79	male
8	9	Tes Qry	Six	78	NaN
6	7	My John Rob	Fifth	78	male
23	24	Tiddy Now	Seven	78	male
0	1	John Deo	Four	75	female
17	18	Honny	Five	75	male
20	21	Babby John	Four	69	female
33	34	Gain Toe	Seven	69	male
19	20	Jackly	Nine	65	female
4	5	John Mike	Four	60	female
3	4	Krish Star	Four	60	female
21	22	Reggid	Seven	55	female
9	10	Big John	Four	55	female
28	29	Tess Played	Seven	55	male

```
1 #sorting for the top 10 students by mark
2 df.sort_values(by='mark', ascending=False).head(10)
```



	id	name	class	mark	gender
32	33	Kenn Rein	Six	96	female
11	12	Recky	Six	94	female
31	32	Binn Rott	Seven	90	female
10	11	Ronald	Six	89	female
24	25	Giff Tow	Seven	88	male
15	16	Gimmy	Four	88	male
14	15	Tade Row	NaN	88	male
13	14	Bigy	Seven	88	female
12	13	Kty	Seven	88	female
34	35	Rows Noup	Six	88	female

```
1 #sorting by multiple columns with different ordering
2 df.sort_values(by=['mark', 'id'], ascending=[False, True])
```



Show hidden output

```
1 #selecting columns
2 df['name']
```



		name	mark
0		John Deo	75
1		Max Ruin	85
2		Arnold	55
3		Krish Star	60
4		John Mike	60

```
1 df.head()
```



	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

Locating data .iloc() and .loc()

.loc is label based way of retrieving data from the dataframe

.iloc is interger based way of retrieving data from the dataframe

✓ Locating data .iloc[] and .loc[]

Lesson Plan: Understanding .loc and .iloc in Pandas

Course: Data Technician – Python for Data Analysis

Day: 3

Duration: 1 Hour

Teaching Style: Hands-on with explanations and real dataset usage

Lesson Objectives

By the end of this lesson, learners will be able to:

1. Understand the difference between .loc and .iloc .
2. Retrieve specific rows and columns using .loc and .iloc .
3. Apply filtering techniques to extract meaningful data.
4. Practice .loc and .iloc using the **Student Dataset**.

1. Introduction: What are .loc and .iloc?

When working with large datasets in Pandas, we often need to **select specific rows and columns**.

This can be done using:

1. .loc[] → Selects data **by label (name of the row or column)**.
2. .iloc[] → Selects data **by position (row or column index number)**.

Think of it as:

- .loc → Uses **explicit labels** (like column names or row labels).
- .iloc → Uses **integer-based indexing** (position numbers).

2. Key Differences Between .loc and .iloc

Feature	.loc[]	.iloc[]
Selection Type	Label-based (row/column name)	Position-based (row/column index)
Index Type	Works with row names/column names	Works with integer index
Inclusive Range?	Yes, includes both start and end index	No, excludes the last index
Example (Row 1 to 3)	<code>df.loc[1:3]</code> → Includes row 3	<code>df.iloc[1:3]</code> → Excludes row 3
Selecting Single Row	<code>df.loc[2]</code> → Selects row 2	<code>df.iloc[2]</code> → Selects row 2
Selecting Single Column	<code>df.loc[:, 'mark']</code>	<code>df.iloc[:, 2]</code> (if 'mark' is column 2)
Filtering with Conditions?	Yes	No

```
1 row_data =df.loc[2]
2 print(row_data)
```

```
↩ id      3
  name    Arnold
  class    Three
  mark     55
  gender   male
  Name: 2, dtype: object
```

```
1 subset_data =df.iloc[0:2, 0:2]
2 print(subset_data)
3
```

```
↩ id      name
0  1  John Deo
1  2  Max Ruin
```

```
1 #using .loc i want to access row 10
2 df.loc[10]
```

```
↩      10
id      11
name    Ronald
class    Six
mark     89
gender   female
```

dtype: object

```
1 #selecting multiple rows
2 df.loc[[0,1,2,3]]
```

```
↩ id      name  class  mark  gender
0  1  John Deo   Four   75  female
1  2  Max Ruin   Three   85   male
2  3   Arnold   Three   55   male
3  4  Krish Star   Four   60  female
```

```
1 #selecting multiple rows from a range
2 df.loc[3:8] #in label based, the row 8 will be included
```

```
↩ id      name  class  mark  gender
3  4  Krish Star   Four   60  female
4  5   John Mike   Four   60  female
5  6   Alex John   Four   55   male
6  7  My John Rob   Fifth  78   male
7  8     Asruid   Five   85   male
8  9    Tes Qry    Six   78   NaN
```

```
1 #accessing a specific column
2 df.loc[:, "gender"]
```



	gender
0	female
1	male
2	male
3	female
4	female
5	male
6	male
7	male
8	NaN
9	female
10	female
11	female
12	female
13	female
14	male
15	male
16	male
17	male
18	male
19	female
20	female
21	female
22	male
23	male
24	male
25	male
26	NaN
27	female
28	male
29	female
30	male

```
1 #accessing multiple columns
2 df.loc[:, ['name', 'mark']]
```



38 row hidden output

```
1 #accessing rows and columns
2 df.loc[[1,2,3], ['gender', 'mark']]
```




	gender	mark
1	male	85
2	male	55
3	female	60

```
1 #accessing rows and columns using range for rows
2 df.loc[0:3, ['gender', 'mark']]
```




	gender	mark
0	female	75
1	male	85
2	male	55
3	female	60


```
1 #filtering data with conditions
2 df.loc[df['mark'] > 80].head()
```




	id	name	class	mark	gender
1	2	Max Ruin	Three	85	male
7	8	Asruid	Five	85	male
10	11	Ronald	Six	89	female
11	12	Recky	Six	94	female
12	13	Kty	Seven	88	female

```
1 #filtering data with multiple conditions
2 #showing marks from class seven for females only
3 df.loc[(df['gender'] == 'female') & (df['class'] == 'Seven')]
```



	id	name	class	mark	gender
12	13	Kty	Seven	88	female
13	14	Bigy	Seven	88	female
21	22	Reggid	Seven	55	female
27	28	Rojj Base	Seven	86	female
31	32	Binn Rott	Seven	90	female

```
1 df.loc[(df["class"] == "Four")]
```



	id	name	class	mark	gender
0	1	John Deo	Four	75	female
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
9	10	Big John	Four	55	female
15	16	Gimmy	Four	88	male
20	21	Babby John	Four	69	female
30	31	Marry Toeey	Four	88	male

```
1 #select specific columns
2 df[['name', 'mark']]
```



	name	mark
0	John Deo	75
1	Max Ruin	85
2	Arnold	55
3	Krish Star	60
4	John Mike	60
5	Alex John	55
6	My John Rob	78
7	Asruid	85
8	Tes Qry	78
9	Big John	55
10	Ronald	89
11	Recky	94
12	Kty	88
13	Bigy	88
14	Tade Row	88
15	Gimmy	88
16	Tumyu	54
17	Honny	75
18	Tinny	18
19	Jackly	65
20	Babby John	69
21	Reggid	55
22	Herod	79
23	Tiddy Now	78
24	Giff Tow	88
25	Crelea	79
26	NaN	81
27	Rojj Base	86
28	Tess Played	55
29	Reppy Red	79
30	Marry Toeey	88

```

1 # add a new column based on condition
2 df["passed"] = df["mark"] >= 60
3 df.head()

```



	id	name	class	mark	gender	passed
0	1	John Deo	FOUR	75	female	True
1	2	Max Ruin	THREE	85	male	True
2	3	Arnold	THREE	55	male	False
3	4	Krish Star	FOUR	60	female	True
4	5	John Mike	FOUR	60	female	True

```

1 #add a new column with filled info
2 df["new"] = "stuff"
3 df.head()

```



	id	name	class	score	gender	new	new2
0	1	John Deo	FOUR	75	female	stuff	stuff
1	2	Max Ruin	THREE	85	male	stuff	stuff
2	3	Arnold	THREE	55	male	stuff	stuff
3	4	Krish Star	FOUR	60	female	stuff	stuff
4	5	John Mike	FOUR	60	female	stuff	stuff

```
1 # rename column
2 df = df.rename(columns={"mark" : "score"})
3 df.head()
```



	id	name	class	score	gender	passed
0	1	John Deo	FOUR	75	female	True
1	2	Max Ruin	THREE	85	male	True
2	3	Arnold	THREE	55	male	False
3	4	Krish Star	FOUR	60	female	True
4	5	John Mike	FOUR	60	female	True

```
1 # IGNORE to revert column back to original
2 df = df.rename(columns={"score" : "mark"})
3 df.head()
```



Show hidden output

```
1 # dropping column - removing a column
2 df = df.drop(columns=["passed"])
3 df.head()
4 #once it is run it can't be repeated since the column has been dropped and does not exist to run again
```



Show hidden output

```
1 # dropping columns - removing multiple columns
2 df = df.drop(columns=["new", "new2"])
3 df.head()
4 #once it is run it can't be repeated since the columns have been dropped and does not exist to run again
```



	id	name	class	score	gender
0	1	John Deo	FOUR	75	female
1	2	Max Ruin	THREE	85	male
2	3	Arnold	THREE	55	male
3	4	Krish Star	FOUR	60	female
4	5	John Mike	FOUR	60	female

```
1 #group df by "class" and calculating the mean "mark" for each group
2 df.groupby("class")["mark"].mean()
```



	mark
class	
Eight	79.000000
Fifth	78.000000
Five	80.000000
Four	68.750000
Nine	41.500000
Seven	77.600000
Six	82.571429
Three	73.666667

dtype: float64

```
1 # count of students in each class
2 df["class"].value_counts()
```



	count
class	
Seven	10
Four	8
Six	7
Three	3
Five	2
Nine	2
Fifth	1
Eight	1

dtype: int64

```
1 # average "mark" for each "gender"
2 df.groupby("gender")["mark"].mean()
```



	mark
gender	
female	77.312500
male	71.588235

dtype: float64

Integer based selection

Using .iloc

```
1 #accessing single row by index position
2 df.iloc[3]
```



	3
id	4
name	Krish Star
class	Four
mark	60
gender	female

dtype: object

```
1 #selecting a range of rows
2 df.iloc[2:5] #exclusive of row 5
```



	id	name	class	mark	gender
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

```
1 #selecting a column using .iloc
2 df.iloc[:, 2]
```



	class
0	Four
1	Three
2	Three
3	Four
4	Four
5	Four
6	Fifth
7	Five
8	Six
9	Four
10	Six
11	Six
12	Seven
13	Seven
14	NaN
15	Four
16	Six
17	Five
18	Nine
19	Nine
20	Four
21	Seven
22	Eight
23	Seven
24	Seven
25	Seven
26	Three
27	Seven
28	Seven
29	Six
30	Four

```
1 #selecting range of columns and rows
2 df.iloc[2:5, 1:3]
```



	name	class
2	Arnold	Three
3	Krish Star	Four
4	John Mike	Four

```
1 #do you think negative indexing will work on .iloc?
2 #using negative indexing to retrieve the data
3 df.iloc[-1, -1]
```




```
'female'
```

Student Activity Questions


1. use loc to select row where marks are greater than 85
2. use iloc to select the first 10 rows
3. use .loc to select name and mark column for all students
4. use .iloc to select the first three columns

```
1 #use .loc to select row where marks are greater than 85
2 df.loc[df["mark"] > 85]
```




	id	name	class	mark	gender
10	11	Ronald	Six	89	female
11	12	Recky	Six	94	female
12	13	Kty	Seven	88	female
13	14	Bigy	Seven	88	female
14	15	Tade Row	NaN	88	male
15	16	Gimmy	Four	88	male
24	25	Giff Tow	Seven	88	male
27	28	Rojj Base	Seven	86	female
30	31	Marry Toeey	Four	88	male
31	32	Binn Rott	Seven	90	female
32	33	Kenn Rein	Six	96	female
34	35	Rows Noump	Six	88	female

```
1 #use .iloc to select the first 10 rows
2 df.iloc[0:10]
```



	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
6	7	My John Rob	Fifth	78	male
7	8	Asruid	Five	85	male
8	9	Tes Qry	Six	78	NaN
9	10	Big John	Four	55	female

```
1 #use .loc to select name and mark column for all students
2 df.loc[:,["name", "mark"]]
```

 Show hidden output

```
1 #use .iloc to select the first three columns
2 df.iloc[:,0:3].head()
```



	id	name	class
0	1	John Deo	Four
1	2	Max Ruin	Three
2	3	Arnold	Three
3	4	Krish Star	Four
4	5	John Mike	Four

```
1 Start coding or generate with AI.
```

✓ Hands-On Activity: Extracting Data from the Student Dataset

Task 1: Extract Specific Rows

1. Use `.loc[]` to **select rows where the mark is greater than 85**.
2. Use `.iloc[]` to **select the first 10 rows**.

🚩 Expected Code, Task 1:

```
# Using .loc
df.loc[df['mark'] > 85]

# Using .iloc
df.iloc[:10]
```

▼ Task 2: Select Specific Columns

1. Use `.loc[]` to **select "name" and "mark" columns** for all students.
2. Use `.iloc[]` to **select the first 3 columns**.

🚩 Expected Code, Task 2:

```
# Using .loc
df.loc[:, ['name', 'mark']]

# Using .iloc
df.iloc[:, :3]
```

▼ Task 3: Filtering Based on Conditions

1. Select **all female students who scored more than 75** using `.loc[]`.
2. Find students from index **10 to 20** and select only **name and mark** using `.iloc[]`.

🚩 Expected Code, Task 3:

```
# Using .loc
df.loc[(df['gender'] == 'female') & (df['mark'] > 75)]

# Using .iloc
df.iloc[10:21, [1, 3]] # 1st and 3rd column (name and mark)
```

▼ Bonus Task: Sorting & Filtering

- Find **top 5 students based on marks** using `.loc[]` and `sort_values()`.
- Find **students from the last 5 rows** using `.iloc[]`.

🚩 Expected Code, Bonus Task:

```
# Top 5 students using .loc and sorting
df.loc[:, ['name', 'mark']].sort_values(by='mark', ascending=False).head(5)

# Last 5 students using .iloc
df.iloc[-5:]
```

▼ 6. Summary

Feature	.loc[]	.iloc[]
Selection Type	Label-based (column/row names)	Index-based (position numbers)
Includes Last Index?	Yes	No
Supports Filtering?	Yes	No
Best For?	Named columns/rows	Numerical indexing

7. Assessment & Discussion

Quiz Questions

1. How do you select only the **name** and **mark** columns using `.loc[]` ?
2. How do you select the **first 5 rows** using `.iloc[]` ?
3. What is the difference between `.loc[]` and `.iloc[]` ?
4. How do you select all students with marks **greater than 80** using `.loc[]` ?
5. How do you use `.iloc[]` to get the **last 3 rows**?

✓ Discussion

- When should you use `.loc[]` instead of `.iloc[]` ?
- Why does `.iloc[]` **exclude** the last index?

8. Further Learning

- **Pandas Documentation** → [Pandas .loc and .iloc](#)
- **Interactive Pandas Exercise** → [Kaggle Pandas Course](#)
- **Python for Data Analysis Book** → [O'Reilly Pandas Guide](#)

```
1 df.isnull().sum()
```



	0
id	0
name	1
class	1
mark	0
gender	2

dtvna: int64

```
1 #filling missing values in gender column with "not specified"
2 df["gender"].fillna("not specified", inplace=True)
3 df
```




	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

```
1 #replacing missing values in "mark" column with mean of all the values in the "mark" column
2 #one step method
3 #two step method
```

```
1 #fill missing values
2 df["class"].fillna("Unknown", inplace=True)
3 df
```

```
1 #removing duplicates
2 df.drop_duplicates(inplace=True)
3 df
```

```
1 #inconsistent data
2 df['class'] = df['class'].str.upper()
3 df
```



	id	name	class	mark	gender
0	1	John Deo	FOUR	75	female
1	2	Max Ruin	THREE	85	male
2	3	Arnold	THREE	55	male
3	4	Krish Star	FOUR	60	female
4	5	John Mike	FOUR	60	female
5	6	Alex John	FOUR	55	male
6	7	My John Rob	FIFTH	78	male
7	8	Asruid	FIVE	85	male
8	9	Tes Qry	SIX	78	not specified
9	10	Big John	FOUR	55	female
10	11	Ronald	SIX	89	female
11	12	Recky	SIX	94	female
12	13	Kty	SEVEN	88	female
13	14	Bigy	SEVEN	88	female
14	15	Tade Row	NaN	88	male
15	16	Gimmy	FOUR	88	male
16	17	Tumyu	SIX	54	male