Lab2

October 21, 2022

1 CS418 LAB 2

1.1 DESCRIPTIVE STATISTICS

1.1.1 Loading the data into dataFrame from the CSV File

```
[1]: import pandas as pd
```

```
[2]: df = pd.read_csv (r'/Users/praveenrajveluswami/Documents/CS418 - IDS/lab2/
     ↪heroes_information.csv')
```

1.1.2 Data type description of each column

- **ID:** Discrete
- **name:** Nominal
- **Gender:** Nominal
- **Eye color:** Nominal
- **Race:** Nominal
- **Hair color:** Nominal
- **Height:** Continuous
- **Publisher:** Nominal
- **Skin color:** Nominal
- **Alignment:** Ordinal
- **Weight:** Continuous

1.1.3 Compute the following measures for each column : Central tendencies and Dispersion

- For several columns here, especially the nominal and ordinal data types: name, Gender, eye color, Race, Hair color, Publisher, Skin color and Alignment, central tendencies and dispersion except mode cannot be calculated since their values are not numerical.
- Proposed solution: Replace their values with integers i.e. 0,1,2,3… We replace blank/invalid values with 0.
- We follow this approach for columns other than 'name' (name is neither qualitative not quantitative) and compute these values. Although we would get a mean,median,etc. values for 'Gender',etc., this will not logically make sense.

Column name: ID **a.i. Mean :**

Although mean can be computer for this column as the values are integers, it is not going to be helpful in gaining any insights about the data since these are just indexes of rows in our data.

```
[3]: print(df["ID"].mean())
```

```
366.5
```

a.ii. Median :

Since the values are just indexes of rows, the median is simply the total number of records divided by 2. It gives us the row at the middle of our dataset although it doesn't serve much purpose.

```
[4]: print(df["ID"].median())
```

```
366.5
```

a.iii. Mode :

The values in this column are never repeated. This means frequency of each value is 1.

```
[5]: print(df["ID"].mode())
```

```
0        0
1        1
2        2
3        3
4        4
      …
729    729
730    730
731    731
732    732
733    733
Length: 734, dtype: int64
```

b.i. Standard Deviation :

```
[6]: print(df["ID"].std())
```

```
212.03183723205342
```

b.ii. Variance :

```
[7]: print(df.var()['ID'])
```

```
44957.5
```

b.iii. IQR :

```
[8]: import numpy as np
     q75, q25 = np.percentile(df['ID'], [75 ,25])
     iqr = q75 - q25
     print(iqr)
```

```
366.5
```

b.iv. Skew :

[9]:
```
print(df['ID'].skew())
```

```
0.0
```

Column name: Name **a.i. Mean :**

Since this column has nominal data, it is not possible to compute mean. The values can also not be encoded into numercial form.

a.ii. Median :

Since this column has nominal data, it is not possible to compute median. The values can also not be encoded into numercial form.

a.iii. Mode :

Although the values are nominal, frequency for each value can be calculated through which mode can be computed. In this case, this might give insights into duplicate records.

[10]:
```
print(df["name"].mode())
```

```
0       Goliath
1    Spider-Man
dtype: object
```

b.i. Standard Deviation :

Nominal data. Standard Deviation cannot be computed.

b.ii. Variance :

Nominal data. Variance cannot be computed.

b.iii. IQR :

Nominal data. IQR cannot be computed.

b.iv. Skew :

Nominal data. Skew cannot be computed.

Column name: Gender Gender can be encoded into numerical values. For example, male:1 ; female: 2

[11]:
```
df = df.replace('Male', 1)
```

[12]:
```
df = df.replace('Female', 2)
```

[13]:
```
df['Gender'] = df['Gender'].replace('-',0)
```

[14]:
```
df['Gender'] = df['Gender'].astype('int')
```

Column name: Eye color Eye color, being categorical data, can be encoded into numerical values. The unique values in the column were retrieved using .unique() method. These can be encoded as 1,2,3...

```
[15]: df['Eye color'] = df['Eye color'].replace(['yellow', 'blue', 'green', 'brown',␣
       ↪'-', 'red', 'violet', 'white',
             'purple', 'black', 'grey', 'silver', 'yellow / red',
             'yellow (without irises)', 'gold', 'blue / white', 'hazel',
             'green / blue', 'white / red', 'indigo', 'amber', 'yellow / blue',
             'bown'],[1,2,3,4,0,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22])
```

```
[16]: df['Eye color'] = df['Eye color'].astype('int')
```

Column name: Race This is same as the Eye color column. Race, being categorical data, can be encoded into numerical values. The unique values in the column were retrieved using .unique() method. These can be encoded as 1,2,3...

```
[17]: df['Race'] = df['Race'].replace(['Human', 'Icthyo Sapien', 'Ungaran', 'Human /␣
       ↪Radiation',
             'Cosmic Entity', '-', 'Cyborg', 'Xenomorph XX121', 'Android',
             'Vampire', 'Mutant', 'God / Eternal', 'Symbiote', 'Atlantean',
             'Alien', 'Neyaphem', 'New God', 'Alpha', 'Bizarro', 'Inhuman',
             'Metahuman', 'Demon', 'Human / Clone', 'Human-Kree',
             'Dathomirian Zabrak', 'Amazon', 'Human / Cosmic',
             'Human / Altered', 'Kryptonian', 'Kakarantharaian',
             'Zen-Whoberian', 'Strontian', 'Kaiju', 'Saiyan', 'Gorilla',
             'Rodian', 'Flora Colossus', 'Human-Vuldarian', 'Asgardian',
             'Demi-God', 'Eternal', 'Gungan', 'Bolovaxian', 'Animal',
             'Czarnian', 'Martian', 'Spartoi', 'Planet', 'Luphomoid',
             'Parademon', 'Yautja', 'Maiar', 'Clone', 'Talokite', 'Korugaran',
             'Zombie', 'Human-Vulcan', 'Human-Spartoi', 'Tamaranean',
             'Frost Giant', 'Mutant / Clone', "Yoda's␣
       ↪species"],[1,2,3,4,5,0,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,
```

```
[18]: df['Race'] = df['Race'].astype('int')
```

Column name: Hair color This is same as the previous 2 columns. Hair color, being categorical data, can be encoded into numerical values. The unique values in the column were retrieved using .unique() method. These can be encoded as 1,2,3...

```
[19]: df['Hair color'] = df['Hair color'].replace(['No Hair', 'Black', 'Blond',␣
       ↪'Brown', '-', 'White', 'Purple',
             'Orange', 'Pink', 'Red', 'Auburn', 'Strawberry Blond', 'black',
             'Blue', 'Green', 'Magenta', 'Brown / Black', 'Brown / White',
             'blond', 'Silver', 'Red / Grey', 'Grey', 'Orange / White',
             'Yellow', 'Brownn', 'Gold', 'Red / Orange', 'Indigo',
       'Red / White', 'Black /␣
↪Blue'],[1,2,3,4,0,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29])
```

[20]:
```
df['Hair color'] = df['Hair color'].astype('int')
```

Column name: Height This is a continuous data. Hence, calculating central tendencies and dispersion shouldn't be a problem. Several rows have a value '-99' which is going to affect the below values. We will be handling these values during the data cleaning process.

Column name: Publisher Publisher, being categorical data, can be encoded into numerical values. The unique values in the column were retrieved using .unique() method. These can be encoded as 1,2,3...

[21]:
```
df['Publisher'] = df['Publisher'].replace(['Marvel Comics', 'Dark Horse␣
↪Comics', 'DC Comics', 'NBC - Heroes',
       'Wildstorm', 'Image Comics', 'nan', 'Icon Comics', 'SyFy',
       'Hanna-Barbera', 'George Lucas', 'Team Epic TV', 'South Park',
       'HarperCollins', 'ABC Studios', 'Universal Studios', 'Star Trek',
       'IDW Publishing', 'Shueisha', 'Sony Pictures', 'J. K. Rowling',
       'Titan Books', 'Rebellion', 'Microsoft', 'J. R. R.␣
↪Tolkien'],[1,2,3,4,5,6,0,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24])
```

[22]:
```
df['Publisher'] = df['Publisher'].replace(np.nan, 0)
```

[23]:
```
df['Publisher'] = df['Publisher'].astype('int')
```

Column name: Skin color Skin color, being categorical data, can be encoded into numerical values. The unique values in the column were retrieved using .unique() method. These can be encoded as 1,2,3...

[24]:
```
df['Skin color'] = df['Skin color'].replace(['-', 'blue', 'red', 'black',␣
↪'grey', 'gold', 'green', 'white',
       'pink', 'silver', 'red / black', 'yellow', 'purple',
       'orange / white', 'gray', 'blue-white',␣
↪'orange'],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
```

[25]:
```
df['Skin color'] = df['Skin color'].astype('int')
```

Column name: Alignment

[26]:
```
df['Alignment'].unique()
```

[26]:
```
array(['good', 'bad', '-', 'neutral'], dtype=object)
```

[27]:
```
df['Alignment'] = df['Alignment'].replace('good', 1)
```

[28]:
```
df['Alignment'] = df['Alignment'].replace('bad', 2)
df['Alignment'] = df['Alignment'].replace('neutral', 3)
```

[29]:
```
df['Alignment'] = df['Alignment'].replace('-',0)
```

[30]:
```
df['Alignment'] = df['Alignment'].astype('int')
```

Column name: Weight This is a continuous data. Hence, calculating central tendencies and dispersion shouldn't be a problem. Several rows have a value '-99' which is going to affect the below values. We will be handling these values during the data cleaning process.

Computing the values for each of the above columns:

[31]:
```
for (columnName, columnData) in df[["Gender", "Eye color","Race","Hair␣
↪color","Height","Publisher","Skin color","Alignment","Weight"]].iteritems():

    mean = df[columnName].mean()
    median = df[columnName].median()
    mode = df[columnName].mode()
    std = df[columnName].std()
    var = df[columnName].var()
    skw = df[columnName].skew()

    q75, q25 = np.percentile(df[columnName], [75 ,25])
    iqr = q75 - q25

    print("\033[1m" + "Column : " + columnName + "\033[0m")
    print('---------------------------------')
    print('Mean :', mean)
    print('Median :', median)
    print('Mode :', mode)
    print('Standard Deviation :', std)
    print('Variance :', var)
    print('IQR :', iqr)
    print('Skew :', skw)
  ␣
↪print('=======================================================================')
```

```
Column : Gender
---------------------------------
Mean : 1.2329700272479565
Median : 1.0
Mode : 0    1
dtype: int64
Standard Deviation : 0.5080016292373221
Variance : 0.2580656553077736
```

```
IQR : 1.0
Skew : 0.307949299993484
========================================================================
Column : Eye color
---------------------------------
Mean : 2.9850136239782015
Median : 2.0
Mode : 0    2
dtype: int64
Standard Deviation : 3.1937941080673116
Variance : 10.200320804725475
IQR : 3.0
Skew : 2.492321498022643
========================================================================
Column : Race
---------------------------------
Mean : 5.8732970027247955
Median : 1.0
Mode : 0    0
dtype: int64
Standard Deviation : 11.488485048328451
Variance : 131.98528870566636
IQR : 8.75
Skew : 2.6983562409134008
========================================================================
Column : Hair color
---------------------------------
Mean : 3.587193460490463
Median : 2.0
Mode : 0    0
dtype: int64
Standard Deviation : 4.622431755281572
Variance : 21.366875332235477
IQR : 3.0
Skew : 2.4911001104075314
========================================================================
Column : Height
---------------------------------
Mean : 102.25408719346049
Median : 175.0
Mode : 0   -99.0
dtype: float64
Standard Deviation : 139.62454255007276
Variance : 19495.01288231708
IQR : 284.0
Skew : -0.02365759744219501
========================================================================
Column : Publisher
```

```
---------------------------------
Mean : 2.8079019073569484
Median : 1.0
Mode : 0    1
dtype: int64
Standard Deviation : 3.5810464697443325
Variance : 12.823893818468346
IQR : 2.0
Skew : 3.135892758441947
=======================================================================
Column : Skin color
---------------------------------
Mean : 0.5817438692098093
Median : 0.0
Mode : 0    0
dtype: int64
Standard Deviation : 2.0935147596577903
Variance : 4.382804048905016
IQR : 0.0
Skew : 4.16200426766068
=======================================================================
Column : Alignment
---------------------------------
Mean : 1.337874659400545
Median : 1.0
Mode : 0    1
dtype: int64
Standard Deviation : 0.555521809524758
Variance : 0.3086044808576616
IQR : 1.0
Skew : 1.0724462861527282
=======================================================================
Column : Weight
---------------------------------
Mean : 43.8551912568306
Median : 62.0
Mode : 0   -99.0
dtype: float64
Standard Deviation : 130.82373271995178
Variance : 17114.849042781385
IQR : nan
Skew : 1.674863003138279
=======================================================================
```

1.1.4 Cleaning dataset

A part of data cleaning has been done earlier in the previous section to calculate central tendencies and dispersion. —> Categorical data were numerically encoded while '-' values were encoded as 0.

There are so many rows which have blank/invalid values in certain columns. However, removing these rows will result in loosing a major chunk of our data. Therefore, the method followed here is to find useless/junk rows with many blank or invalid values. * The code snippet below looks for rows which have more than two columns with blank or invalid values. These rows are then removed from the data.

[32]:
```
df = df
```

[33]:
```
df = df.fillna(0)
for ind, row in df.iterrows():
    count = 0
    for column in df:
        if column != 'ID':
            if row[column] == 0:
                count = count + 1
            elif row[column] == -99:
                count = count + 1
    if count >= 2:
        df.drop(ind,axis=0,inplace=True)
#          print('deleting ',ind)
```

1. Meanwhile, in the other rows, these blank/invalid values are handled as follows:
    - Categorical data i.e. Gender, Eye color, Race, Hair color, Publisher, Skin color, Alignment : Blank/invalid values are replaced with the mode of the column, since logically, the row has more probability of having the value of mode than any other value in the column. In case the mode itself is 0, we replace it with the next most frequent value.
    - Continuous data i.e. Height and Weight : Blank/invalid values are replaced with the mean of the column, since this will make our assumption impact the distribution of data neither positively nor negatively.
2. Handling outliers: Outliers usually end up skewing the data, thus affecting the central tendencies and dispersion values. They also affect classification, prediction or any kind of inference. For example, there are few records which have a height around 800-1000 which are far from the other records. The idea is to replace these outliers with the upper and low quartiles respectively. Only continous data like height and weight are prone to outliers. Therefore, we will implement outlier removal only for those columns.

[34]:
```
# Confirming that the mode is not 0
for (columnName, columnData) in df.iteritems():
    if columnName != 'ID' and columnName != 'name':
        init_mode = df[columnName].value_counts().idxmax()
        if init_mode == 0:  ## Checking whether the mode itself is 0
            df_temp = df[df[columnName] != 0]   ## If mode is 0, remove the␣
 ↪zeroes
            df[columnName] = df[columnName].replace([0], df_temp[columnName].
 ↪value_counts().idxmax()) ## Replace zeroes in the original df with the new␣
 ↪mode
        else:
            df[columnName] = df[columnName].replace([0], df[columnName].
↪value_counts().idxmax())
```

Outlier removal Here, we replace outliers with median value so as to not affect the distribution since mean, mode can be influenced by the outliers. Replacing with upper bound or lower bound was eliminated because height and weight columns mostly will have outliers that are greater than Q3 more than outliers that are less than Q1. We want to avoid the upper bound getting over populated.

[35]:
```
# Height
Q1 = df['Height'].quantile(0.25)
Q3 = df['Height'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

for index, row in df.iterrows():
    if row['Height'] < lower_bound:
        df['Height'] = df['Height'].replace(row['Height'], df['Height'].
↪median())
    elif row['Height'] > upper_bound:
        df['Height'] = df['Height'].replace(row['Height'], df['Height'].
↪median())
```

[36]:
```
# Weight
Q1 = df['Weight'].quantile(0.25)
Q3 = df['Weight'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

for index, row in df.iterrows():
    if row['Weight'] < lower_bound:
        df['Weight'] = df['Weight'].replace(row['Weight'], df['Weight'].
↪median())
    elif row['Weight'] > upper_bound:
        df['Weight'] = df['Weight'].replace(row['Weight'], df['Weight'].
↪median())
```

1.1.5 Scatter plot

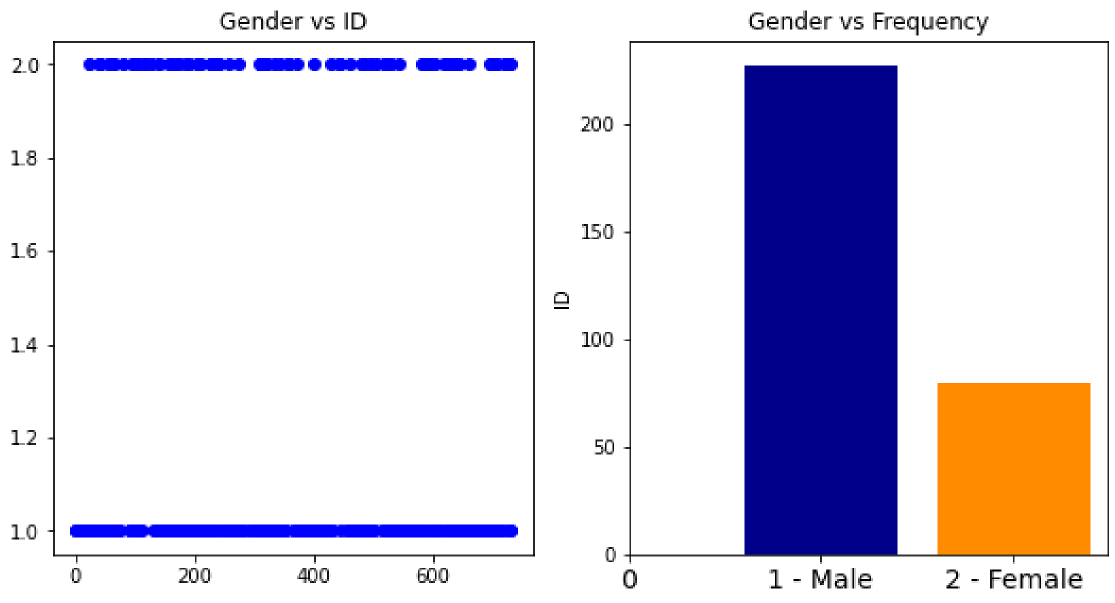
[37]:
```
import matplotlib.pyplot as plt
```

For each column, the first plot represents the scatter plot as asked in the question and the second plot is an alternative.

Gender

```
[38]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Gender vs ID')
ax1.scatter(df['ID'], df['Gender'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Gender vs Frequency')
ax2.bar(df['Gender'].value_counts().index,
       df['Gender'].value_counts().values,
       color = ['darkblue', 'darkorange'])
ax2.set_xticks(range(0, 3))
ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```



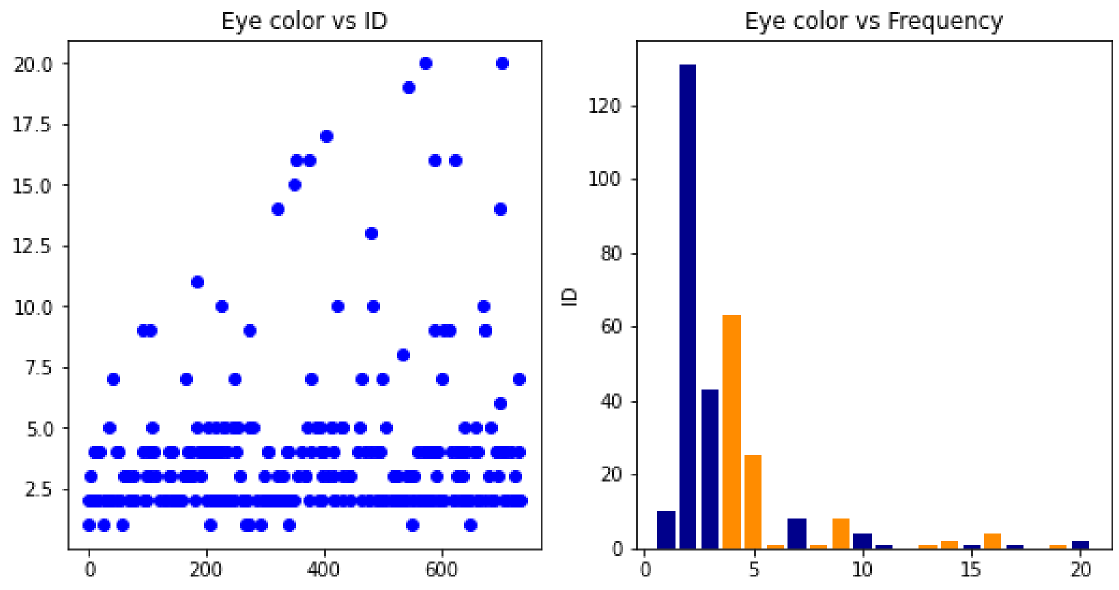
Learning from the first plot : Shows that 2 i.e. Females are less in number than 1 i.e. males. Both males and females are evenly distributed in the data. **Alternative plot :** The first plot does not give us any quantitave understanding of gender with respect to ID since ID is just an index. The alternative plot helps us compare the frequency of both the genders. **Learning from the alternative plot :** Apparently there are more males in the data than females.

Eye color

```
[39]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Eye color vs ID')
ax1.scatter(df['ID'], df['Eye color'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Eye color vs Frequency')
```

```
ax2.bar(df['Eye color'].value_counts().index,
       df['Eye color'].value_counts().values,
       color = ['darkblue', 'darkorange'])
# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```

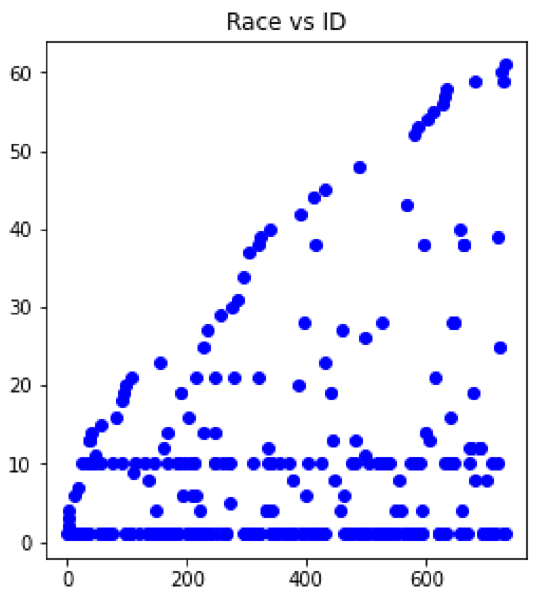


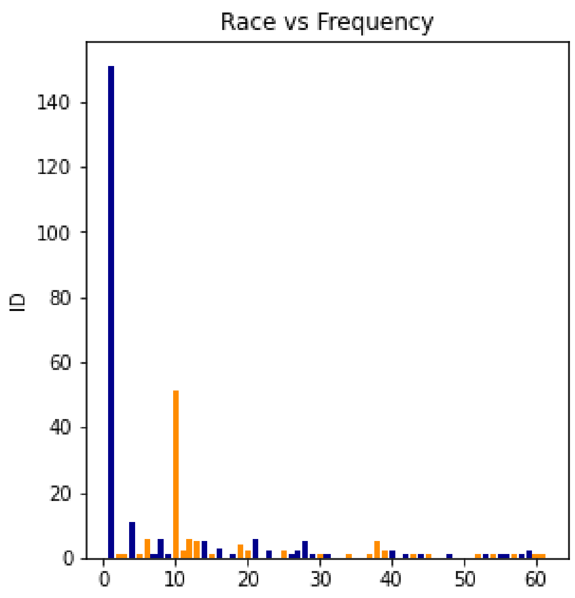
Learning from the first plot : Shows that the eye colors are not evenly distributed. The later part of the data seems to be more diverse than the initial part. The difference between the frequency of mode and the least occuring value is way too high. **Alternative plot :** The first plot does not give us any quantitave understanding of eye color with respect to ID since ID is just an index. The alternative plot helps us compare the frequency of the eye colors. **Learning from the alternative plot :** 2 i.e. Blue seems to be the most common eye color.

Race

```
[40]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Race vs ID')
ax1.scatter(df['ID'], df['Race'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Race vs Frequency')
ax2.bar(df['Race'].value_counts().index,
       df['Race'].value_counts().values,
       color = ['darkblue', 'darkorange'])
# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```





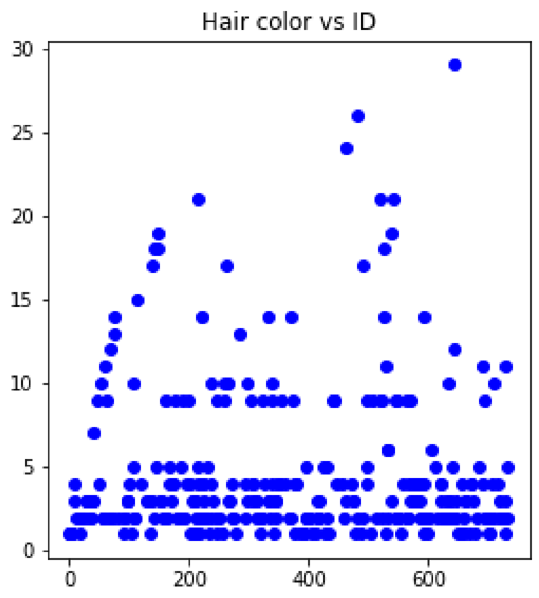
Learning from the first plot : Shows that Race is not evenly distributed. The later part of the data seems to be more diverse than the initial part. The difference between the frequency of mode and the least occuring value is way too high. The two dominant races are Human and Mutant. **Alternative plot :** The first plot does not give us any quantitave understanding of race with respect to ID since ID is just an index. The alternative plot helps us compare the frequency of the race. **Learning from the alternative plot :** 1 i.e. Human seems to be the most common race.

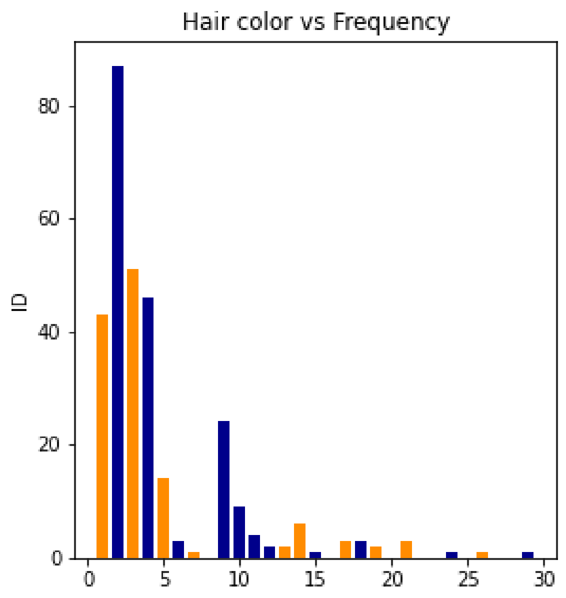
However in this case, the scatter plot makes more sense since even the least occuring race is able to be visualized.

Hair color

[41]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Hair color vs ID')
ax1.scatter(df['ID'], df['Hair color'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Hair color vs Frequency')
ax2.bar(df['Hair color'].value_counts().index,
       df['Hair color'].value_counts().values,
       color = ['darkblue', 'darkorange'])
# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```





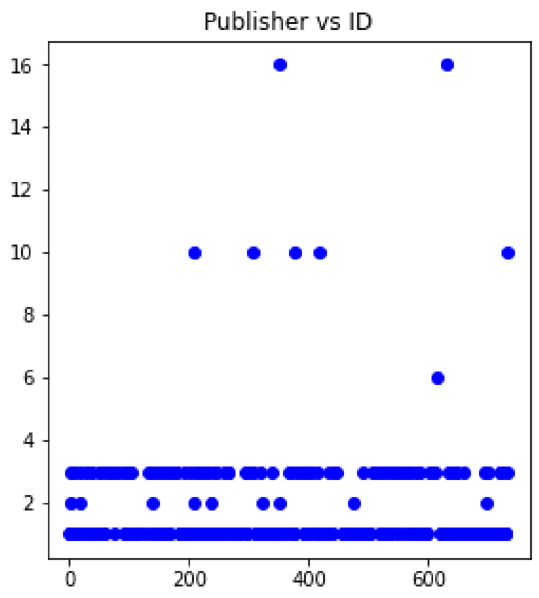
Learning from the first plot : Shows that Hair color is evenly distributed barring few hair colors like Indigo, Red/White and Black/Blue. The least occuring hair colors can be seen more easily. The most occuring colors are more evenly distributed than the less occuring ones. **Alternative plot :** The alternative plot helps us compare the frequency of the hair colors. **Learning from the alternative plot :** 2 i.e. Black seems to be the most common race.

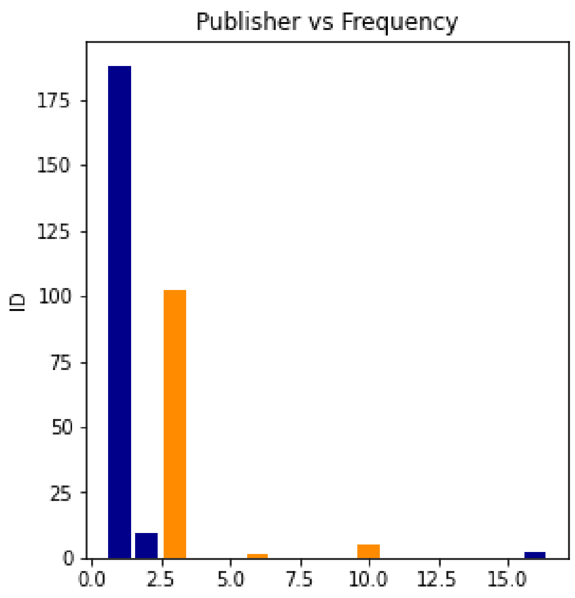
In this case too, the scatter plot makes more sense since even the least occuring race is able to be visualized.

Publisher

[42]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Publisher vs ID')
ax1.scatter(df['ID'], df['Publisher'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Publisher vs Frequency')
ax2.bar(df['Publisher'].value_counts().index,
       df['Publisher'].value_counts().values,
       color = ['darkblue', 'darkorange'])
# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```





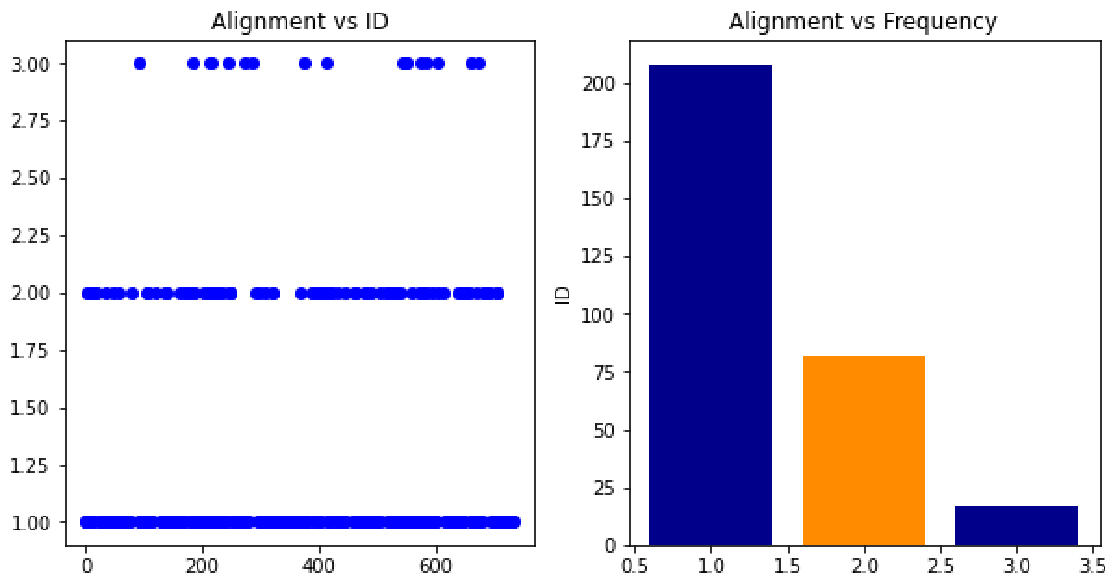
Learning from the first plot : Shows how the distribution of Publishers have been affected due to data cleaning. Records having values SouthPark, HaperCollins have been completely removed as part of data cleaning as thy might have had more than 2 invalid values. **Alternative plot :** The alternative plot helps us compare the frequency of the hair colors. **Learning from the alternative plot :** How only 3-4 values form the major part of the data

In this case too, both the plots are equally good in representing the nature of the data

Alignment

[43]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Alignment vs ID')
ax1.scatter(df['ID'], df['Alignment'], c ="blue")
plt.ylabel('ID')

ax2.title.set_text('Alignment vs Frequency')
ax2.bar(df['Alignment'].value_counts().index,
       df['Alignment'].value_counts().values,
       color = ['darkblue', 'darkorange'])
# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
plt.show()
```



Learning from the first plot : 3 i.e. neutral seems to be very low compared to good and bad. **Alternative plot :** The alternative plot helps us compare the frequency of the hair colors. **Learning from the alternative plot :** How only 3-4 values form the major part of the data

In this case, the scatter plot doesn't help much in understanding the difference between the distribution of 'good' and 'bad' i.e. 1 and 2. However, the alternative plot acheives this.

Height

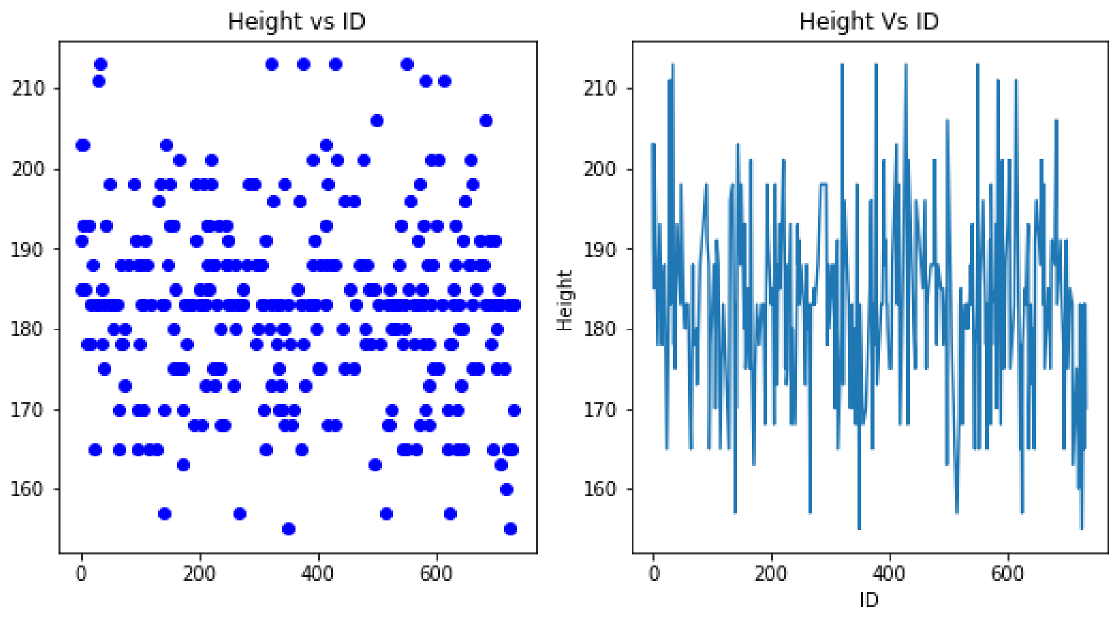
```
[44]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Height vs ID')
ax1.scatter(df['ID'], df['Height'], c ="blue")
plt.ylabel('Height')
plt.xlabel('ID')

# ax2.title.set_text('Height vs Frequency')
# ax2.bar(df['Height'].value_counts().index,
#         df['Height'].value_counts().values,
#         color = ['darkblue', 'darkorange'])

plt.plot(df['ID'], df['Height'])
plt.title('Height Vs ID')
plt.xlabel('ID')
plt.ylabel('Height')
plt.show()

# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
```

```
plt.show()
```



Learning from the first plot : Shows that the heights are mostly concentrated around 180-190. Records in the bottom of the data tend to be more taller than the ones on the top. We can also see the values being capped at the upper and lower bounds which we did during outlier removal. **Alternative plot :** The alternative plot helps us infer the trend in the height as we go down the records.While scatter plots doesn't perform well in showing the trends in the densely populated areas like height:200, the alternative plot achieves this. **Learning from the alternative plot :** The values are more skewed as we go further down our data.

Weight

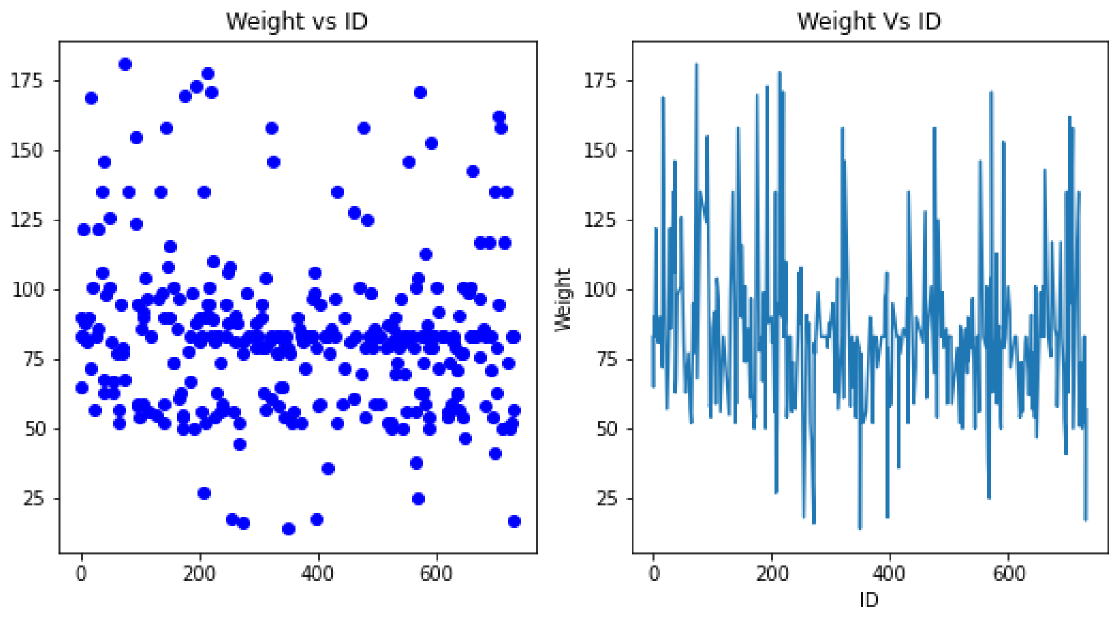
```
[45]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))
ax1.title.set_text('Weight vs ID')
ax1.scatter(df['ID'], df['Weight'], c ="blue")
plt.ylabel('Weight')
plt.xlabel('ID')



plt.plot(df['ID'], df['Weight'])
plt.title('Weight Vs ID')
plt.xlabel('ID')
plt.ylabel('Weight')
plt.show()

# ax2.set_xticks(range(0, 3))
# ax2.set_xticklabels(['0','1 - Male','2 - Female'], fontsize = 14);
```

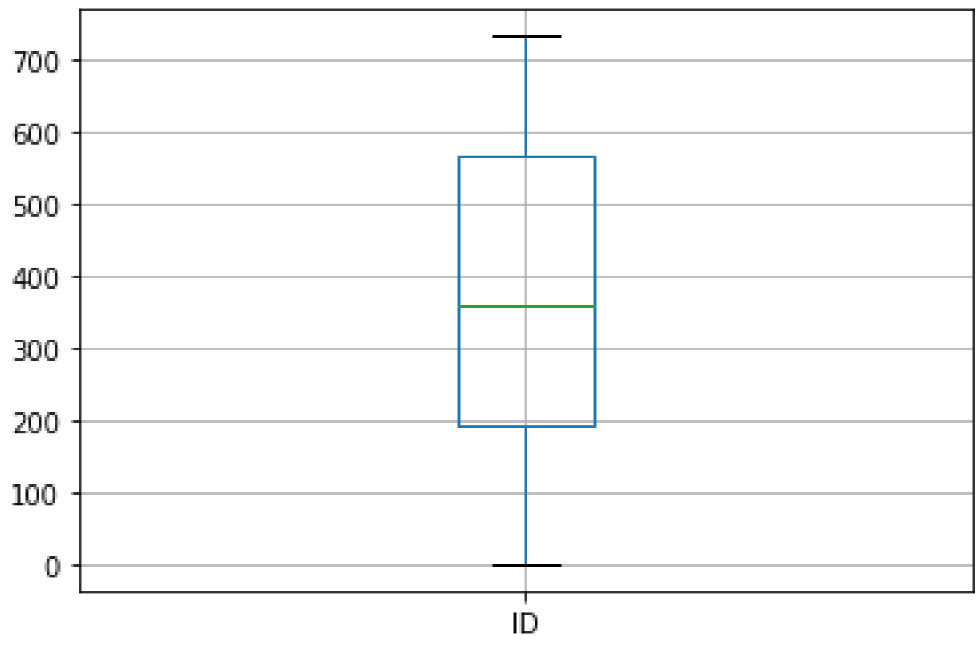
```
plt.show()
```

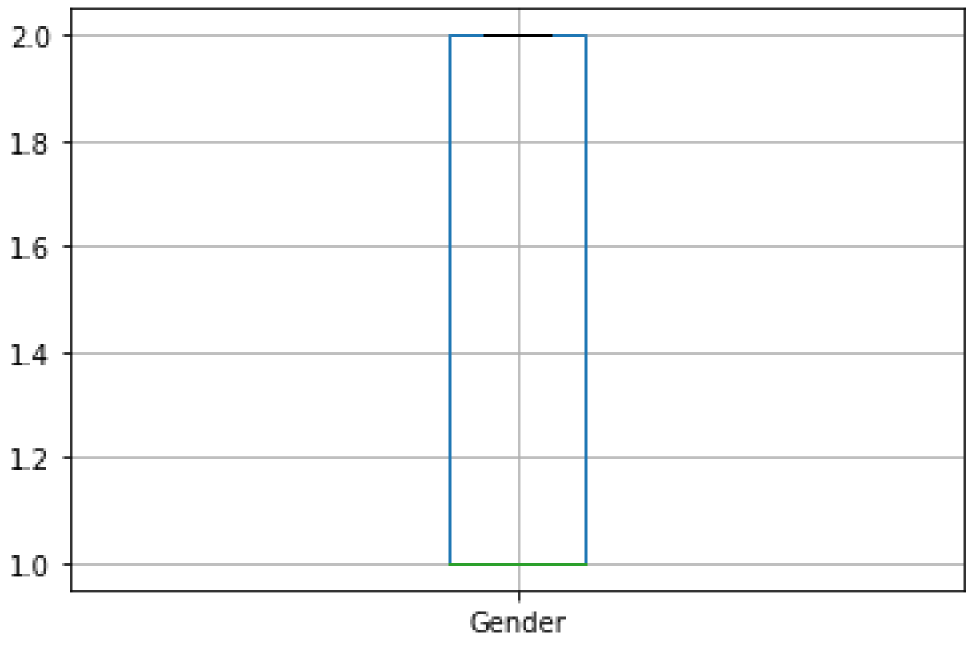


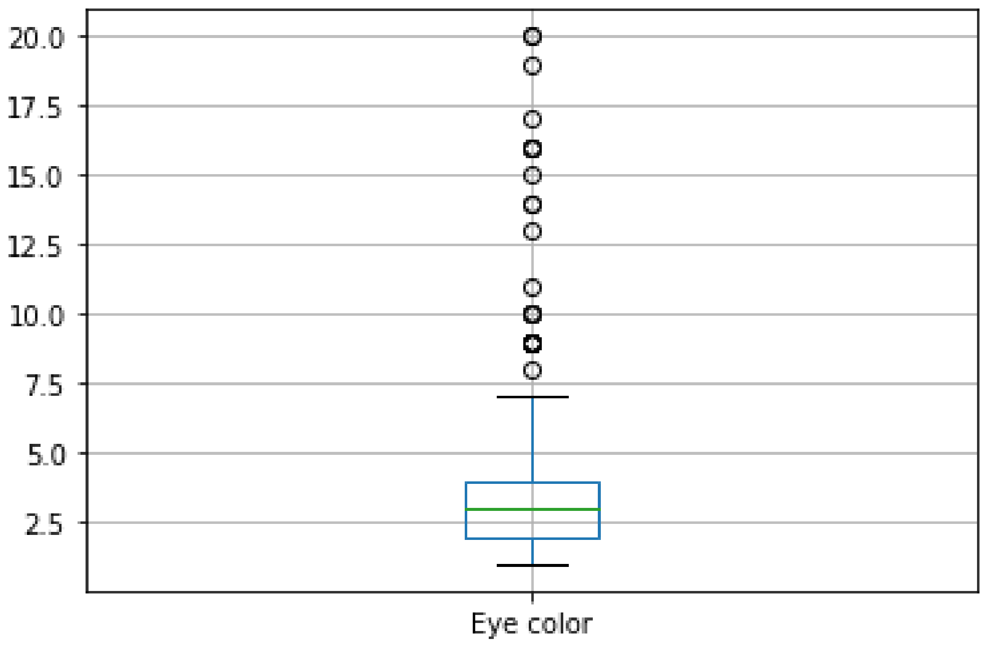
Learning from the first plot : Shows that the weights are mostly concentrated around 50-75. There's a nearly even distribution of weights among all the records. We can also see the values being capped at the upper and lower bounds which we did during outlier removal. **Alternative plot :** The alternative plot helps us infer the trend in the weight as we go down the records. Helps us infer the trends even in the highly dense areas like weight:50-75. **Learning from the alternative plot :** There are more overweight superheroes as we go down the data.

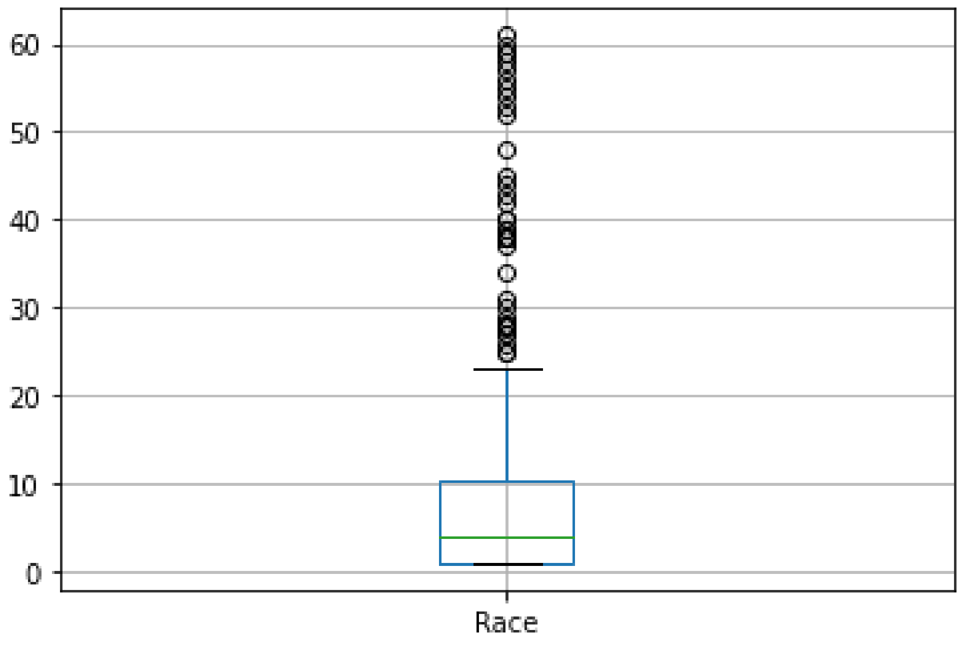
1.1.6 Box-plots

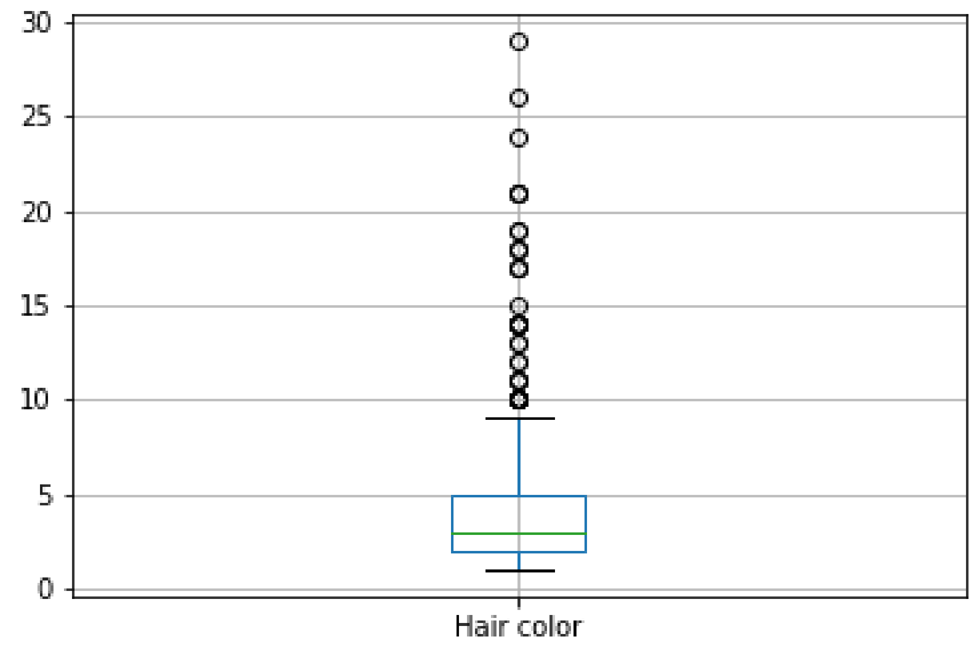
[46]:
```
# c = 1
# for (columnName, columnData) in df.iteritems():
#     plt.subplot(4, 2, c)
# #     plt.title('Weight Vs ID')
# #     plt.xlabel('ID')
# #     plt.ylabel('Weight')
#     boxplot = df.boxplot(column=columnName)
#     c = c + 1
for (columnName, columnData) in df.iteritems():
    if columnName != 'name':
        plt.figure()
        boxplot = df.boxplot(column=columnName)
```

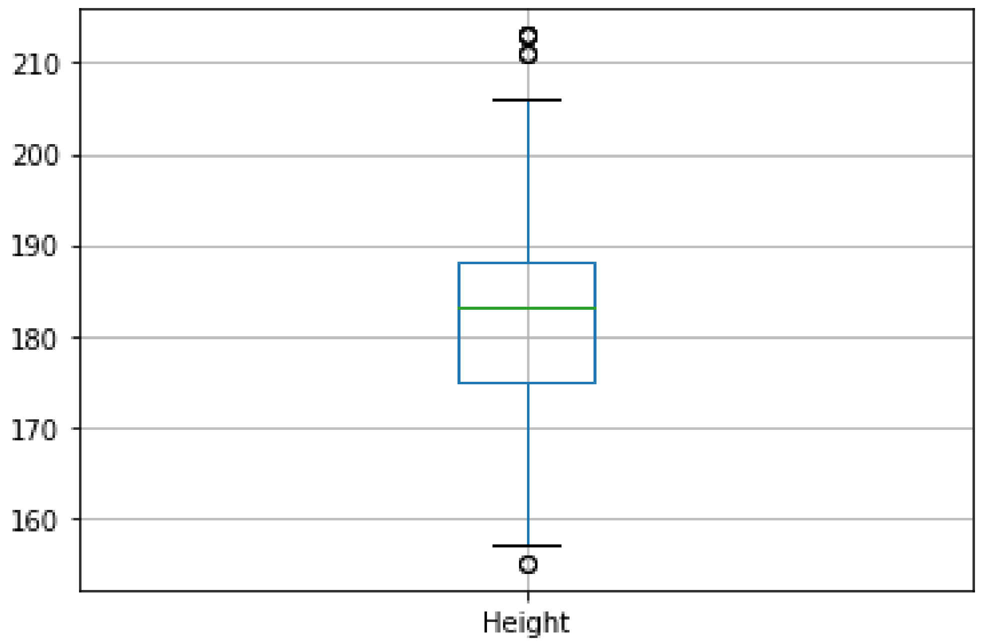

700
600
500
400
300
200
100
0
ID

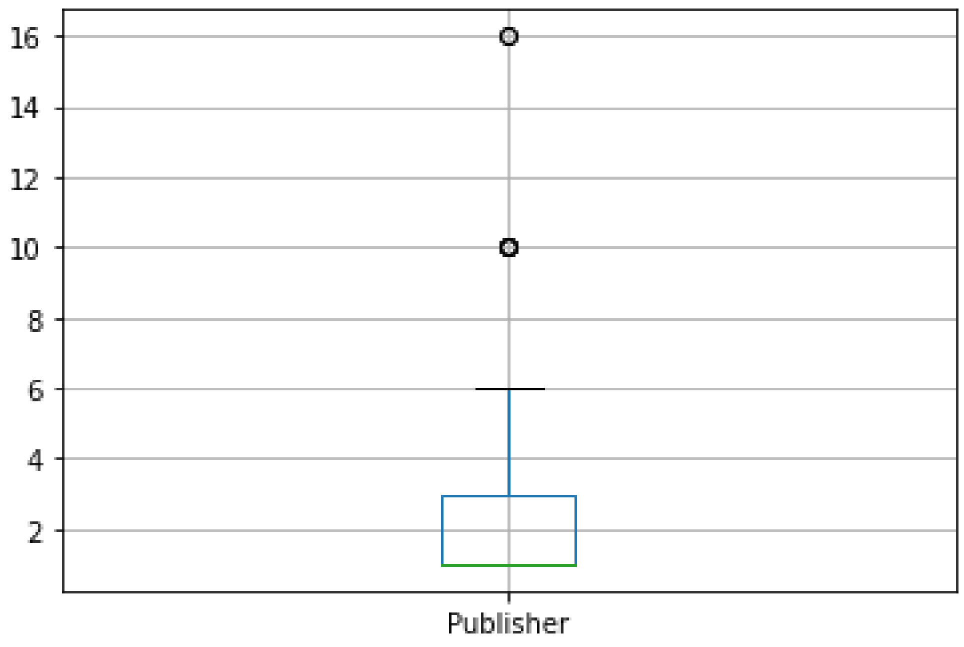

2.0
1.8
1.6
1.4
1.2
1.0
Gender

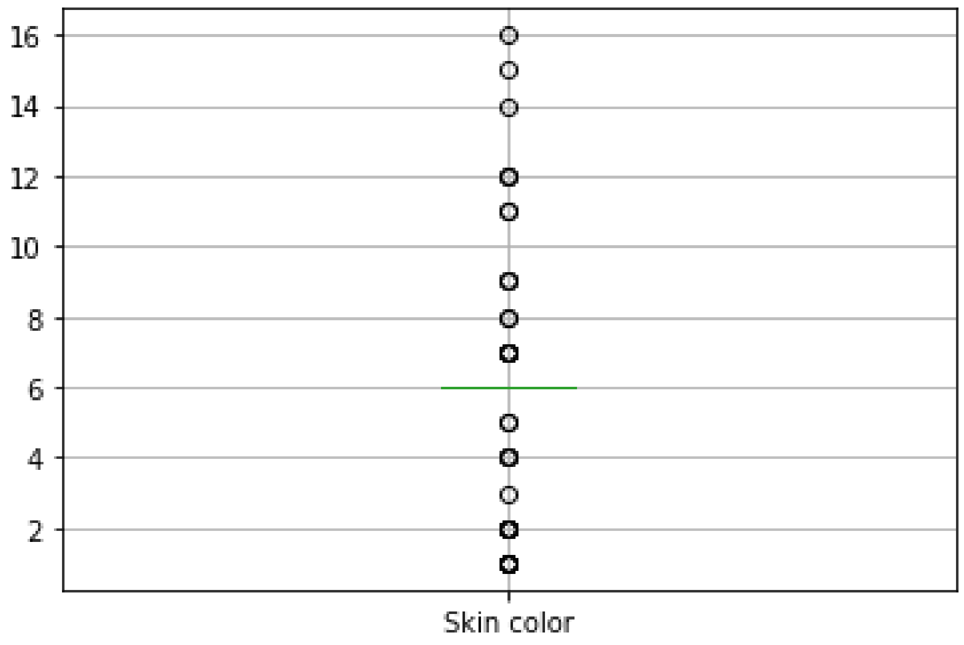

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Eye color

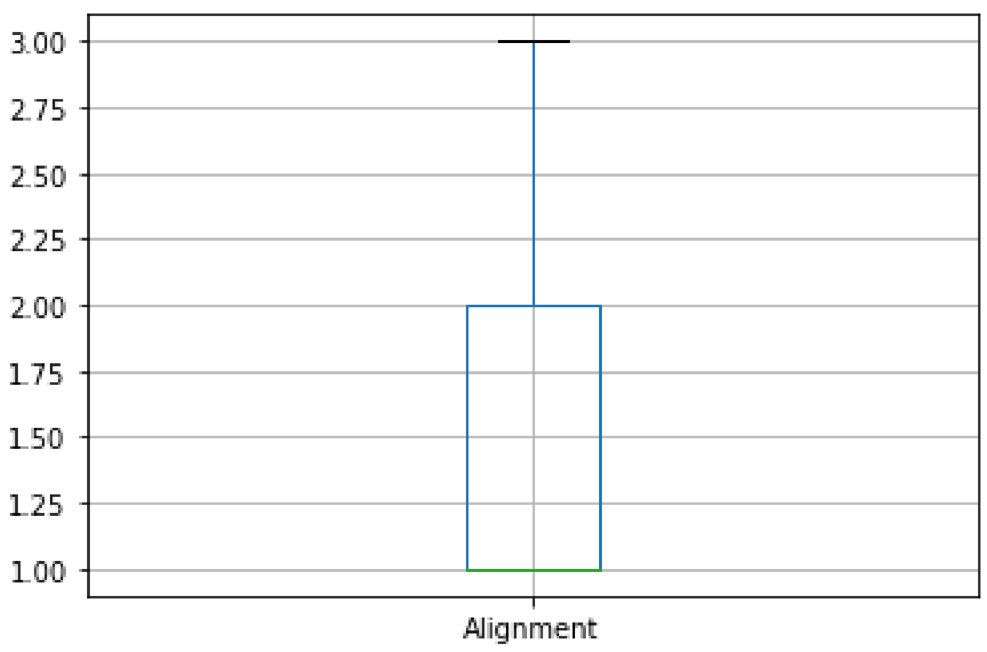

60
50
40
30
20
10
0
Race

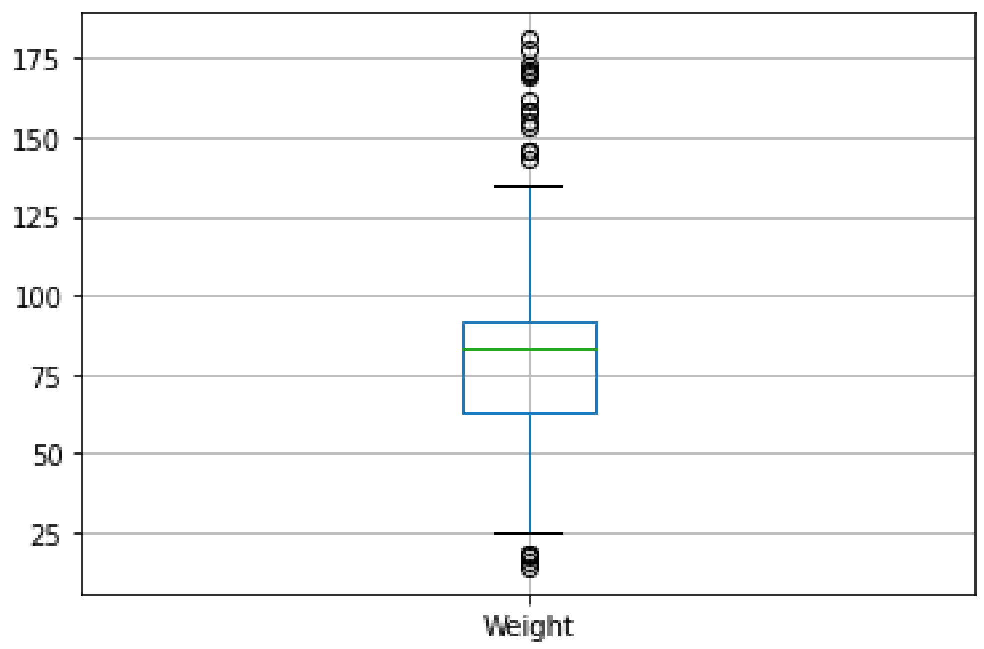

30
25
20
15
10
5
0
Hair color


210
200
190
180
170
160
Height


16
14
12
10
8
6
4
2
Publisher


16
14
12
10
8
6
4
2
Skin color





Boxplot for 'name' cannot be plotted since it is not numerical. Although the question asks for boxplot for each column, box plots for categorical data are not usually helpful in any means since they are not quantitative data. In our case the boxplots only for height and weight are useful.

1.2 EXPLORATORY DATA ANALYSIS

1.2.1 Describing the data

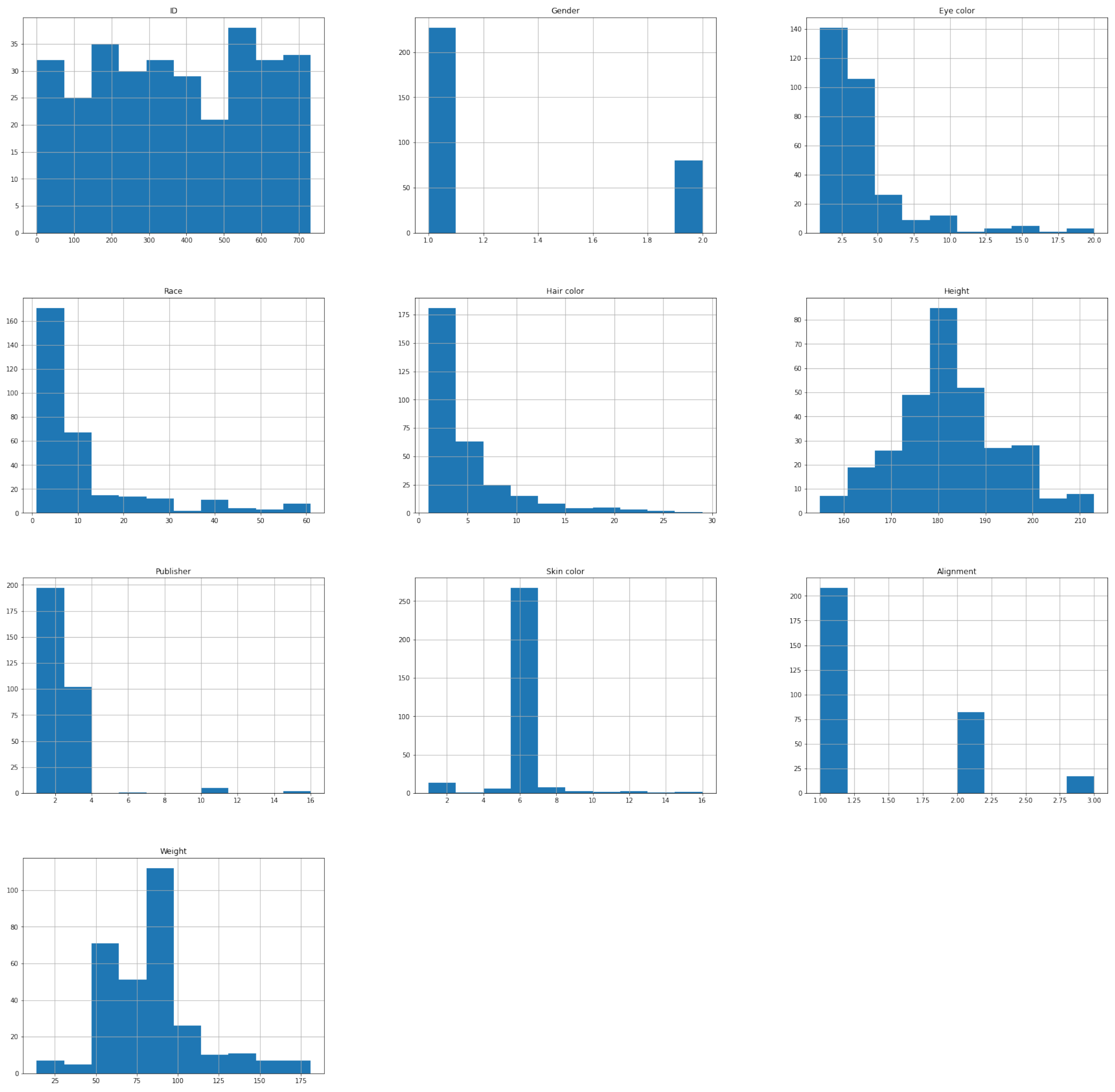
[47]:
```
df.describe()
```

[47]:
```
                ID      Gender   Eye color        Race  Hair color      Height  \
count   307.000000  307.000000  307.000000  307.000000  307.000000  307.000000
mean    370.123779    1.260586    3.801303    9.895765    4.667752  182.801303
std     215.397879    0.439671    3.151346   13.729966    4.691848   11.207910
min       0.000000    1.000000    1.000000    1.000000    1.000000  155.000000
25%     194.000000    1.000000    2.000000    1.000000    2.000000  175.000000
50%     360.000000    1.000000    3.000000    4.000000    3.000000  183.000000
75%     565.500000    2.000000    4.000000   10.500000    5.000000  188.000000
max     732.000000    2.000000   20.000000   61.000000   29.000000  213.000000

         Publisher  Skin color   Alignment      Weight
count   307.000000  307.000000  307.000000  307.000000
mean      1.954397    5.993485    1.377850   83.156352
std       1.823379    1.573875    0.589032   28.760136
min       1.000000    1.000000    1.000000   14.000000
25%       1.000000    6.000000    1.000000   63.000000
50%       1.000000    6.000000    1.000000   83.000000
75%       3.000000    6.000000    2.000000   92.000000
max      16.000000   16.000000    3.000000  181.000000
```

1.2.2 Visualizing Distribution with Histograms

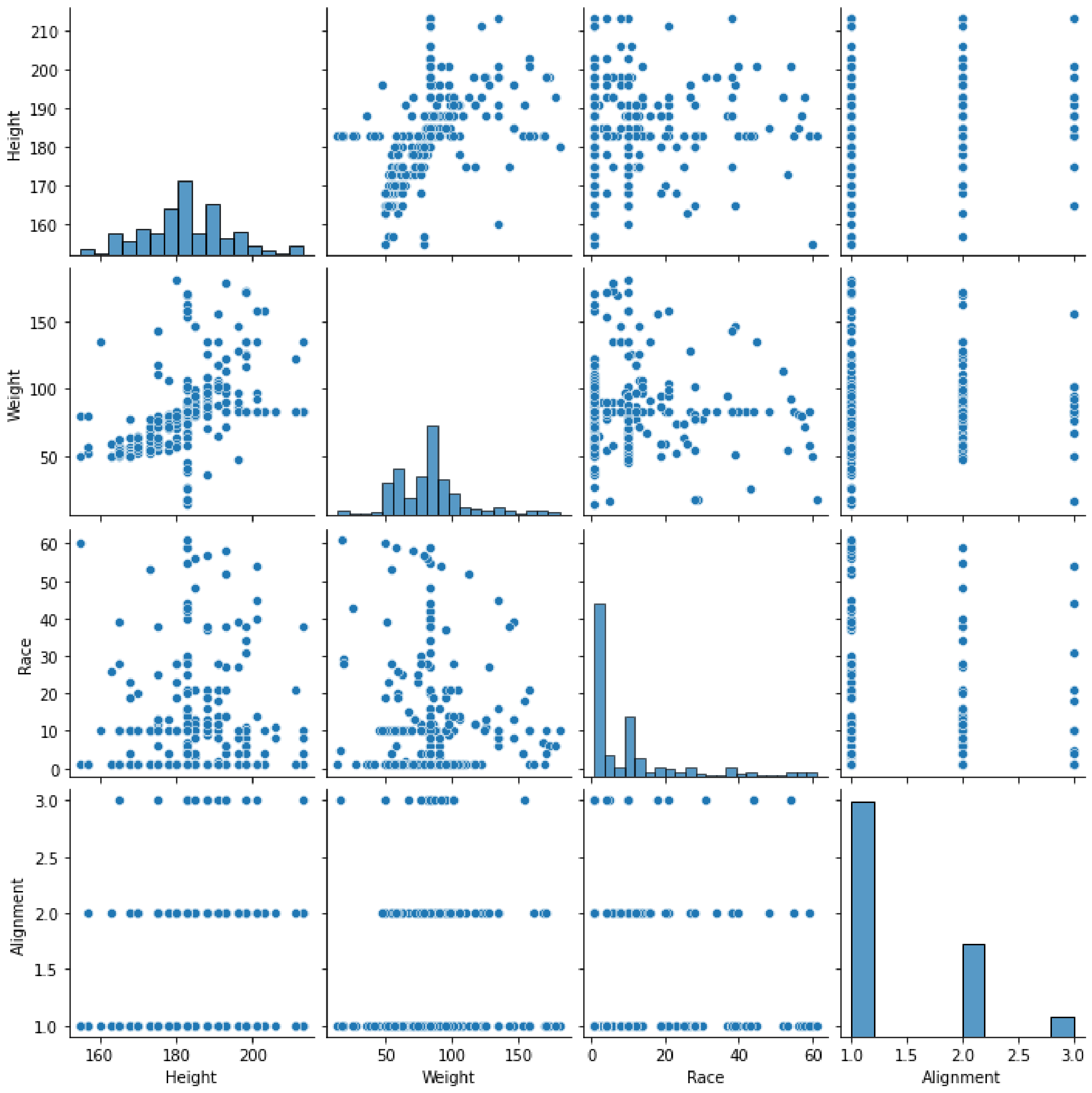
[48]:
```
his = df.hist(figsize=(30,30))
```



1.2.3 Scatter matrix for four columns : Height, Weight, Race, Alignment

```
[49]: import seaborn as sns
      sns.pairplot(df[["Height", "Weight", "Race", "Alignment"]])
```

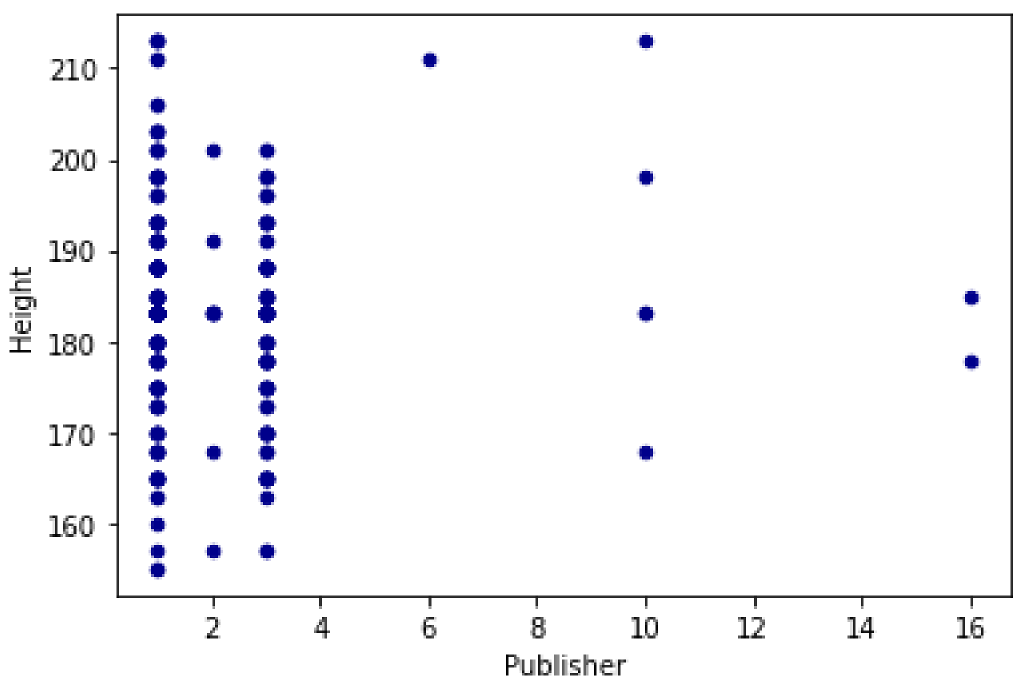
```
[49]: <seaborn.axisgrid.PairGrid at 0x7f940ac3b4c0>
```



1.2.4 Scatter plot for variables with no relationship

Columns chosen: Publisher and Height

```
[50]:  ax1 = df.plot.scatter(x='Publisher',y='Height',c='DarkBlue')
```



Covariance:

```
[51]: cov_unr = df[["Publisher","Height"]].cov()
      print(cov_unr)
```

```
           Publisher      Height
Publisher   3.324711   -0.247653
Height     -0.247653  125.617253
```

Correlation:

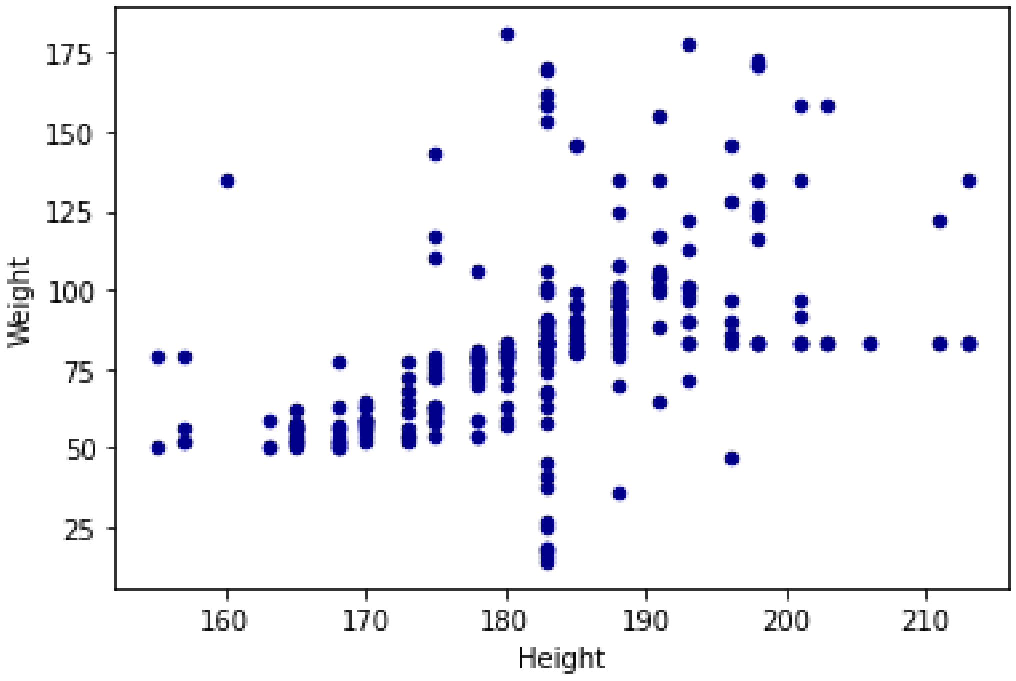
```
[52]: corr = df['Publisher'].corr(df['Height'])
      print(corr)
```

```
-0.012118298863661\18
```

1.2.5 Scatter plot for variables which seem to have a relationship

Columns chosen: Height and Weight

```
[53]: ax1 = df.plot.scatter(x='Height',y='Weight',c='DarkBlue')
```



Covariance:

```
[54]: cov_r = df[["Weight","Height"]].cov()
      print(cov_r)
```

```
            Weight      Height
Weight  827.145409  157.335090
Height  157.335090  125.617253
```

Correlation:

```
[55]: corr_r = df['Weight'].corr(df['Height'])
      print(corr_r)
```

```
0.4881013558742148
```

```
[ ]:
```