Software Engineering For Data Science (SEDS)

Class: 2nd Year 2nd Cycle
Branch: AIDS

Dr. Belkacem KHALDI| ESI-SBA

Lecture 05:

Data Processing & Cleaning for Data Science: Data Ingestion and Wrangling with Pandas

Data Processing & Cleaning for Data Science

Part I: Data Ingestion and Wrangling with Pandas

1. Basics Understanding of Pandas
2. Loading Data From External Resources
3. Basic Exploratory Data Analysis
4. Basic Data Cleaning Operations

Data Ingestion: the process, of shifting data, from a variety of sources, into the Pandas DataFrame structure

Data wrangling: the process of cleaning, structuring and enriching raw data into a desired format for better decision making in less time.

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Basics Understanding of Pandas

What is Pandas?

- Pandas ➔ A modern, powerful and feature-rich library designed for doing data analysis in Python.
- Combines functionalities from:
  - NumPy , Matplotlib, and Scipy
- Allows manipulating data from numerous different data formats directly:

Conceptual Model

- Built on a Two **Data Structures**:
  - **DataFrame:** (a 2-dimensional data structure) used for storing and mainpulating table-like data (data with rows and columns).
  - **Series** (a 1-dimensional data structure) used for storing and manipulating a sequence of values.

Column Index (df.columns)

Row Index (df.indexe)

	Track	Composer	UnitPrice	Genre	Album	Artist
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
1	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
2	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
3	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
4	Snowballed	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	None	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy

Pandas Series

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames Creation: From Series

```
import pandas as pd

Tracks = pd.Series({1:'Aicha',
                    2:‘Respect',
                    3:'Courage'})

Genre = pd.Series({1:'Rai',
                   2:'Soul',
                   3:'Pop'})
```

```
df = pd.DataFrame({'Track':Tracks,
                   'Genre':Genres})
```

Or

```
df = pd.DataFrame({'Track':Tracks,
                   'Genre':Genres},
                  axis=1)
```

```
Tracks
```

```
1      Aicha
2    Respect
3    Courage
dtype: object
```

```
Genres
```

```
1     Rai
2    Soul
3     Pop
dtype: object
```

	Track	Genre
1	Aicha	Rai
2	Respect	Soul
3	Courage	Pop

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames Creation: From Python Dictionary

```
import pandas as pd

# default ---  data by columns
df = pd.DataFrame({
 'Track' : ['Aicha', 'Respect', 'Courage'],
 'Genre' : ['Rai', 'Soul', 'Pop']
                  })
```

	Track	Genre
0	Aicha	Rai
1	Respect	Soul
2	Courage	Pop

```
import pandas as pd

# ---- data by rows
df = pd.DataFrame.from_dict({
 10 : {'Track':'Aicha', 'Genre':'Rai'},
 11 : {'Track':'Respect', 'Genre':'Soul'},
 12 : {'Track':'Courage', 'Genre':'Pop'}
                       }, orient='index')
```

	Track	Genre
10	Aicha	Rai
11	Respect	Soul
12	Courage	Pop

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames: Working with Columns

Get column index and labels

```
idx = df.columns # get col indexes
label = df.columns[0] # first col label
l = df.columns.tolist() # list of col labels
a = df.columns.values # array of col labels
```

```
idx
```

```
Index(['Track', 'Genre'], dtype='object')
```

```
label
```

```
'Track'
```

```
l
```

```
['Track', 'Genre']
```

```
a
```

```
array(['Track', 'Genre'], dtype=object)
```

Adding new columns to a DataFrame

```
df['Bytes'] = np.repeat(np.nan,len(df))
df['UnitPrice'] = np.random.rand(len(df))
df['Composer'] = df.index
df['Artist'] = np.array(['Khaled','Aretha','Celine'])
```

	Track	Genre	Bytes	UnitPrice	Composer	Artist
0	Aicha	Rai	NaN	0.053272	0	Khaled
1	Respect	Soul	NaN	0.992519	1	Aretha
2	Courage	Pop	NaN	0.584433	2	Celine

Original DataFrame Added Columns

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames: Working with Columns

Changing Column labels

```
df = df.rename(columns={'Artist':'Singer'})
```

	Track	Genre	Bytes	UnitPrice	Composer	Singer
0	Aicha	Rai	NaN	0.053272	0	Khaled
1	Respect	Soul	NaN	0.992519	1	Aretha
2	Courage	Pop	NaN	0.584433	2	Celine

Dropping Columns

```
df = df.drop('Bytes', axis=1)
s = df.pop('Bytes') # or drops from frame
df = df.drop(df.columns[2], axis=1) # or 3rd col
```

	Track	Genre	UnitPrice	Composer	Artist
0	Aicha	Rai	0.838593	0	Khaled
1	Respect	Soul	0.859112	1	Aretha
2	Courage	Pop	0.530924	2	Celine

Selecting Columns

```
s1 = df['Artist'] # select col to Series
df1 = df[['Genre']] # select col to df
df2 = df[['Track','Artist']] # select 2-plus cols
s2 = df[df.columns[0]] # select by number
df3 = df[df.columns[[0, 2, 4]]] # by numbers
```

s1

```
0    Khaled
1    Aretha
2    Celine
Name: Artist, dtype: object
```

df1

	Genre
0	Rai
1	Soul
2	Pop

df2

	Track	Artist
0	Aicha	Khaled
1	Respect	Aretha
2	Courage	Celine

df3

	Track	UnitPrice	Artist
0	Aicha	0.838593	Khaled
1	Respect	0.859112	Aretha
2	Courage	0.530924	Celine

s2

```
0      Aicha
1    Respect
2    Courage
Name: Track, dtype: object
```

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames: Working with Columns

Changing Column Values Based on Criteria

```
df['UP_']= df['UnitPrice'].where(df['UnitPrice']>0.5)
df['Bytes_'] = df['Bytes'].where(df.Bytes == np.NaN,
                                 other=1.5)
```

	Track	Genre	Bytes	UnitPrice	Composer	Artist	UP_	Bytes_
0	Aicha	Rai	NaN	0.231963	0	Khaled	NaN	1.5
1	Respect	Soul	NaN	0.931151	1	Aretha	0.931151	1.5
2	Courage	Pop	NaN	0.742499	2	Celine	0.742499	1.5

Common Column-wide Methods/Attributes

```
dt = df['UP_'].dtype # type of data
sz = df['UP_'].size # col dimensions
cnt = df['UP_'].count() # non-NA count
sm = df['UP_'].sum()
prd = df['UP_'].prod()
mn = df['UP_'].min()
mx = df['UP_'].max()
mdn = df['UP_'].median() # also mean()
cv = df['UP_'].cov(df['Composer'])
```

dt	sz	cnt	sm	prd
dtype('float64')	3	2	1.6736492121877766	0.6913780108932741

mdn	mn	mx
0.8368246060938883	0.7424985202484697	0.9311506919393069

cv
-0.09432608584541857

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames: Working with Rows

Getting the row index and labels

```
idx = df.index # get row index
frst_label = df.index[0] # first row label
lst_label = df.index[-1] # last row label
l = df.index.tolist() # get as a list
a = df.index.values # get as an array
```

idx	frst_label	lst_label	l
RangeIndex(start=0, stop=3, step=1)	0	2	[0, 1, 2]

a
array([0, 1, 2], dtype=int64)

Adding Rows

```
new_row = {'Track':'It is Now or Never',
           'Genre':'Rock',
           'UnitPrice':0.65,
           'Composer':3,
           'Artist':'Elvis'}
df = df.append(new_row, ignore_index=True)
```

	Track	Genre	Bytes	UnitPrice	Composer	Artist
0	Aicha	Rai	NaN	0.040820	1	Khaled
1	Respect	Soul	NaN	0.607273	2	Aretha
2	Courage	Pop	NaN	0.774541	3	Celine
3	It is Now or Never	Rock	NaN	0.650000	3	Elvis

Changing the Row Index

```
df = df.set_index(keys=['Track'])
```

Track	Genre	Bytes	UnitPrice	Composer	Artist
Aicha	Rai	NaN	0.040820	1	Khaled
Respect	Soul	NaN	0.607273	2	Aretha
Courage	Pop	NaN	0.774541	3	Celine
It is Now or Never	Rock	NaN	0.650000	3	Elvis

Basics Understanding of Pandas

DataFrames Basic Practices

DataFrames: Working with Rows

Dropping rows (by name)

```
df = df.drop('It is Now or Never')
```

Track	Genre	Bytes	UnitPrice	Composer	Artist
Aicha	Rai	NaN	0.040820	1	Khaled
Respect	Soul	NaN	0.607273	2	Aretha
Courage	Pop	NaN	0.774541	3	Celine

Sorting by Row index

```
df.sort_index(inplace=True)
```

Track	Genre	Bytes	UnitPrice	Composer	Artist
Aicha	Rai	NaN	0.040820	1	Khaled
Courage	Pop	NaN	0.774541	3	Celine
Respect	Soul	NaN	0.607273	2	Aretha

Select a slice of rows by integer position

```
df = df[:] # copy entire DataFrame
df1 = df[0:2] # rows 0 and 1
df2 = df[2:3] # row 2 (the third row)
df3 = df[-1:] # the last row
df4 = df[:-1] # all but the last row
df5 = df[::2] # every 2nd row (0 2 ..)
```

Selecting data using loc() function

```
# rows with Genre = Rai
df1 = df.loc[(df.Genre =='Rai')]
# rows with Genre in List[‘Rai’,’Soul’]
df2 = df.loc[(df.Genre.isin(['Rai','Soul']))]
# rows with Genre = ‘Pop’ or Artist=‘Aretha’
df3 = df.loc[(df.Genre == 'Pop') |
             (df.Artist == 'Aretha')
            ]
```

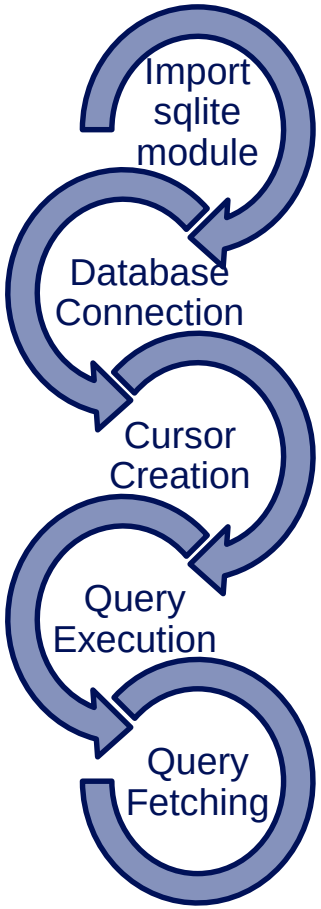
Loading Data From External Resources

Working with SQLite Databases

What is SQLite?

- **SQLite**– a lightweight database version
  - Lacks the richer functionalities of other SQL databases such as **Oracle**, but:
    - Faster and easier to use.
    - Can still hold a lot of data, with a maximum potential database size of around **281 TB.**
- You can interact with **SQLite** database, since **SQLite3** comes installed with Python.

Steps To Interact with SQLite :




```
import sqlite3
```

```
connection = sqlite3.connect('<db_name>.db')
```

```
cursor = connection.cursor()
```

```
cursor.execute('<sql statement>')
```

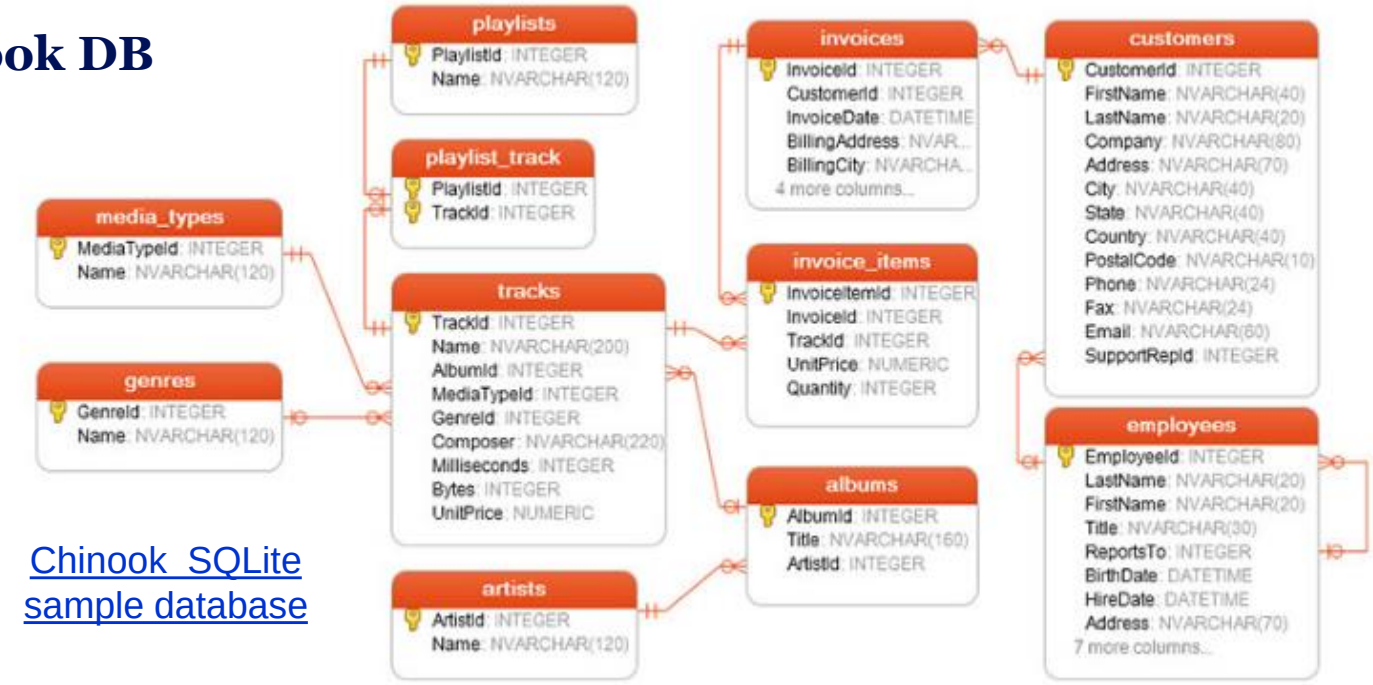
```
cursor.fetchall()
```

Loading Data From External Resources

Working with SQLite Databases

SQLite Hands-On Experience with the Chinook DB

- **Chinook** database ➔ A dataset of songs and purchases from customers, similar to a the **iTunes** songs and purchases dataset.
- The **Chinook** database is available online as a zip file that contains the **chinook.db** file.
- **Interacting with sqlite3 in terminal:**
  - Open a terminal or a shell.
  - Navigate to the folder cotaining the **chinook.db** file and tape:



Chinook_SQLite sample database

```
$ sqlite3 chinook.db
```

```
(base) C:\Users\user\Downloads\chinook-db>sqlite3 chinook.db
SQLite version 3.38.2 2022-03-26 13:51:10
Enter ".help" for usage hints.
sqlite>
```

```
sqlite> .tables
```

```
sqlite> .tables
albums          employees       invoices        playlists
artists         genres          media_types     tracks
customers       invoice_items   playlist_track
```

Loading Data From External Resources

Working with SQLite Databases

SQLite Hands-On Experience with the Chinook DB

1. Database Connection & Cursor Creation

```
import sqlite3
# Using relative path
connection = sqlite3.connect('chinook.db')
cursor = connection.cursor()
```

r ➔ raw string: means it treats special characters (such as the backslash, \)

```
# Using absolute path
connection = sqlite3.connect(r'C:\Users\user\Downloads\chinook-db\chinook.db')
```

2. Query Execution & Fetching

```
cursor.execute('SELECT * FROM artists LIMIT 5;')
cursor.fetchall()
```

```
[(1, 'AC/DC'),
 (2, 'Accept'),
 (3, 'Aerosmith'),
 (4, 'Alanis Morissette'),
 (5, 'Alice In Chains')]
```

```
# For Bigger SQL Queries
query = """
        SELECT *
        FROM artists
        LIMIT 5;
        """
cursor.execute(query)
cursor.fetchall()
```

Loading Data From External Resources

Working with SQLite Databases

SQLite Hands-On Experience with the Chinook DB

3. Fetching Queries into Pandas DataFrame

```
query = """
SELECT
        t.name as Track,
        t.composer,
        t.unitprice,
        g.name as Genre,
        a.title as Album,
        r.name as Artist
FROM tracks t
JOIN genres g ON t.genreid = g.genreid
JOIN albums a ON t.albumid = a.albumid
JOIN artists r ON a.artistid = r.artistid;
"""

sql_df = pd.read_sql_query(query, connection)
```

	Track	Composer	UnitPrice	Genre	Album	Artist
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
1	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
2	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
3	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
4	Snowballed	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	None	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy

Loading Data From External Resources

Working with CSV & Excel Files

```
csv_df = pd.read_csv('itunes_data.csv', sep=',', decimal='.')
```

	Track	Composer	UnitPrice	Genre	Album	Artist
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
1	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
2	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
3	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
4	Snowballed	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC
...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	None	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy

Loading Data From External Resources

Working with CSV & Excel Files

```
excel_df = pd.read_excel('data/itunes_data.xlsx')
```

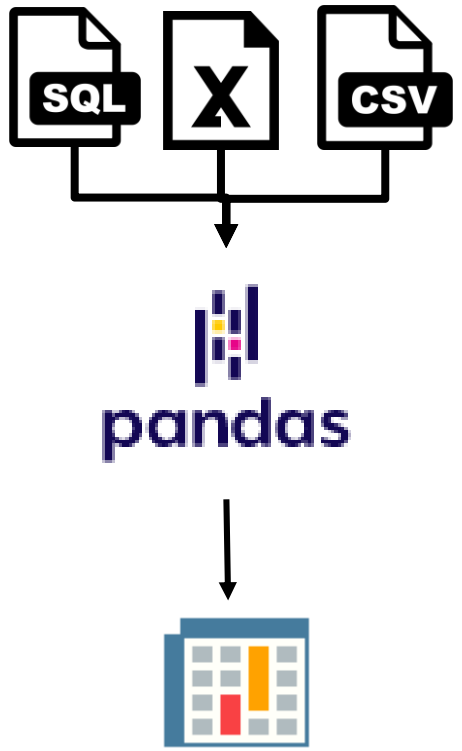
	Track	Composer	Milliseconds	Bytes	UnitPrice	Genre	Album	Artist
0	My Time After Awhile	Robert Geddins/Ron Badger/Sheldon Feinberg	182491	6022698	0.99	Blues	The Best Of Buddy Guy - The Millenium Collection	Buddy Guy
1	Be Quick Or Be Dead	Bruce Dickinson/Janick Gers	204512	8181888	0.99	Rock	Fear Of The Dark	Iron Maiden
2	Água E Fogo	Chico Amaral/Edgard Scandurra/Samuel Rosa	278987	9272272	0.99	Rock	Maquinarama	Skank
3	Ozone Baby	Jimmy Page, Robert Plant	215954	7079588	0.99	Rock	Coda	Led Zeppelin
4	Bop Boogie	NaN	189596	6093124	0.99	Jazz	Up An' Atom	Gene Krupa
...	...	...	...	...	...	...	...	...
210	Black Dog	John Paul Jones/Robert Plant	317622	10267572	0.99	Rock	BBC Sessions [Disc 2] [Live]	Led Zeppelin

Basic Exploratory Data Analysis (EDA)

Getting All Data into One DataFrame

```
itunes_df = pd.concat([csv_df, excel_df, sql_df])
```

	Track	Composer	Milliseconds	Bytes	UnitPrice	Genre	Album	Artist
0	All the Best Cowboys Have Daddy Issues	NaN	2555492.0	211743651.0	1.99	TV Shows	Lost, Season 1	Lost
1	Beira Mar	Gilberto Gil	295444.0	9597994.0	0.99	Latin	Unplugged	Eric Clapton
2	Brasil	Milton Nascimento, Fernando Brant	155428.0	5252560.0	0.99	Latin	Milton Nascimento Ao Vivo	Milton Nascimento
3	Ben Franklin	NaN	1271938.0	264168080.0	1.99	Comedy	The Office, Season 3	The Office
4	O Último Romântico (Ao Vivo)	NaN	231993.0	7692697.0	0.99	Latin	Lulu Santos - RCA 100 Anos De Música - Álbum 02	Lulu Santos
...	...	...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	None	NaN	NaN	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy
3499	String Quartet No. 12 in C Minor, D. 703 "Quar...	Franz Schubert	NaN	NaN	0.99	Classical	Schubert: The Late String Quartets & String Qu...	Emerson String Quartet
3500	L'orfeo, Act 3, Sinfonia (Orchestra)	Claudio Monteverdi	NaN	NaN	0.99	Classical	Monteverdi: L'Orfeo	C. Monteverdi, Nigel Rogers - Chiaroscuro; Lon...
3501	Quintet for Horn, Violin, 2 Violas, and Cello ...	Wolfgang Amadeus Mozart	NaN	NaN	0.99	Classical	Mozart: Chamber Music	Nash Ensemble
3502	Koyaanisqatsi	Philip Glass	NaN	NaN	0.99	Soundtrack	Koyaanisqatsi (Soundtrack from the Motion Pict...	Philip Glass Ensemble



Basic Exploratory Data Analysis (EDA)

General EDA Checklist

- **Examine the top and bottom of the data**
- Examine the data's dimensions
- Examine the datatypes and missing values
- Investigate statistical properties of the data
- Create plots of the data

This EDA can provide a starting point for further analysis

```
itunes_df.head()
```

	Track	Composer	UnitPrice	Genre	Album	Artist
0	All the Best Cowboys Have Daddy Issues	NaN	1.99	TV Shows	Lost, Season 1	Lost
1	Beira Mar	Gilberto Gil	0.99	Latin	Unplugged	Eric Clapton
2	Brasil	Milton Nascimento, Fernando Brant	0.99	Latin	Milton Nascimento Ao Vivo	Milton Nascimento
3	Ben Franklin	NaN	1.99	Comedy	The Office, Season 3	The Office
4	O Último Romântico (Ao Vivo)	NaN	0.99	Latin	Lulu Santos - RCA 100 Anos De Música - Álbum 02	Lulu Santos

```
itunes_df.tail()
```

	Track	Composer	UnitPrice	Genre	Album	Artist
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	None	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy
3499	String Quartet No. 12 in C Minor, D. 703 "Quar...	Franz Schubert	0.99	Classical	Schubert: The Late String Quartets & String Qu...	Emerson String Quartet
3500	L'orfeo, Act 3, Sinfonia (Orchestra)	Claudio Monteverdi	0.99	Classical	Monteverdi: L'Orfeo	C. Monteverdi, Nigel Rogers - Chiaroscuro; Lon...
3501	Quintet for Horn, Violin, 2 Violas, and Cello ...	Wolfgang Amadeus Mozart	0.99	Classical	Mozart: Chamber Music	Nash Ensemble
3502	Koyaanisqatsi	Philip Glass	0.99	Soundtrack	Koyaanisqatsi (Soundtrack from the Motion Pict...	Philip Glass Ensemble

Basic Exploratory Data Analysis (EDA)

General EDA Checklist

- Examine the top and bottom of the data
- **Examine the data's dimensions, datatypes and missing values**
- Investigate statistical properties of the data
- Create basic plots of the data

This EDA can provide a starting point for further analysis

```
itunes_df.shape()
```

```
(4021, 8)
```

```
itunes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4021 entries, 0 to 3502
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Track         4021 non-null   object
 1   Composer      2908 non-null   object
 2   Milliseconds  518 non-null    float64
 3   Bytes         518 non-null    float64
 4   UnitPrice     4021 non-null   float64
 5   Genre         4021 non-null   object
 6   Album         4021 non-null   object
 7   Artist        4021 non-null   object
dtypes: float64(3), object(5)
memory usage: 282.7+ KB
```

Columns Labels | # of Non-Null Values | Columns Data Types

```
itunes_df.isna().sum()
```

```
Track           0
Composer     1113
Milliseconds 3503
Bytes        3503
UnitPrice       0
Genre           0
Album           0
Artist          0
dtype: int64
```

Columns Labels | # of Null Values

Basic Exploratory Data Analysis (EDA)

General EDA Checklist

- Examine the top and bottom of the data
- Examine the data's dimensions, datatypes and missing values
- **Investigate statistical properties of the data**
- Create basic plots of the data

This EDA can provide a starting point for further analysis

For Numeric Columns

```
itunes_df.describe()
```

	Milliseconds	Bytes	UnitPrice
count	5.180000e+02	5.180000e+02	4021.000000
mean	3.868336e+05	3.040734e+07	1.050184
std	5.258469e+05	9.602387e+07	0.237857
min	4.884000e+03	1.612660e+05	0.990000
25%	2.049758e+05	6.493416e+06	0.990000
50%	2.526950e+05	8.098298e+06	0.990000
75%	3.225330e+05	1.010645e+07	0.990000
max	2.935894e+06	5.701522e+08	1.990000

```
itunes_df.corr()
```

	Milliseconds	Bytes	UnitPrice
Milliseconds	1.000000	0.942266	0.956721
Bytes	0.942266	1.000000	0.941954
UnitPrice	0.956721	0.941954	1.000000

Very Correlated

For Non-Numeric Columns

```
itunes_df['Genre'].mode()
```

```
0    Rock
Name: Genre, dtype: object
```

```
itunes_df['Genre'].value_counts()
```

```
Rock                1498
Latin                656
Metal                420
Alternative & Punk   393
Jazz                 160
TV Shows             105
Blues                 92
Classical             85
Drama                 71
R&B/Soul              69
Reggae                64
Pop                   51
Soundtrack            50
Alternative           45
Hip Hop/Rap           40
Electronica/Dance     35
World                 32
Heavy Metal           31
Sci Fi & Fantasy      31
Easy Listening        28
Comedy                20
Bossa Nova            17
Science Fiction       15
Rock And Roll         12
Opera                  1
Name: Genre, dtype: int64
```

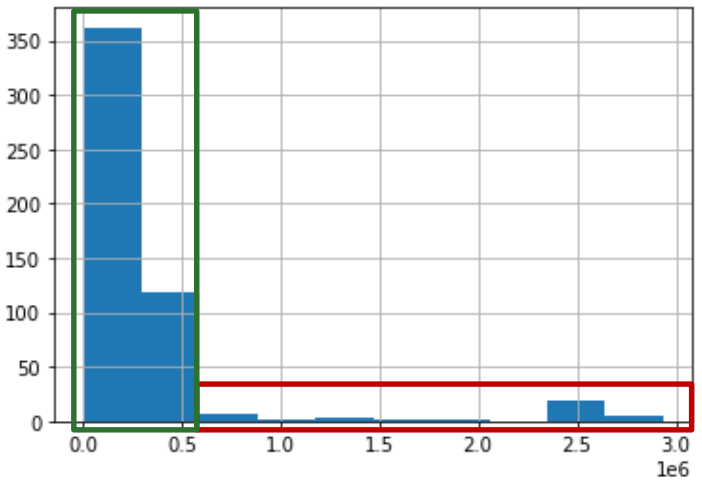
Basic Exploratory Data Analysis (EDA)

General EDA Checklist

- Examine the top and bottom of the data
- Examine the data's dimensions, datatypes and missing values
- Investigate statistical properties of the data
- **Create basic plots of the data**

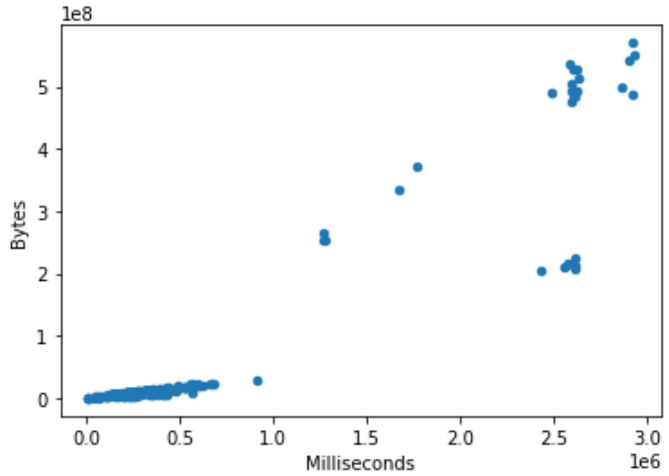
This EDA can provide a starting point for further analysis

```
import matplotlib.pyplot as plt
itunes_df['Milliseconds'].hist()
plt.show()
```



- Most of the songs have a shorter song length
- Some outliers with very long lengths

```
itunes_df.plot.scatter(x='Milliseconds',
                       y='Bytes')
plt.show()
```



Song length is correlated to other columns, so let's look at a scatter plot of song length and song size in bytes

Basic Exploratory Data Analysis (EDA)

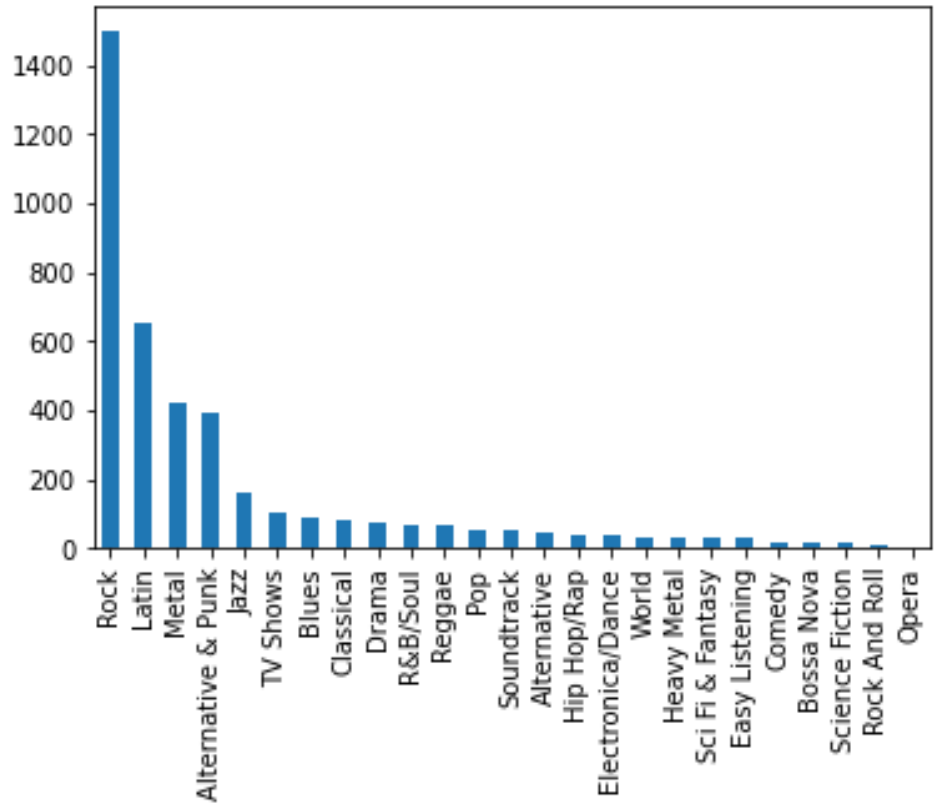
General EDA Checklist

- Examine the top and bottom of the data
- Examine the data's dimensions, datatypes and missing values
- Investigate statistical properties of the data
- **Create basic plots of the data**

This EDA can provide a starting point for further analysis

```
import matplotlib.pyplot as plt
```

```
itunes_df['Genre'].value_counts().plot.bar()
plt.show()
```



For non-numeric data, it is a good idea to use bar plots

Basic Data Cleaning Operations

General Data Cleaning Checklist

- **Removing irrelevant data**
- Dealing with missing values (filling in or dropping them)
- Dealing with outliers
- Dealing with duplicate values
- Ensuring datatypes are correct
- Standardizing data formats (e.g. mismatched capitalization, converting units)

- Data scientists spend 25% - 75% of their time cleaning data.

```
itunes_df.drop('Composer', axis=1, inplace=True)
itunes_df.columns
```

```
Index(['Track', 'Milliseconds', 'Bytes', 'UnitPrice', 'Genre', 'Album',
       'Artist'],
      dtype='object')
```

We may not really interested of the '**Composer**' column, since it contains a lot of NaN values (1113 NaN values)

```
only_music = itunes_df[~itunes_df['Genre'].isin(['Drama', 'TV Shows', 'Sci Fi & Fantasy', 'Science Fiction', 'Comedy'])]
```

	Track	Milliseconds	Bytes	UnitPrice	Genre	Album	Artist
1	Beira Mar	295444.0	9597994.0	0.99	Latin	Unplugged	Eric Clapton
2	Brasil	155428.0	5252560.0	0.99	Latin	Milton Nascimento Ao Vivo	Milton Nascimento
4	O Último Romântico (Ao Vivo)	231993.0	7692697.0	0.99	Latin	Lulu Santos - RCA 100 Anos De Música - Álbum 02	Lulu Santos
5	Freewheel Burning	265952.0	8713599.0	0.99	Metal	Living After Midnight	Judas Priest
6	That's The Way	343431.0	11248455.0	0.99	Rock	BBC Sessions [Disc 2] [Live]	Led Zeppelin
...	...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della V...	NaN	NaN	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy

We may only interested in **music** '**Genre**' songs

Basic Data Cleaning Operations

General Data Cleaning Checklist

- Removing irrelevant data
- **Dealing with missing values (filling in or dropping them)**
- Dealing with outliers
- Dealing with duplicate values
- Ensuring datatypes are correct
- Standardizing data formats (e.g. mismatched capitalization, converting units)

- Data scientists spend 25% - 75% of their time cleaning data.

- Drop the missing values

```
itunes_df.dropna(inplace=True)
```

```
itunes_df.dropna(thresh = 2, inplace=True)
```

Drop the rows with at least a missing value

- Fill the missing values with a specific value

```
itunes_df['Composer'].fillna('Unknown', inplace=True)
```

Good for Machine Learning Clustering Models Building

- Fill the missing values with the **mode** (most common value for a series of data)

```
itunes_df['UnitPrice'].fillna(itunes_df['UnitPrice'].mode(),inplace=True)
```

Make sense since **94%** of the values are **0.99** for UnitPrice

- Fill the missing values with the **mean**

```
itunes_df['UnitPrice'].fillna(itunes_df['UnitPrice'].mean(),inplace=True)
```

Make sense in cases the distribution of values is somewhat **Gaussian** (**Normal**) distribution

- Fill the missing values with the **a Machine Learning Technique (More Advanced)**

Most used one is the **Imputation** Techniques Implemented in **sklearn.impute** module.
Example: **sklearn.KNNImputer**

Basic Data Cleaning Operations

General Data Cleaning Checklist

- Removing irrelevant data
- Dealing with missing values (filling in or dropping them)
- **Dealing with outliers**
- Dealing with duplicate values
- Ensuring datatypes are correct
- Standardizing data formats (e.g. mismatched capitalization, converting units)

- Data scientists spend 25% - 75% of their time cleaning data.

```
itunes_df_clean = remove_outliers(itunes_df, 'Milliseconds')
```

- **For Categorical Data**
  - Remove rows with minority classes, like 'TV Shows', or,
  - Group all minority classes into a class labeled as '**Other**' for example
- **For Numerical Data**
  - Use the Interquartile Range (**IQR**) method formula:

$$IQR = 75_{Percentile} - 25_{Percentile}$$

  - For the outlier boundaries:

$$Upper_{bandary} = 75_{Percentile} + 1.5 * IQR$$

$$Lower_{bandary} = 25_{Percentile} - 1.5 * IQR$$

  - Then exclude outliers from a DataFrame using the following generic function:

```
def remove_outliers(df, column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    upper_boundary = q3 + 1.5 * iqr
    lower_boundary = q1 - 1.5 * iqr
    new_df = df.loc[(df[column] > lower_boundary) & (df[column] < upper_boundary)]
    return new_df
```

Other methods for dropping outliers can be found in : https://stackoverflow.com/questions/23199796/detect-andexclude-outliers-in-pandas-data-frame

Basic Data Cleaning Operations

General Data Cleaning Checklist

- Removing irrelevant data
- Dealing with missing values (filling in or dropping them)
- Dealing with outliers
- **Dealing with duplicate values**
- **Ensuring datatypes are correct**
- Standardizing data formats (e.g. mismatched capitalization, converting units)

- Data scientists spend 25% - 75% of their time cleaning data.

- **Checking For Duplicated Rows**

```
itunes_df.duplicated().sum()
```

```
20
```

- **Delete Duplicated Rows**

```
itunes_df.drop_duplicates(inplace=True)
```

- **Checking Again**

```
itunes_df.duplicated().sum()
```

```
0
```

- **Ensuring Datatypes**

```
itunes_df['Milliseconds'] = itunes_df['Milliseconds'].astype('int')
```

Basic Data Cleaning Operations

General Data Cleaning Checklist

- Removing irrelevant data
- Dealing with missing values (filling in or dropping them)
- Dealing with outliers
- Dealing with duplicate values
- Ensuring datatypes are correct
- **Standardizing data formats (e.g. mismatched capitalization, converting units)**

- Data scientists spend 25% - 75% of their time cleaning data.

- **Data transformations: Adding Culumns Data**

```
itunes_df['Seconds'] = itunes_df['Milliseconds'] / 1000
```

```
itunes_df['len_byte_ratio'] = itunes_df['Milliseconds']/ itunes_df['Bytes']
```

- **Using replace, map, and apply to clean and transform data**

```
genre_dict = {'metal': 'Metal', 'met': 'Metal'}
itunes_df['Genre'].replace(genre_dict)
```

- **Using Pandas pre-built function to transform data**

```
itunes_df['Genre'] = itunes_df['Genre'].str.lower()
```

```
itunes_df['Genre'] = itunes_df['Genre'].str.upper()
```

More on data cleaning can be found in: https://datagy.io/pandas-data-cleaning/
More on pandas can be found in: https://pandas.pydata.org/docs/getting_started/index.html

Thanks for your Listening

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz