Data Cleaning in Pandas 1

Data cleaning is one of the fundamental tasks for Data professionals. Here, I have presented with some basic cleaning/wrangling operations. The dataset is sales data and contains information like ID, date of birth (dob), gender, city, last purchase date and income(in $ amount).

In [23]:
```
# Import necessary libraries
import pandas as pd
```

In [24]:
```
# check pandas version
print(pd.__version__)
```

```
2.0.3
```

Task:

1. Load the raw data file into a pandas DataFrame.
2. Check the data for missing values and duplicated records.
3. Remove any duplicate records from the data.
4. Fill in any missing values in the gender, marital_status, and city columns with the mode of the respective columns.
5. Create a new column named "age" that contains the age of each customer based on their date of birth.
6. Create a new column named "income_group" that categorizes customers into three groups based on their income amount- "Low", "Medium", "High" based on each 33% percentile.
7. Create another column "score_group" that categorizes customers into three groups based on their score -"Poor", "Fair", and "Good" based on each 33% percentile.
8. Keep only the records of 2018 and beyonds.
9. The ID column here is rather long.Shorten it, keep only the last ten characters.
10. Save the cleaned data to a new CSV file named sales_demo_final_data.csv.

My Approach:

To prepare the data based on the above instructions, I followed the below process. First, I created a function "dataprep", and inside this function, I make all the operations. So, the functio returns a cleaned dataset.

In [25]:
```
def dataprep(filename):

    # Read the data in a .csv file
    df = pd.read_csv(filename)

    # Remove duplicate rows
    df.drop_duplicates(inplace=True)

    # Keeping only the last ten characters of the ID
    df['id'] = df['id'].str[:10]

    # converting dob to datetime
    df['dob'] = pd.to_datetime(df['dob'])



    # Creating age column
    df['age'] = ((pd.to_datetime('today') - df['dob']).dt.days/365.2425).floordiv(1)



    # Find out the modes for gender, marital_status and city. Taking the first mode in case of multiple modes
    mode_city = df['city'].mode().values[0]
    mode_gender = df['gender'].mode().values[0]
    mode_marital_status = df['marital_status'].mode().values[0]

    #filling the missing values of the above columns by their modes

    df['city'].fillna(mode_city, inplace=True)
    df['gender'].fillna(mode_gender, inplace=True)
    df['marital_status'].fillna(mode_marital_status, inplace=True)

    # Creating income groups using qcut to divide the column into three groups
    df['income_group'] = pd.qcut(df['income'], q=[0, 0.33, 0.67, 1], labels=['Low', 'Medium', 'High'])

    # Creating score groups
    df['score_group'] = pd.qcut(df['score'], q = [0, 0.33, 0.67, 1], labels=['Poor', 'Fair', 'Good'])

    # converting the last_purchase_date to a date column
    df['last_purchase_date'] = pd.to_datetime(df['last_purchase_date'], format='mixed')

    # Masking for the purchase date 2018 and beyond
    mask_purchase_date = df['last_purchase_date'].dt.year > 2017

    # filtering the data according to the above mask
    df = df[mask_purchase_date]


    return df
```

In [26]:
```
df = dataprep('raw_data.csv')
df.head()
```

Out[26]:

	id	gender	dob	income	marital_status	city	last_purchase_date	score	age	income_group	score_group
0	660ba2ad-e	Male	1977-05-19	46532	Single	Oklahoma City	2020-06-22	63	46.0	Medium	Poor
1	6da25e92-a	Female	2000-11-28	13734	Single	Columbus	2020-08-04	43	22.0	Low	Poor
2	11261389-d	Female	1975-06-05	36282	Single	Miami	2022-08-21	882	48.0	Medium	Good
3	c9f67b5a-f	Male	1992-05-27	83451	Divorced	Colorado Springs	2020-12-18	653	31.0	High	Fair
4	d3024d83-c	Female	1987-09-13	58351	Divorced	New Orleans	2021-05-25	535	36.0	Medium	Fair

In [27]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 38573 entries, 0 to 984525
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  38573 non-null  object
 1   gender              38573 non-null  object
 2   dob                 38573 non-null  datetime64[ns]
 3   income              38573 non-null  int64
 4   marital_status      38573 non-null  object
 5   city                38573 non-null  object
 6   last_purchase_date  38573 non-null  datetime64[ns]
 7   score               38573 non-null  int64
 8   age                 38573 non-null  float64
 9   income_group        38573 non-null  category
 10  score_group         38573 non-null  category
dtypes: category(2), datetime64[ns](2), float64(1), int64(2), object(4)
memory usage: 3.0+ MB
None
```

In [17]:
```
# Saving the cleaned data as a .csv file
df.to_csv('sales_demo_final_data.csv', index=False)
```

Data Cleaning in Pandas 2

In this dataset, I have analyzed some ODI Cricket data. The data was obtained from cricinfo.

Task

1. How many rows and columns are present in this dataset?
2. Are there any missing values present in this dataset? If so, in which columns?
3. What are the data types in this dataset?
4. Rename the column names accordingly: 1. 'Mat':'Match', 2. 'Inns':'Innings', 3. 'NO': 'NotOut', 4. 'HS': 'Highest_score', 5. 'Ave': 'Average', 6. 'BF': 'Balls_Faced', 7. 'SR': 'Strike_Rate'.
5. Remove the columns: 'BF', 0, 4s, and 6s.
6. Show the top 10 batsmen with the highest batting average. If players have the same average, reorder them according to the highest number of centuries. Is there any Bangladeshi player present in the Top 10?
7. Which player(s) had played for the longest and the shortest period of time in this dataset?
8. Based on the country column, how many players played for "Asia XI"?
9. Save the cleaned file in a csv file named "batsmen".

My Approach

To answer the above questions, I started the proceeding by creating a function. All the wrangling were done inside the function. Later I answered all the questions.

In [54]:
```
# importing necessary libraries
import pandas as pd
```

In [55]:
```
# Import Data
def data_prep(filename):

    #Read data into a csv file
    df = pd.read_csv(filename)

    # Renaming the columns
    df = (df.rename(columns={"Mat": "Match", "NO": "NotOut","HS": "Highest_score","Ave": "Average",
                             "Ave": "Average","SR": "Strike_Rate"}))

    # splitting the span column to claculate years played
    df[['Start_career', 'End_career']] = df['Span'].str.split("-", expand=True).astype(int)
    df['Years_active'] = df['End_career'] - df['Start_career']

    # Creating columns for Name and Teams/Country represented

    df[['Player', 'Country']] = df['Player'].str.split("(", expand=True)
    df['Country'] = df["Country"].str.replace(")","", regex=False)
    df[["Team 1", "Team 2", "Team 3"]] = df['Country'].str.split("/", expand=True).fillna("None")

    # dropping the columns
    df.drop(columns = ['0', '4s', '6s', 'Span', 'Country', 'BF'], inplace=True)



    return df
```

In [56]:
```
# seeing the file
df = data_prep('batsman.csv')
df.head()
```

Out[56]:

	Player	Match	Inns	NotOut	Runs	Highest_score	Average	Strike_Rate	100	50	Start_career	End_career	Years_active	Team 1	Team 2	Team 3
0	SR Tendulkar	463	452	41	18426	200*	44.83	86.23	49	96	1989	2012	23	INDIA	None	None
1	KC Sangakkara	404	380	41	14234	169	41.98	78.86	25	93	2000	2015	15	Asia	ICC	SL
2	RT Ponting	375	365	39	13704	164	42.03	80.39	30	82	1995	2012	17	AUS	ICC	None
3	ST Jayasuriya	445	433	18	13430	189	32.36	91.20	28	68	1989	2011	22	Asia	SL	None
4	DPMD Jayawardene	448	418	39	12650	144	33.37	78.96	19	77	1998	2015	17	Asia	SL	None

In [57]:
```
# Checking the basic information of the data.
print(df.info(verbose=True, show_counts=True))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Player         92 non-null     object
 1   Match          92 non-null     int64
 2   Inns           92 non-null     int64
 3   NotOut         92 non-null     int64
 4   Runs           92 non-null     int64
 5   Highest_score  92 non-null     object
 6   Average        92 non-null     float64
 7   Strike_Rate    92 non-null     float64
 8   100            92 non-null     int64
 9   50             92 non-null     int64
 10  Start_career   92 non-null     int32
 11  End_career     92 non-null     int32
 12  Years_active   92 non-null     int32
 13  Team 1         92 non-null     object
 14  Team 2         92 non-null     object
 15  Team 3         92 non-null     object
dtypes: float64(2), int32(3), int64(6), object(5)
memory usage: 10.5+ KB
None
```

With a few newly created rows, there are 16 columns and 92 rows. None of the rows contain any missing values. The types of data across the datasets are string, float and integer.

In [58]:
```
# The command adds up the null values in each column.
df.isnull().sum()
```

Out[58]:
```
Player           0
Match            0
Inns             0
NotOut           0
Runs             0
Highest_score    0
Average          0
Strike_Rate      0
100              0
50               0
Start_career     0
End_career       0
Years_active     0
Team 1           0
Team 2           0
Team 3           0
dtype: int64
```

In [59]:
```
df[['Player', 'Years_active']].sort_values(by='Years_active', ascending=False).head(1)
```

Out[59]:

	Player	Years_active
0	SR Tendulkar	23

Sachin Played for the longest Period. 23 Years.

In [60]:
```
df[['Player', 'Years_active']].sort_values(by='Years_active', ascending=False).tail(1)
```

Out[60]:

	Player	Years_active
82	AJ Finch	7

AJ Finch has the shortest career, only 7 years long.

In [61]:
```
df[['Player', 'Team 1', 'Team 2','Team 3', 'Average']].sort_values(by='Average', ascending=False).head(10)
```

Out[61]:

	Player	Team 1	Team 2	Team 3	Average
5	V Kohli	INDIA	None	None	58.07
45	MG Bevan	AUS	None	None	53.58
16	AB de Villiers	Afr	SA	None	53.50
60	JE Root	ENG	None	None	51.33
10	MS Dhoni	Asia	INDIA	None	50.57
28	HM Amla	SA	None	None	49.46
19	RG Sharma	INDIA	None	None	48.60
24	LRPL Taylor	NZ	None	None	48.20
77	MEK Hussey	AUS	None	None	48.15
58	KS Williamson	NZ	None	None	47.48

Here are the top 10 batsmen with highest Average. No, Bangladeshi Batsman in the top 10.

Let's do following to figure out how many players have played for Asia XI.

In [62]:
```
(df['Team 1'] == "Asia").value_counts()
```

Out[62]:
```
Team 1
False    79
True     13
Name: count, dtype: int64
```

In [63]:
```
(df['Team 2'] == "Asia").value_counts()
```

Out[63]:
```
Team 2
False    92
Name: count, dtype: int64
```

In [64]:
```
(df['Team 3'] == "Asia").value_counts()
```

Out[64]:
```
Team 3
False    92
Name: count, dtype: int64
```

So, only 13 players have played for Asia XI.

In [65]:
```
# Saving the new file in csv mode as 'batsmen'
df.to_csv('batsmen.csv', index=False)
```