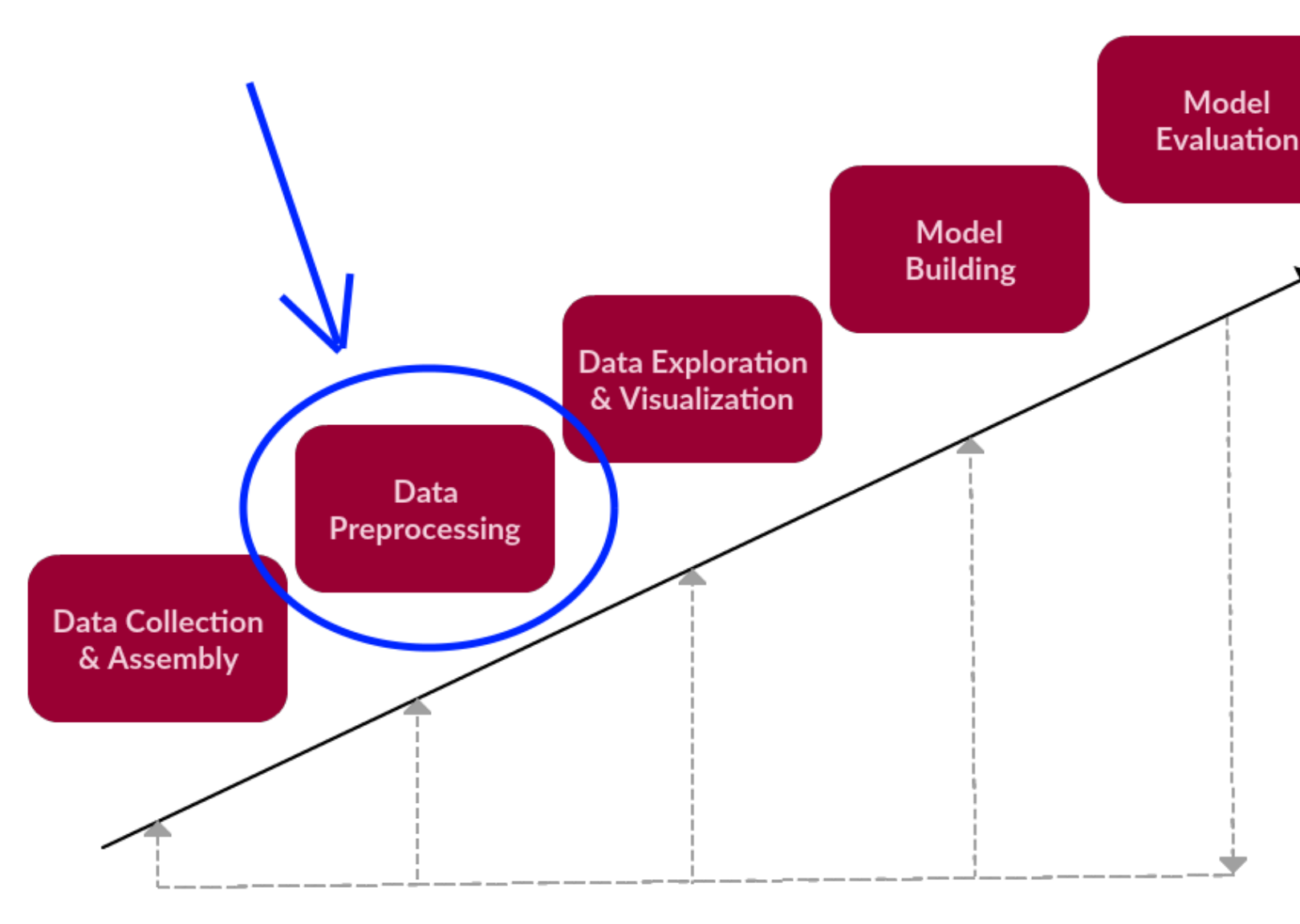
Data Preprocessing - Data Analysis

Data Preprocessing

It is used for maintaining the **Quality** of the data. It includes important factors like -

- Selecting the valid data variables
- Data editing is important in some aspects
- Maintaining uniformity in data values
- Manipulation of the data for achieving the above factors (Data Wrangling)



Credits - Image from Internet

In [ ]:

Classroom Management

Imagine you are the new teacher freshly appointed to manage the classroom.

You want to know how many students are good in -

```
1. sports
2. academics
3. creative work
4. marketing
```

Based on the students data, you have to conclude **who** can do well in **what**.

Dataset description

You are given the **data of the students** that included the following variables -

- Age
- Gender
- Address
- Father's occupation
- Mother's occupation
- Place of birth
- Height - ft
- Weight - kg
- Prev sports performance
- Prev academics performance
- Voluntary experience
- Extra co-curricular activities
- Arts and Design

Note - For our conveneince all the data values are numericals.

In [ ]:

How do we convert a **" feeling "** into a number?

- We can measure a **" feeling "** into a number through a **" scale range "**
- If the scale is **1** to **4**, then we can term -
  - 1 → Not Satisfied
  - 2 → Slightly Satisfied
  - 3 → Satisfied
  - 4 → Highly Satisfied

In [ ]:

Sports

Scenario 1

Considering the variables that are direclty related -

- Height
- Weight
- Prev sports performance

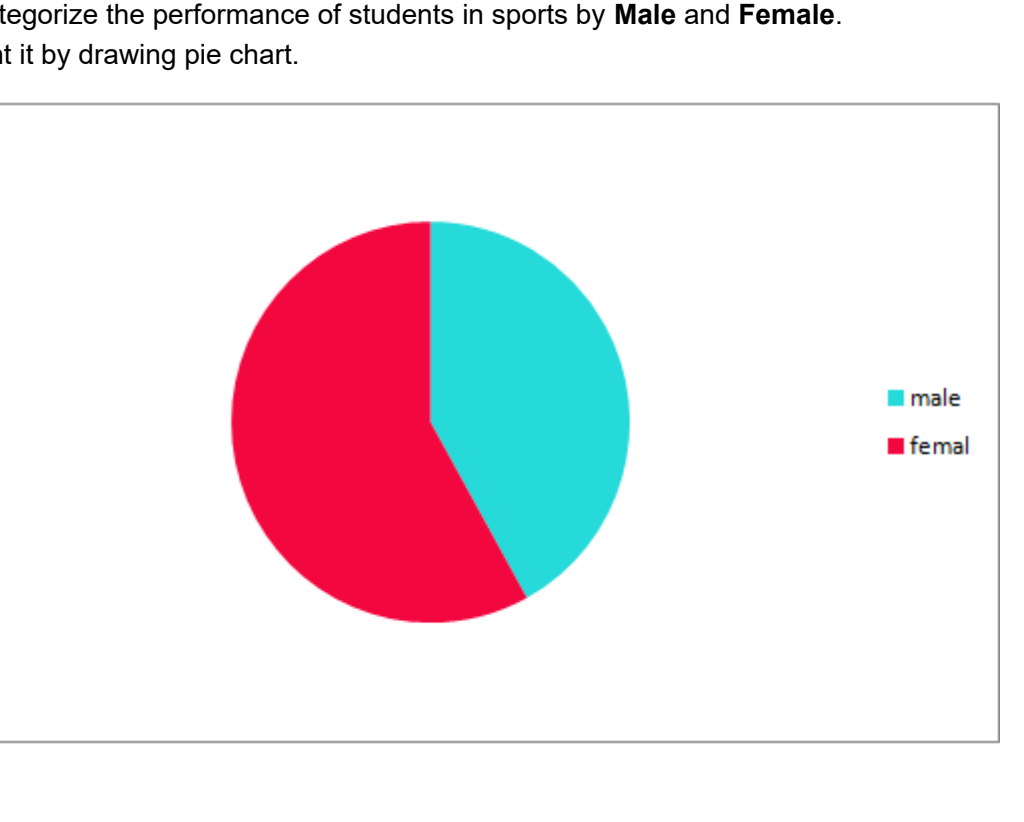
Based on this, you can only get the information of a student irrespective of **gender**.

Scenario 2

Considering the other important factors like **gender** in order to categorize as per **Male related sports** and **Female related sports**.

- Gender
- Height
- Weight
- Prev sports performance

1. Based on this, you can categorize the performance of students in sports by **Male** and **Female**.
2. Visually, you can represent it by drawing pie chart.



Credits - Image from Internet

Scenario 3

If you want to do further research on how good the person is performing in other areas, you can do so by considering -

- Gender
- Height
- Weight
- Prev sports performance
- Voluntary experience
- Extra co-curricular activities
- Prev academics performace (may be or may not be)

1. With this, you conclude the overall students performance on sports
2. Since you are a kind teacher and well wisher of student, you can give the student a proper career guidance.

In [ ]:

Academics

Scenario 1

Considering the variables that are directly related -

- Prev academics peroformance

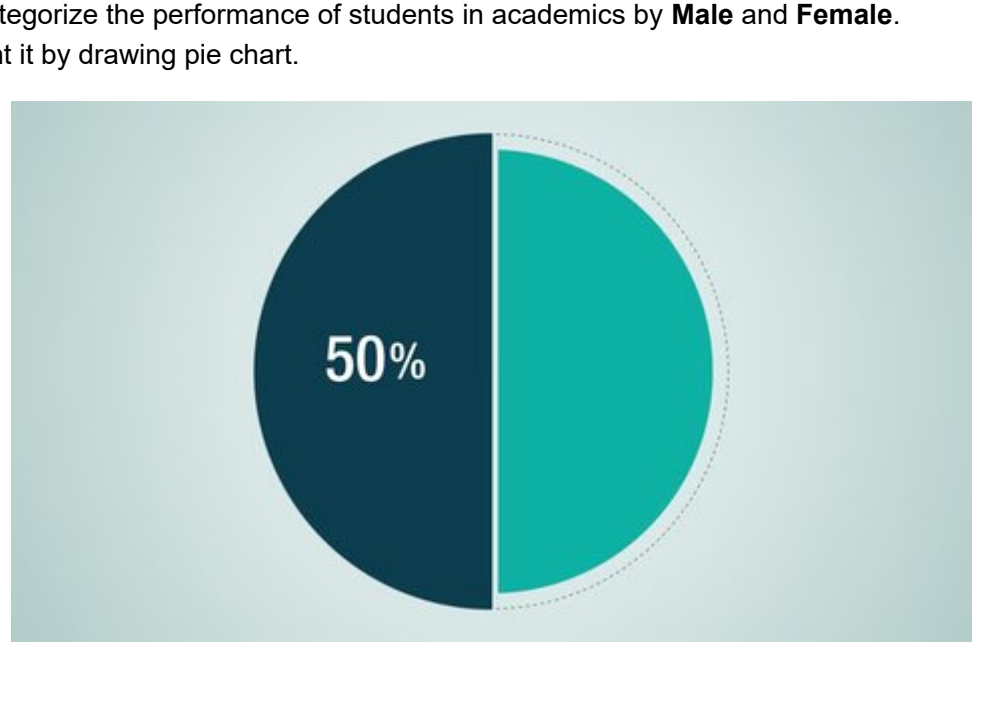
Based on this, you can only get the information of student irrespective of **gender**.

Scenario 2

Considering the other important factors like **gender** to categorize **Male** and **Female** separately.

- Gender
- Prev academics performance

1. Based on this, you can categorize the performance of students in academics by **Male** and **Female**.
2. Visually, you can represent it by drawing pie chart.



Credits - Image from Internet

Scenario 3

If you want to further research on why a particular student is **lagging behind** or **excelling ahead**, you can do so by considering -

- Address
- Father's occupation
- Mother's occupation
- Prev academics performance
- Gender (for categorizing in terms of gender)

and later on, you can decide whether to change your teaching methodology or not.

In [ ]:

Note -

- Data Analyst should be wise enough to select the important data variables.
- This helps to get proper insights pertaining to the problem statement that he/she is assigned to do.

In [ ]:

Let's make our hands dirty

data source → http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_020108_HeightsWeights

In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
df = pd.read_csv('students_hw.csv')
df.head()
```

Out[2]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

Check the length of the `df`

In [3]:
```python
# shape
df.shape
```

Out[3]: `(200, 2)`

Data Preprocessing

Check for `NaN`

In [4]:
```python
# isnull().any()
df.isnull().any()
```

Out[4]:
```
Height(Inches)    True
Weight(Pounds)    True
dtype: bool
```

In [5]:
```python
# list of columns
df.columns
```

Out[5]: `Index(['Height(Inches)', 'Weight(Pounds)'], dtype='object')`

In [6]:
```python
def check_for_nan(dframe):
    """
    dframe → pandas data frame object
    returns `nan_places` a dictionary of column names and the `nan_indices`
    """
    nan_places = {}

    for col in dframe.columns:
        indices = list(np.where(dframe[col].isnull())[0])
        nan_places[col] = indices

    return nan_places
```

In [7]:
```python
check_for_nan(dframe=df)
```

Out[7]: `{'Height(Inches)': [10, 32], 'Weight(Pounds)': [19]}`

- In the column `Height(Inches)`, there are two `NaN` values at indices `10` and `32`.
- In the column `Weight(Pounds)`, there is one `NaN` value at index `19`.

What can we do for those?

- Remove the entire row which ever column has a `NaN`.

For this, we will remove the rows which ever column has `NaN`. In total, there are 3 rows that need to be removed.

Remove 3 rows

- axis (0) → row
- axis (1) → column

In [8]:
```python
# pdf
# dropna
pdf = df.dropna(axis=0)
```

check the length of `pdf`

In [9]:
```python
# shape
pdf.shape
```

Out[9]: `(197, 2)`

Since the index of the data frame is not in order, we need to reindex the index values to get the perfect order.

In [10]:
```python
# head
pdf.head(12)
```

Out[10]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30
5	68.70	123.30
6	69.80	141.49
7	70.01	136.46
8	67.90	112.37
9	66.78	120.67
11	67.62	114.14
12	68.30	125.61

Reset the index

In [11]:
```python
# rdf
# reset with drop
pdf = pdf.reset_index(drop=True)
```

In [12]:
```python
# shape
pdf.shape
```

Out[12]: `(197, 2)`

In [13]:
```python
# head
pdf.head(12)
```

Out[13]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30
5	68.70	123.30
6	69.80	141.49
7	70.01	136.46
8	67.90	112.37
9	66.78	120.67
10	67.62	114.14
11	68.30	125.61

Check if `Height(Inches)` < 40

In [14]:
```python
# inch_thresh
inch_thresh = 40
```

In [15]:
```python
# filter with <
pdf[pdf['Height(Inches)'] < 40]
```

Out[15]:

	Height(Inches)	Weight(Pounds)
68	30.84	134.02
93	36.29	120.03

Remove the rows where `Height(Inches)` < 40

- In the above case, we can see two values where height is less than 40.
- We remove by specifying the index values in `drop()` method.

In [16]:
```python
# drop by index
rdf = pdf.drop(index=[68, 93], axis=0)
```

In [17]:
```python
# shape
rdf.shape
```

Out[17]: `(195, 2)`

Reset the index

In [18]:
```python
# hdf
# drop = True
hdf = rdf.reset_index(drop=True)
```

In [19]:
```python
# shape
hdf.shape
```

Out[19]: `(195, 2)`

Since the index of the data frame is not in order, we need to reindex the index values to get the perfect order.

Categorize the data

In [20]:
```python
# avg_height → mean
avg_height = hdf['Height(Inches)'].mean()
```

In [21]:
```python
avg_height
```

Out[21]: `67.95584615384617`

In [22]:
```python
# round()
avg_height = round(avg_height)
```

In [23]:
```python
avg_height
```

Out[23]: `68`

In [24]:
```python
# write code
# height_cat -> consider round()

height_cat = []

for i in hdf['Height(Inches)'].to_list():
    if round(i) < avg_height:
        height_cat.append('Short')
    elif round(i) == avg_height:
        height_cat.append('Average')
    else:
        height_cat.append('Tall')
```

In [25]:
```python
# check len
len(height_cat)
```

Out[25]: `195`

In [26]:
```python
# display
print(height_cat)
```

```
['Short', 'Tall', 'Tall', 'Average', 'Average', 'Tall', 'Tall', 'Tall', 'Average', 'Short', 'Averag
e', 'Average', 'Short', 'Average', 'Tall', 'Short', 'Tall', 'Tall', 'Average', 'Tall', 'Short', 'Aver
age', 'Average', 'Short', 'Tall', 'Short', 'Short', 'Short', 'Tall', 'Short', 'Tall', 'Tall', 'Tall',
'Short', 'Average', 'Average', 'Short', 'Tall', 'Short', 'Tall', 'Average', 'Short', 'Tall', 'Short',
'Average', 'Tall', 'Tall', 'Tall', 'Short', 'Tall', 'Tall', 'Short', 'Tall', 'Tall', 'Short', 'Shor
t', 'Average', 'Short', 'Tall', 'Average', 'Short', 'Tall', 'Average', 'Tall', 'Average', 'Short', 'S
hort', 'Short', 'Tall', 'Short', 'Average', 'Short', 'Average', 'Short', 'Tall', 'Average', 'Tall',
'Short', 'Tall', 'Short', 'Tall', 'Tall', 'Short', 'Tall', 'Tall', 'Average', 'Average', 'Tall', 'Tal
l', 'Tall', 'Tall', 'Tall', 'Short', 'Short', 'Tall', 'Short', 'Short', 'Average', 'Short', 'Averag
e', 'Average', 'Average', 'Short', 'Tall', 'Short', 'Average', 'Tall', 'Tall', 'Short', 'Tall', 'Tal
l', 'Tall', 'Tall', 'Short', 'Tall', 'Short', 'Short', 'Tall', 'Tall', 'Average', 'Short', 'Short',
'Average', 'Tall', 'Tall', 'Average', 'Tall', 'Tall', 'Short', 'Tall', 'Average', 'Short', 'Short',
'Tall', 'Tall', 'Tall', 'Short', 'Average', 'Average', 'Short', 'Average', 'Tall', 'Short', 'Tall',
'Tall', 'Tall', 'Short', 'Short', 'Short', 'Tall', 'Short', 'Tall', 'Average', 'Tall', 'Short', 'Shor
t', 'Short', 'Tall', 'Average', 'Short', 'Short', 'Average', 'Short', 'Short', 'Short', 'Tall', 'Aver
age', 'Average', 'Short', 'Tall', 'Short', 'Short', 'Short', 'Short', 'Short', 'Short', 'Short', 'Sho
rt', 'Tall', 'Short', 'Short', 'Tall', 'Short', 'Short', 'Average', 'Tall', 'Short', 'Short', 'Shor
t', 'Tall', 'Short', 'Short', 'Average', 'Average', 'Tall']
```

Make a new column `height_category` in the dataframe - `hdf`

In [27]:
```python
hdf.head()
```

Out[27]:

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

In [28]:
```python
# new column
hdf['height_cat'] = height_cat
```

In [29]:
```python
# check head
hdf.head()
```

Out[29]:

	Height(Inches)	Weight(Pounds)	height_cat
0	65.78	112.99	Short
1	71.52	136.49	Tall
2	69.40	153.03	Tall
3	68.22	142.34	Average
4	67.79	144.30	Average

Plotting the pie chart to show

- how many are short
- how many are tall

Take `value_counts()` of `height_category` variable

In [30]:
```python
hdf['height_cat'].value_counts()
```

Out[30]:
```
Short      79
Tall       74
Average    42
Name: height_cat, dtype: int64
```

In [31]:
```python
# df_pie → to_frame
df_pie = hdf['height_cat'].value_counts().to_frame()
```

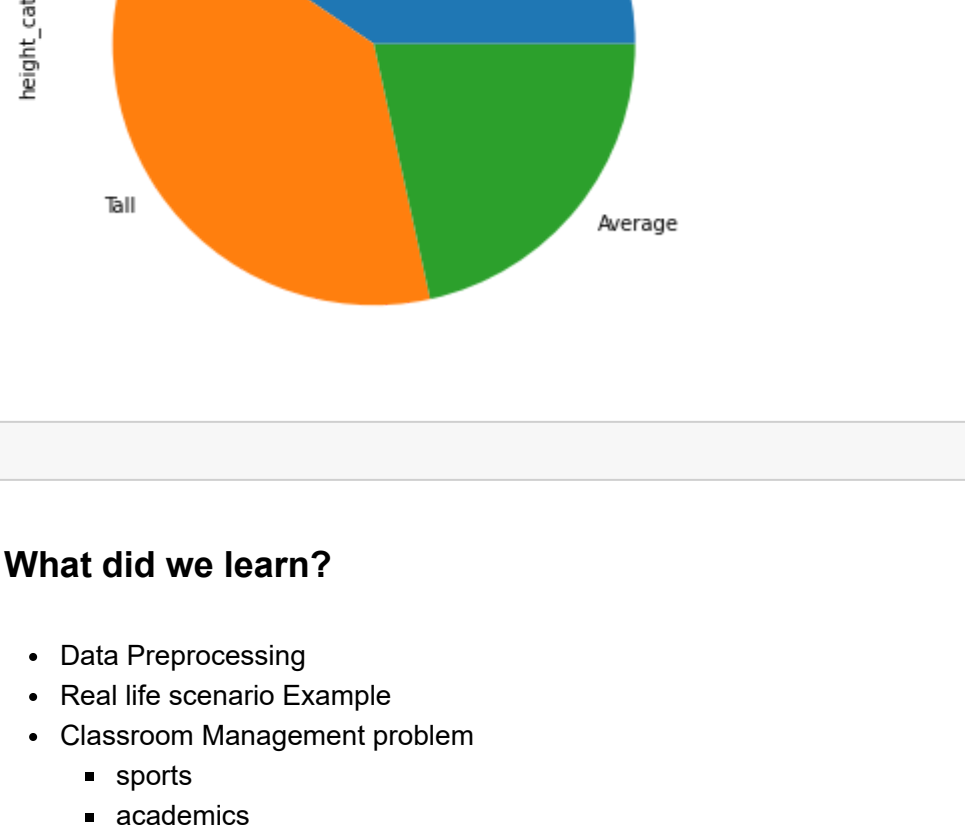
In [32]:
```python
# display df_pie
df_pie
```

Out[32]:

	height_cat
Short	79
Tall	74
Average	42

In [33]:
```python
# plot pie of df_pie with size (width=10, height=6)
df_pie.plot(kind='pie', figsize=(10, 6), subplots=True)
```

Out[33]: `array([<AxesSubplot:ylabel='height_cat'>], dtype=object)`



In [ ]:

What did we learn?

- Data Preprocessing
- Real life scenario Example
- Classroom Management problem
  - sports
  - academics
  - creative work
  - marketing
- Getting hands dirty by writing code
- Plotting the pie chart for showing categorization