Annexure-I

Introduction To Data Science

Upgrad

SIX WEEKS SUMMER TRAINING REPORT

Submitted in partial fulfilment of the requirements for the award of degree of

Computer Science and Engineering

(Data Science)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 28-05-2022 to 10-07-2022

Name of student: Singam Setty Sai Sumanth

Registration Number: 12001822

Signature of the student:

S. Sai. Sumanth

Annexure-II: Student Declaration

To whom so ever it may concern

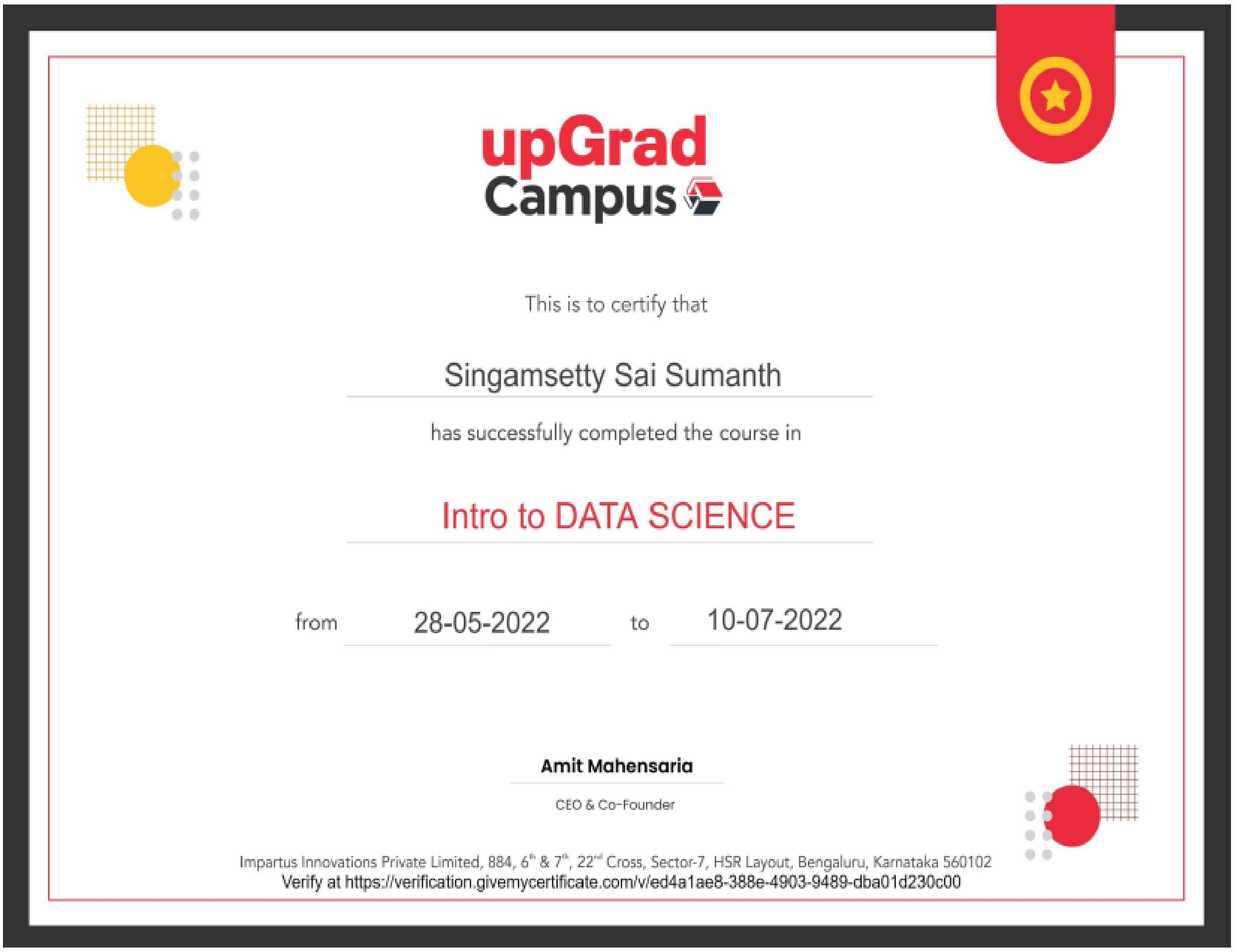
I, Sai Sumanth,12001822, hereby declare that the work done by me on “Introduction to Data Science” from May 2022 to July 2022, is a record of original work for the partial fulfilment of the requirements for the award of the degree, Computer Science Engineering.

Singam Setty Sai Sumanth (12001822),

S. sai. Sumanth

Dated: 23-07-2022

Student Training Certification



upGrad Campus

This is to certify that

Singamsetty Sai Sumanth

has successfully completed the course in

Intro to DATA SCIENCE

from 28-05-2022 to 10-07-2022

Amit Mahensaria

CEO & Co-Founder

Impartus Innovations Private Limited, 884, 6th & 7th, 22nd Cross, Sector-7, HSR Layout, Bengaluru, Karnataka 560102
Verify at https://verification.givemycertificate.com/v/ed4a1ae8-388e-4903-9489-dba01d230c00

Acknowledgement

The opportunity of attaining a course based **on Introduction to Data Science Specialization** through Coursera, provided by UPGRAD was worth learning. It was a prestige for me to be part of it. During the period of my course, I received tremendous knowledge related to Excel, Python, SQL and Statistics.

Pre-eminently, I would like to express my deep gratitude and special thanks to my course instructors of UPGRAD for their theoretical knowledge and encouragement on this project and for their valuable guidance and affection for the successful completion of this work.

Secondly, I would like to thank Lovely Professional University for giving me an opportunity to learn this course by sponsoring my Coursera subscription.

Lastly, I would like to thank my parents for their constant encouragement, moral support, personal attention, care and guidance for choosing this course.

Singam Setty Sai Sumanth (12001822).

Abstract

Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses complex machine learning algorithms to build predictive models.

The data used for analysis can come from many different sources and presented in various formats. Data science may detect patterns in seemingly unstructured or unconnected data, allowing conclusions and predictions to be made. Tech businesses that acquire user data can utilise strategies to transform that data into valuable or profitable information. Data Science has also made inroads into the transportation industry, such as with driverless cars. It is simple to lower the number of accidents with the use of driverless cars. For example, with driverless cars, training data is supplied to the algorithm, and the data is examined using data Science approaches, such as the speed limit on the highway, busy streets, etc. Data Science applications provide a better level of therapeutic customisation through genetics and genomics research.

In this report, I have shared a project where I have done data analysis of a movie's data set. This report also presents my learning and contributions during my summer training.

Table of Contents

Chapter 1- Introduction

"Introduction to Data Science" is an online specialization course offered by UPGRAD consisting of 4 courses "Introduction to Excel", "Fundamentals of Python and Libraries", "Data Visualization Using tableau & SQL" and "Statistics for Data Science". This specialization teaches the essential skills for working with large-scale data.

Application of Data Science:

- Recommendation System:

  Ex: In Amazon recommendations are different for different users according to their past search.

- Social Media:
  1. Recommendation Engine
  2. Ad placement
  3. Sentiment Analysis
  - Deciding the right credit limit for credit card customers.
  - Suggesting right products from e-commerce companies
    1. Recommendation System
    2. Past Data Searched
    3. Discount Price Optimization
  - How google and other search engines know what are the more relevant results for our search query?
    1. Apply ML and Data Science
    2. Fraud Detection
    3. AD placement
    4. Personalized search results

Overview of Training Course

I completed this MOOC through Coursera, it was a specialization course comprised of 4 other courses, the total duration of this specialization is approximately 6 weeks, I completed this specialization in the given time. This specialization course was provided by UPGRAD. All the courses comprised multiple video lectures, reading materials, quizzes, and multiple assignments. Once the course has started all the tasks like completing video lectures, reading materials, including quizzes and assignments were to be completed before the deadline. Also, to pass the course one must have to score a minimum of 80% in all the quizzes. Also, the submitted assignments must have to be checked and awarded a minimum of 80% marks by fellow peers. Also, I must evaluate at least 3 assignments of my peers to pass the courses. Although I had done few assignments in the specialization, I decided to work on some other projects to evaluate my skills.

Course 1: Introduction to Excel

In this course I learned how to prepare well-formatted reports using sort / filter operations and advanced formatting techniques. We analysed a telemarketing business problem to find the types of customers which should be targeted. You will accomplish the same using formulae, pivot tables, visualisations, V-Lookups etc. This course was started with simple things like make us familiarize to the ecosystem of excel and understanding them. We began simple functions like sort and filter, report making: Basic and Advanced formatting, Printing and Page layouts, Passwords and Naming files etc...

This course is taught to us by a very special person, who is the CEO of "Gramener". His name is Anand. Gramener is one of the most prominent data analytics and visualisation companies in India. Anand, currently the CEO, has previously been the Chief Data Scientist at Gramener and has extensive experience in management consulting and equity research.

Course 2: Fundamentals of Python and Libraries

This course consists of 3 modules

In module 1, we learned about the basics, Lists, tuples, Dictionaries, Sets, If-Elif-Else, Loops, comprehensions, Functions such as (Lambda, Map, Filter and reduce)

Lists: A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [].

Tuples: Just like lists, tuples also contain a set of objects in an ordered manner. These objects are kept separated by commas. Tuples are immutable and do not allow the entrance of additional objects once a tuple is created. Tuples are incapable of expanding or making modifications; therefore, they differ from lists. Removing elements is also not possible from tuples which restrict its collection. The immutability often serves as an advantage in delivering faster, efficient results. Tuples are defined by having values between normal brackets ().

Dictionaries: Dictionaries are Python's implementation of a data structure that is more generally known as an associative array. A dictionary consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value. You can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces {}. A colon (:) separates each key from its associated value.

Sets: A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed). However, a set itself is mutable. We can add or remove items from it. Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc. A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set () function. It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

If-Elif-Else: Decision making is required when we want to execute a code only if a certain condition is satisfied. The if…elif…else statement is used in Python for decision making.

Syntax of if...elif...else:

```
if test expression:
    Body of if
elif test expression:
      Body of elif
else: Body of else
```

Loops: loops are available in Python to fulfil the looping needs. Python offers 3 choices for running the loops. The basic functionality of all the techniques is the same, although the syntax and the amount of time required for checking the condition differ.

While loop: Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

For loop: This type of loop executes a code block multiple times and abbreviates the code that manages the loop variable.

Nested loops: We can iterate a loop inside another loop.

Comprehensions: Comprehensions in Python provide us with a short and concise way to construct new sequences (such as lists, set, dictionary etc.) using sequences which have been already defined. Python supports the following 4 types of comprehensions: List Comprehensions, Dictionary Comprehensions, Set Comprehensions, Generator Comprehensions.

List Comprehensions: List Comprehensions provide an elegant way to create new lists. Syntax: output_list = [output_exp for var in input_list if (var satisfies this condition)]

Dictionary Comprehensions: Extending the idea of list comprehensions, we can also create a dictionary using dictionary comprehensions. The basic structure of a dictionary comprehension looks like below.

output_dict = {key: value for (key, value) in iterable if (key, value satisfy this condition)}

Set Comprehensions: Set comprehensions are pretty similar to list comprehensions. The only difference between them is that set comprehensions use curly brackets {}.

Generator Comprehensions: Generator Comprehensions are very similar to list comprehensions. One difference between them is that generator comprehensions use circular brackets whereas list comprehensions use square brackets. The major difference between them is that generators don't allocate memory for the whole list. Instead, they generate each value one by one which is why they are memory efficient.

Functions: A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result. In Python a function is defined using the "def" keyword.

Lambda Function: A lambda function is a small anonymous function. A lambda function can take any number of arguments but can only have one expression.

Syntax: lambda arguments: expression

Map Functions: The map () function executes a specified function for each item in an iterable. The item is sent to the function as a parameter.

Syntax: map (function, iterables)

Filter Function: The filter () function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Syntax: filter (function, iterable)

Reduce Function: The reduce () function is used to apply a particular function passed in its argument to all the list elements mentioned in the sequence passed along. This function is defined in “functools” module.

Syntax: reduce (fun, seq)

In module 2, we learned about “NumPy”

NumPy: NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also, it is optimized to work with latest CPU architectures.

In module 3, we learned about “Pandas”

Pandas: Pandas is a library built using NumPy specifically for data analysis. You'll be using Pandas heavily for data manipulation, visualisation, building machine learning models, etc. There are two main data structures in Pandas - Series and Dataframes. The default way to store data is dataframes, and thus manipulating dataframes quickly is probably the most important skill set for data analysis. The Pandas Series A series is like a 1-D NumPy array and contains scalar values of the same type (numeric, character, datetime etc.). A dataframe is simply a table where each column is a pandas series. You can create pandas series from array-like objects using “pd. Series ()”.

While creating a series, Pandas automatically indexes it from 0 to (n-1), with n being the number of rows. But, if you want, you can also explicitly set the index yourself, using the ‘index’ argument — while creating the series using pd. Series (): Example: pd. Series ([1, 2, 3], index = [‘a’, ‘b’, ‘c’]) creates the

following series:
a 1

b 2

c 3

Indexing and Slicing data: Indexing and Slicing are two of the most common operations that you need to be familiar with when working with Numpy arrays. You will use them when you would like to work with a subset of the array. This guide will take you through a little tour of the world of Indexing and Slicing on multi-dimensional arrays.

Selecting rows: df [start: stop]

Selecting columns: df['column'] or df. column or df [['col_x', 'col_y']]

df ['column'] or df. column return a series

df [['col_x', 'col_y']] returns a dataframe

There are two main ways of indexing dataframes:

Position based indexing using df.iloc

Label based indexing using df.loc

Pandas series and data frames which are the basic data structures in Pandas library. Indexing, Selecting and Subsetting a dataframe. Merging and appending two dataframes which can be done using. merge and. concat commands. Grouping and Summarising dataframes which can be done using groupby () to make an object then using this object to play around. Using Pivot table function on a data frame which are similar to pivot tables provided in M.S Excel.

Course 3: Data Visualization Using Tableau and SQL

In this course, we had gone through the basic concepts of RDMS, SQL, SQL Clauses, Data Retrieval with SQL, Basics of Sorting.

Tableau Module:

In this module, I learnt about a Tableau, a business intelligence software that allows anyone to connect easily to data, visualise and create interactive dashboards. The module focuses on the usage of Tableau to create interactive plots, dashboards and so-called 'stories'.

Tableau is a powerful tool to visualise even the most complex of data. By the end of this module, you will be able to do your analysis on data sets and build dashboards using Tableau.

upGrad has collaborated with Tableau, and hence, we will be using their official teaching videos as a part of this module.
In this module, I got familiar with the Tableau interface and learned to navigate through it. Before any visualisations can be made, data must be imported. You will understand how to import data into Tableau and how to prepare it for creating visualisations. I also learned about the features Tableau offers to manipulate data such as sorting, splitting etc.

RDBMS CONCEPTS

Taking a one real life example, during your supermarket visits, The cashier scans the product codes and the bill for your purchased items is generated automatically. Where does he retrieve the product code details from? Also, when you make a purchase, the transaction details like your card number, the amount of purchase etc. is automatically stored in some 'database in backend'.

A database is an organised collection of related data. Every domain whether it is banking, e-commerce, retail sale outlets etc. maintain a database relevant to their industry through DBMS.

DBMS stands for Database Management System which is a software capable of managing a huge volume of data in a highly organised manner. There are various ways to arrange and manage data in a database. The most common is to arrange the data in tables, which is similar to an Excel file. The table contains multiple columns and rows.

Characteristics of a Database:

1. Schema: A schema is a visual representation of the blueprint of a database. It tells a user the number of tables in a database and the relationship followed between two tables. For e.g., in the ‘Company DB Schema’, there are 6 tables and each of those tables follow a fixed format or structure.
2. Table: A table is a collection of data in the form of rows and columns as you can see in the ‘Company DB Schema’.
3. Primary Key: It is an attribute which allows a user to identify a table. Only the attribute having unique values in a table is eligible to be qualified as a key. For e.g., the ‘Ssn’ column in the ‘EMPLOYEE’ table has unique values so, ‘Ssn’ can be used as a primary key for this table.
4. Relation: It tells a user how two tables in a schema are related. Two tables in a schema might have a column which is common in both. Hence, this common column becomes a relation. For e.g., in the

‘Company DB Schema’, the ‘EMPLOOYEE’ and ‘WORKS_ON’ tables are related to each other through ‘Ssn’ and ‘Essn’ columns. The values of these two columns are common to both tables.

5. Constraint: Refer to the ‘EMPLOYEE’ and the ‘DEPARTMENT’ tables. The ‘EMPLOYEE’ table cannot have departments which do not exist in this company. The information about existing departments is present in the ‘DEPARTMENT’ table. This is an example of restriction. Any table in a schema cannot have information which does not exist in the schema in another form.
6. Foreign Key: It is an attribute which one table uses to refer to the contents of another table. The ‘EMPLOYEE’ table refers to the ‘DEPARTMENT’ table through ‘Dno’ column whose values are also present in the ‘DEPARTMENT’ table in the ‘Dnumber’ column. So, ‘Dno’ is a foreign key for the ‘EMPLOYEE’ table and ‘Dnumber’ is a primary key for the ‘DEPARTMENT’ table. Thus, it can be said that a foreign key in one table refers to the primary key of another.
7. Data types: They describes the nature of data in a column. They tell us whether the data is of string type, integer type, date type, Boolean type etc.

DATA ABSTRACTION:

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. However, for the system to be usable, data must be retrieved efficiently. The efficiency led to the design of complex data structure for the representation of data in the database. Certain complexity must be hidden from the database system users. This accomplished by defining several levels of abstraction at which the database may be viewed.

CLASSIFICATION OF DATABASE:

There are 3 types of database approaches given below,

Hierarchical Database:

In this type of model data is represented in simple tree structured. The record at the top of three is known as root, the root may have any number of dependents. Each of these may have any number of low-level dependents and so on up to any number of levels. The disadvantages of the approach are that no independent record occurrence can exist without it's superior.

Network Database:

In a Network database, data is represented by Network structure. In this approach record occurrence can have any number of superiors as well as any number of immediate dependents thus allow many to many correspondences directly than a hierarchical approach. The main disadvantage of the Network model is data representation is very complex resulting in complexity of the DML (Data Manipulation Language).

Relational Database:

The Relational database is a way of structuring information in tables, rows and columns. An RDB has the ability to establish links or relationships between information by joining tables which makes it easy to understand. The Relational model represents data and relationships among data by a collection of tables each of which has several columns with unique names.

Basically, Managing the data through different files (JSON, CSV, text etc.) is always risky and may generate inconsistencies. An organization cannot efficiently track its business activities by maintaining thousands of files.

For Efficient access and management of data, there is a software known as Database Management System (DBMS). Which is widely used in the industry to analyse the data easily. A language called Structural Query Language (SQL) is used for this purpose.

THE SQL LANGUAGE

SQL is a language for relational database. SQL is a non-procedural i.e., when we use SQL, we specify what we want to be done not how to do it.

Features Of SQL

1. SQL is an interactive query language.

2. SQL is a database administration language.

3. SQL is a database programming language.

4. SQL is a client/server language.

5. SQL is a distributed database language.

6. SQL is a database gateway language.

Basic SQL Commands:

Data Definition Language commands (DDL)

Data Manipulation Language commands (DML)

Transaction Control Language commands (TCL)

Data control Language commands (DCL)

Basics of Sorting:

Sorting and Ordering is a common operation to arrange data in either increasing or decreasing order. In SQL, Sorting is done using the clauses ‘asc’ and ‘desc’ for ascending and descending order respectively.

Some of the clauses that we had learnt are-

‘IN’ and ‘NOT IN’: An Example of an SQL query using ‘IN’ clause is shown below.

```
select * from dept_locations
where dlocation in ('Houston’, ‘Stafford');
```

Similarly, ‘NOT IN” clause can also be used.

‘IS NULL’: An Example of an SQL query using ‘IS NULL’ clause is shown below.

```
select * from employee where super_ssn is null;
```

Course 4: Statistics for Data Science

In this course, we had 3 modules and they had thought us a set of techniques to display data in such a way that interesting features will become apparent. Data Sourcing, Data Cleaning, Values, Univariate and Bivariate Analysis, Derived Metrics and few basics of Inferential Statistics.

Module 1- Exploratory Data Analysis:

Exploratory data analysis popularly known as EDA is a process of performing some initial investigations on the dataset to discover the structure and the content of the given dataset. It is often known as Data Profiling. It is an unavoidable step in the entire journey of data analysis right from the business understanding part to the deployment of the models created.

Data Sourcing:

To solve a business problem using analytics, you need to have historical data. Data is the key - the better the data, the more insights you can get out of it. Typically, data comes from various sources and your first job as a data analyst is to procure the data from them. In this session, you will learn about various sources of data and how to source data from public and private sources. The broad agenda for this session is as follows:

- **Private Data**
- **Public Data**

Data Cleaning:

To identify the various quality issues and techniques to clean data. Though data cleaning is often done in a somewhat haphazard way, and it is too difficult to define a ‘single structured process’, we will study data cleaning in the following steps:

1. Fix rows and columns
2. Fix missing values
3. Standardise values
4. Fix invalid values
5. Filter data

Fixing Rows and Columns

1)Checklist for Fixing Rows

2)Delete summary rows

3)Delete incorrect rows

4)Delete extra rows

Checklist for Fixing Columns

1)Merge columns for creating unique identifiers if needed

2)Split columns for more data: For e.g., split address to get state and city to analyse each separately

3)Add column names (if the names are missing)

4)Rename columns consistently (if required)

5)Delete unnecessary columns (if required)

6)Align misaligned columns

Missing Values:

1)Some of the points that will help you in dealing with the missing values: Set values as missing values: Identify values that indicate missing data, and yet are not recognised by the software as such, e.g., treat blank strings, "NA", "XX", "999", etc. as missing.

2)Adding is good, exaggerating is bad: You should try to get information from reliable external sources as much as possible, but if you can't, then it is better to keep missing values as such rather than extrapolating the information in the existing rows/columns.

3)Delete rows, columns: Rows could be deleted if the number of missing values is insignificant in number, as this would not impact the analysis. Columns could be removed if the missing values are quite significant in number.

4)Fill partial missing values using business judgement: Missing time zone, century, etc. These values are easily identifiable.

Standardising Values:

Removing outliers is an important step in data cleaning. An outlier may disproportionately affect the results of your analysis. This may lead to faulty interpretations. It is also important to understand that there is no fixed definition of an outlier. It is left up to the judgment of the analyst to decide the criteria on which data would be categorised as abnormal or an outlier.

Standardise units: Ensure all observations under a variable have a common and consistent unit, e.g., convert lbs to kg, miles/hr to km/hr, etc. Scale values if required: Make sure the observations under a variable have a common scale. Standardise precision for better presentation of data, e.g., 4.5312341 kgs to 4.53 kgs.

Remove outliers: Remove high and low values that would disproportionately affect the results of your analysis

Remove extra characters like such as common prefix/suffix, leading/trailing/multiple spaces, etc. These are irrelevant to the analysis. Standardise case: There are various cases that string variables may take, e.g., UPPERCASE, lowercase, Title Case, Sentence case, etc.

Standardise format: E.g., 23/10/16 to 2016/10/23, “Modi, Narendra" to “Narendra Modi", etc.

Invalid Values:

A data set can contain invalid values in various forms. Some of the values could be truly invalid, e.g., a string “tr8ml” in a variable containing mobile numbers would make no sense and hence would be better removed. Similarly, a height of 11 ft would be an invalid value in a set containing heights of children.

On the other hand, some invalid values can be corrected. For e.g., numeric value with a data type of string could be converted to its original numeric type. Issues might arise due to python misinterpreting the encoding of a file, thus showing junk characters where there were valid characters. This could be corrected by correctly specifying the encoding or converting the data set to the accurate format before importing.

If you have an invalid value problem, and you do not know what accurate values could replace the invalid values, it is recommended to treat these values as missing. For, e.g., in the case of a string “tr8ml” in a 'Contact' column, it is recommended to remove the invalid value and treat it as a missing value.

Encode Unicode properly: In case the data is being read as junk characters, try to change encoding, e.g., CP1252 instead of UTF-8.
Convert incorrect data types: Correct the incorrect data types to the correct data types for ease of analysis. For e.g., if numeric values are stored as strings, it would not be possible to calculate metrics such as mean, median, etc. Some of the common data type corrections are — a string to a number: "12,300" to “12300”; string to date: "2013-Aug" to "2013/08"; a number to a string: "PIN Code 110001" to "110001"; etc.
Correct values that go beyond range: If some of the values are beyond logical range, e.g., temperature less than -273° C (0° K), you would need to correct them as required. A close look would help you check if there is scope for correction, or if the value needs to be removed.

Correct values not in the list: Remove values that don’t belong to a list. For e.g., in a data set containing blood groups of individuals, strings "E" or "F" are invalid values and can be removed.

Correct wrong structure: Values that don't follow a defined structure can be removed. For e.g., in a data set containing pin codes of Indian cities, a pin code of 12 digits would be an invalid value and needs to be removed. Similarly, a phone number of 12 digits would be an invalid value.

Validate internal rules: If there are internal rules such as a date of a product's delivery must be after the date of the order, they should be correct and consistent.

Module 2- Uni/Bivariate Analysis:

Univariate Analysis:

It deals with analysing variables one at a time. It is important to separately understand each variable before moving on to analysing multiple variables together.

The broad agenda for this session is as follows:

- Metadata description
- Data distribution plots
- Summary metrics

Bivariate Analysis:

To understand the 'relationship between two variables', which is called **Bivariate Analysis** and learned the following topics:

- Bivariate analysis of continuous variables
- Bivariate analysis of categorical variables

Derived Metrics:

To create new variables using existing ones and get meaningful information by analysing them. In other words, we will discuss some methods to **derive new metrics** from the **existing ones**. The agenda for the session is as follows:

1. Type-driven metrics
2. Business-driven metrics
3. Data-driven metrics

Module 3- Inferential Statistics:

It includes Basics of Probability, Random Variables, Probability Distribution, Expected values, Discrete and Continuous Probability distribution, Central Limit Theorem.

However, sometimes you may require a very large amount of data for your analysis which may need too much time and resources to acquire. In such situations, you are forced to work with a smaller sample of the data, instead of having the entire data to work with.

Ex: Amazon QC department wants to know what proportion of the products in its warehouses are defective. Instead of going through all of its products (which would be a lot!), the Amazon QC team can just check a small sample of 1,000 products and then find, for this sample, the defect rate (i.e., the proportion of defective products). Then, based on this sample's defect rate, the team can **infer** what the defect rate is for all the products in the warehouses. This process of inferring insights from sample data is called **Inferential Statistics**. Note that even after using inferential statistics, you would only be able to estimate the population data from the sample data, but not find the exact values. This is because when you don't have the exact data, you can only make reasonable estimates about it with a limited level of certainty. Therefore, when certainty is limited, we talk in terms of probability.

Hypothesis Testing:

Hypothesis testing is a type of inferential statistics that is used to test assumptions and draw conclusions about the population from the available sample data. It involves setting up a null hypothesis and an alternative hypothesis followed by conducting a statistical test of significance. A conclusion is drawn based on the value of the test statistic, the critical value, and the confidence intervals. A hypothesis test can be left-tailed, right-tailed, and two-tailed. Given below are certain important hypothesis tests that are used in inferential statistics.

Z Test: A z test is used on data that follows a normal distribution and has a sample size greater than or equal to 30. It is used to test if the means of the sample and population are equal when the population variance is known. The right tailed hypothesis can be set up as follows: Null Hypothesis: H 0: μ = μ 0

Alternate Hypothesis: H 1: μ > μ 0

Test Statistic: z = $\overline{}$ x − μ σ √ n. $\overline{}$ x is the sample mean, μ is the population mean, σ is the population standard deviation and n is the sample size.

Decision Criteria: If the z statistic > z critical value then rejects the null hypothesis.

T Test: A t test is used when the data follows a student t distribution, and the sample size is lesser than 30. It is used to compare the sample and population mean when the population variance is unknown. The hypothesis test for inferential statistics is given as follows:

Null Hypothesis: H 0: μ = μ 0

Alternate Hypothesis: H 1: μ > μ 0

Test Statistics: t = $\overline{}$ x − μ s √ n Decision Criteria: If the t statistic > t critical value then rejects the null hypothesis.

F Test: An f test is used to check if there is a difference between the variances of two samples or populations. The right tailed f hypothesis test can be set up as follows:

Null Hypothesis: H 0: σ 2 1 = σ 2 2

Alternate Hypothesis: H 1: σ 2 1 > σ 2 2 T

Est Statistic: f = σ 2 1 σ 2 2, where σ 2 1 is the variance of the first population and σ 2 2 is the variance of the second population.

Decision Criteria: If the f test statistic > f test critical value then rejects the null hypothesis.

Confidence Interval: A confidence interval helps in estimating the parameters of a population. For example, a 95% confidence interval indicates that if a test is conducted 100 times with new samples under the same conditions, then the estimate can be expected to lie within the given interval 95 times. Furthermore, a confidence interval is also useful in calculating the critical value in hypothesis testing.

Apart from these tests, other tests used in inferential statistics are the ANOVA test, Wilcoxon signed-rank test, Mann-Whitney U test, Kruskal-Wallis H test, etc.

Regression Analysis:

Regression analysis is used to quantify how one variable will change with respect to another variable. There are many types of regressions available such as simple linear, multiple linear, nominal, logistic, and ordinal regression. The most used regression in inferential statistics is linear regression. Linear regression checks the effect of a unit change of the independent variable in the dependent variable. Some important formulas used in inferential statistics for regression analysis are as follows:

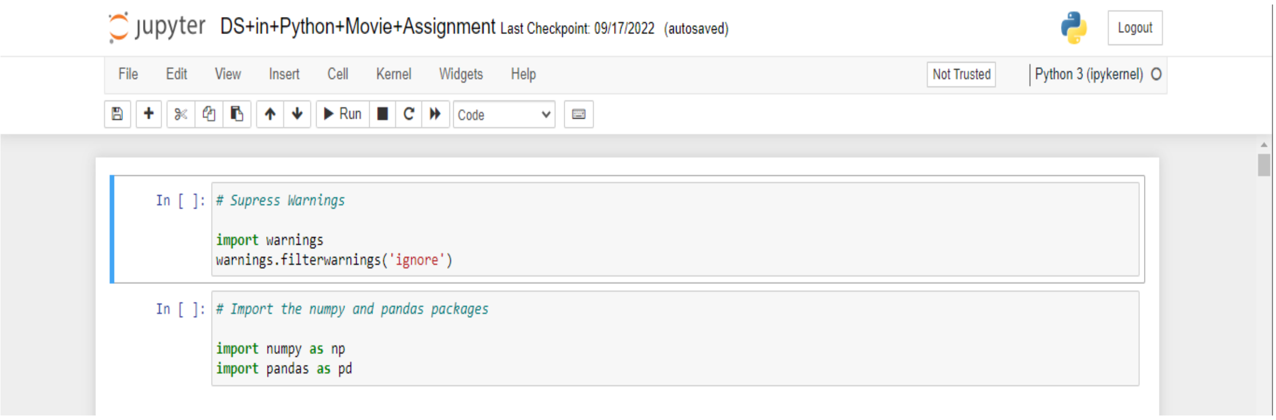
Regression Coefficients: The straight-line equation is given as y = α + β x, where α and β are regression coefficients.

β = ∑ n 1 (x i − ¯¯¯ x) (y i − ¯¯¯ y) ∑ n 1 (x i − ¯¯¯ x) 2 β = r x y σ y σ x
α = ¯¯¯ y − β ¯¯¯ x

Here, ¯¯¯ x is the mean, and σ x is the standard deviation of the first data set. Similarly, ¯¯¯ y is the mean, and σ y is the standard deviation of the second data set.

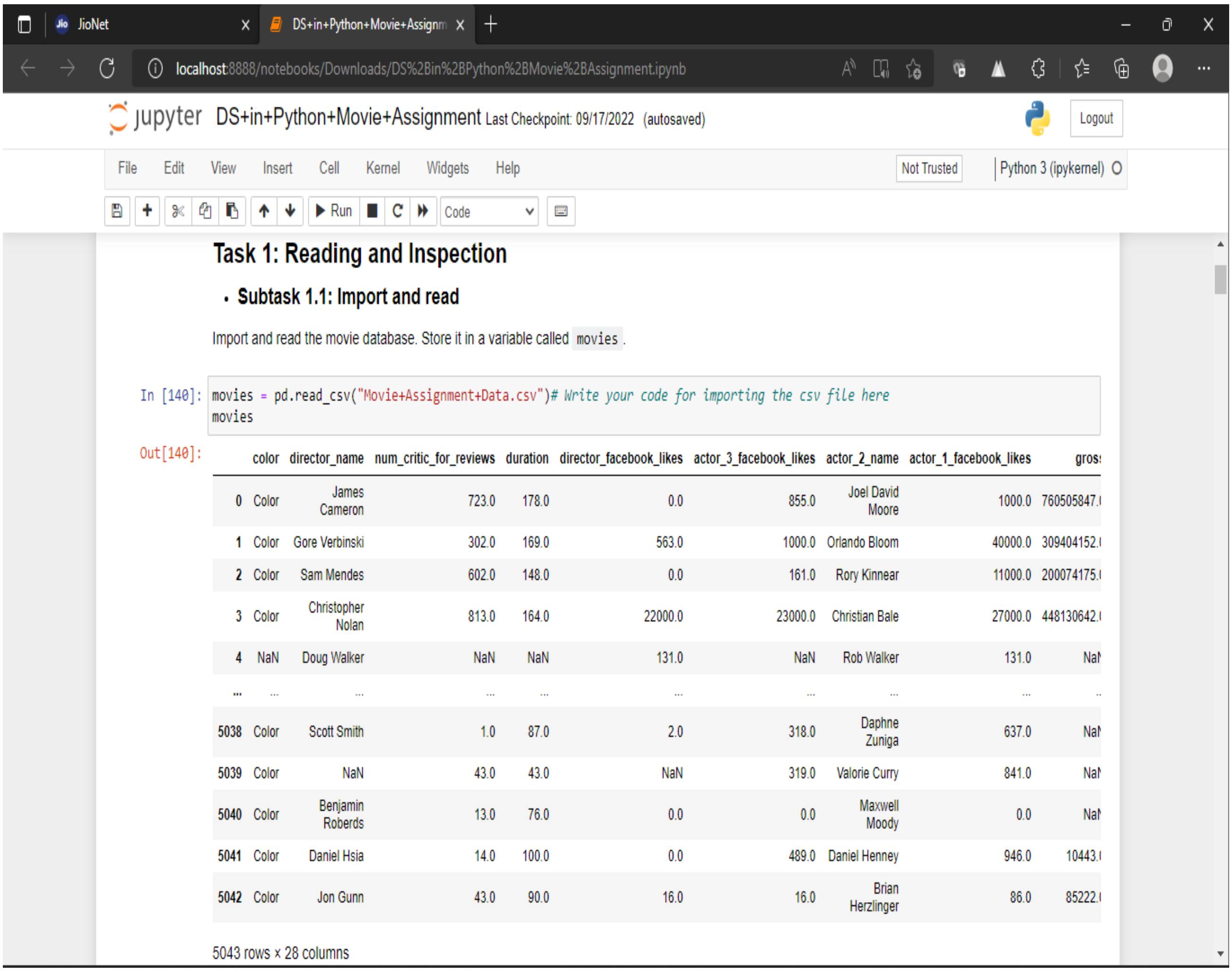
Chapter 3 – Screenshots of Project

Finally, the last chapter of this report is the Project which I had done on "**Movies Data Set**" using python language to extract the details requested from data sets.


jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [ ]: # Supress Warnings

        import warnings
        warnings.filterwarnings('ignore')

In [ ]: # Import the numpy and pandas packages

        import numpy as np
        import pandas as pd
```

The First Task is to import and read the Movies Database


JioNet
DS+in+Python+Movie+Assignm
localhost:8888/notebooks/Downloads/DS%2Bin%2BPython%2BMovie%2BAssignment.ipynb
jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

Task 1: Reading and Inspection

- **Subtask 1.1: Import and read**

Import and read the movie database. Store it in a variable called `movies`.

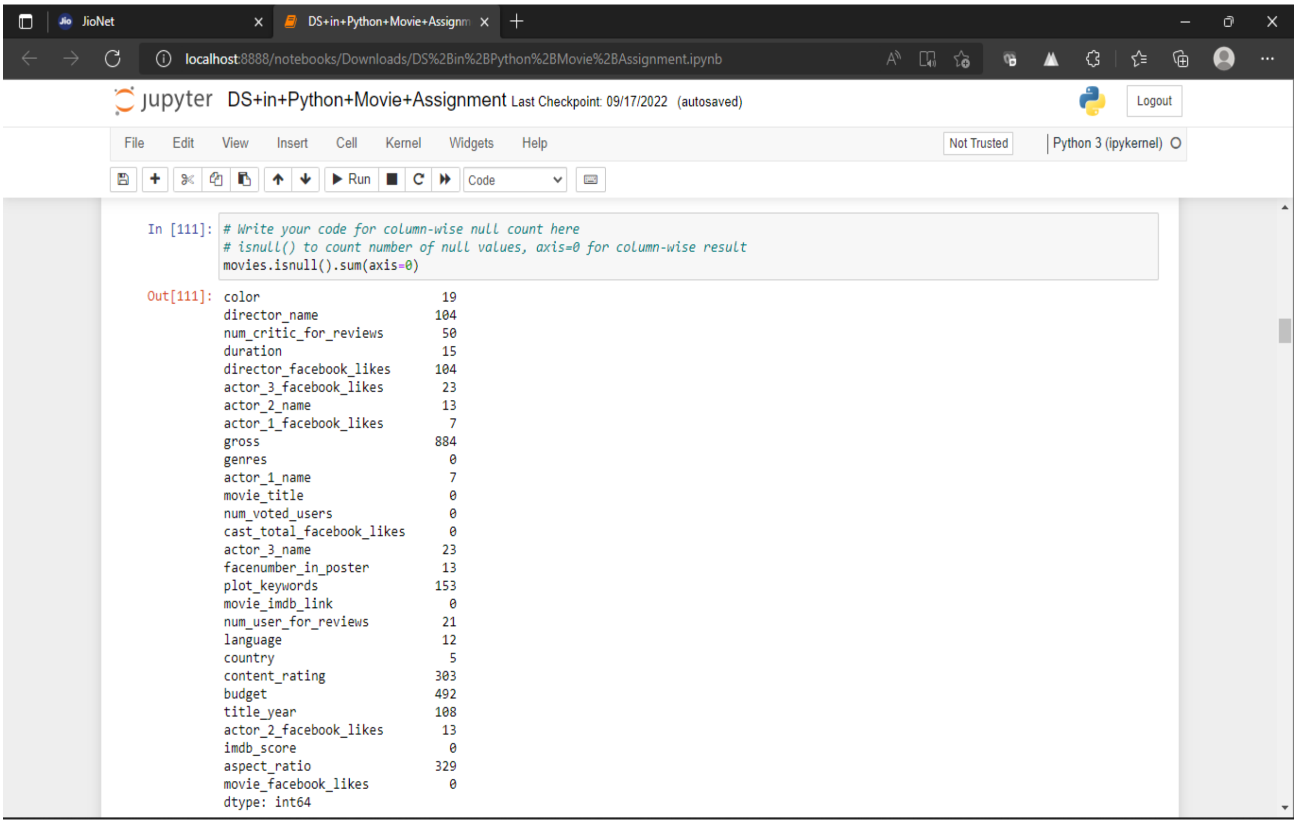
```
In [140]: movies = pd.read_csv("Movie+Assignment+Data.csv")# Write your code for importing the csv file here
          movies
```

Out[140]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gros
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	Na
...	...	...	...	...	...	...	...	...	..
5038	Color	Scott Smith	1.0	87.0	2.0	318.0	Daphne Zuniga	637.0	Na
5039	Color	NaN	43.0	43.0	NaN	319.0	Valorie Curry	841.0	Na
5040	Color	Benjamin Roberds	13.0	76.0	0.0	0.0	Maxwell Moody	0.0	Na
5041	Color	Daniel Hsia	14.0	100.0	0.0	489.0	Daniel Henney	946.0	10443.
5042	Color	Jon Gunn	43.0	90.0	16.0	16.0	Brian Herzlinger	86.0	85222.

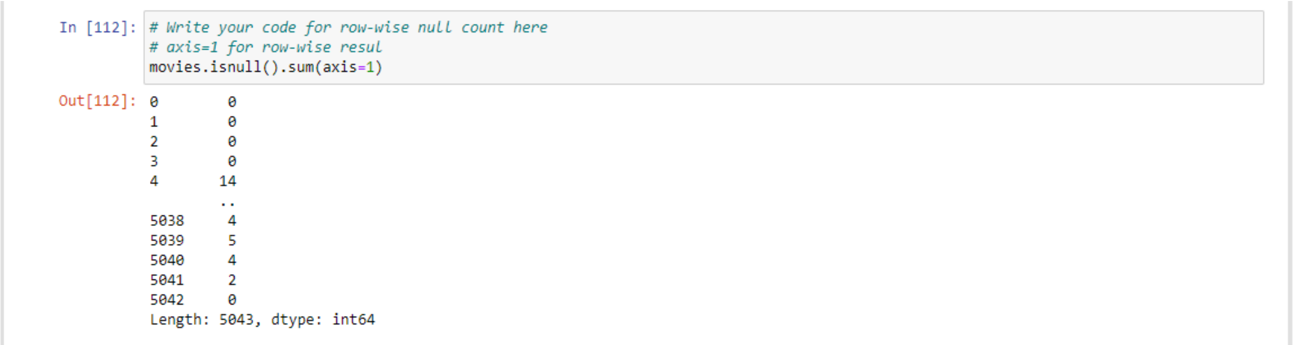
5043 rows × 28 columns

The Task is to clean the data and inspect the null values.

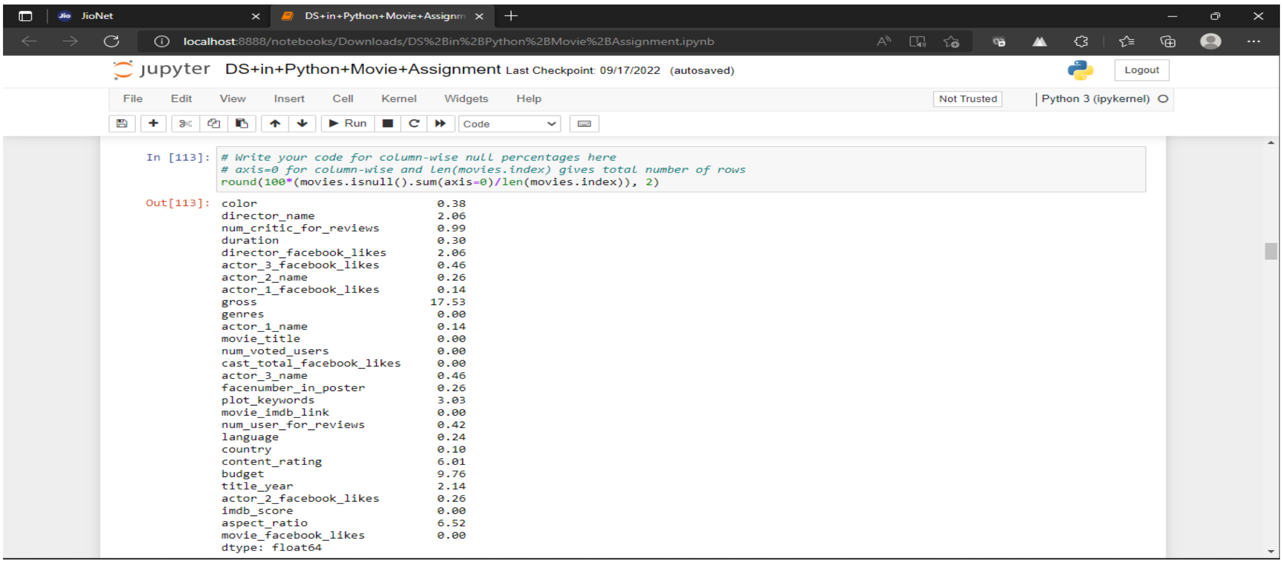


JioNet
DS+in+Python+Movie+Assignm
localhost:8888/notebooks/Downloads/DS%2Bin%2BPython%2BMovie%2BAssignment.ipynb
jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [111]: # Write your code for column-wise null count here
          # isnull() to count number of null values, axis=0 for column-wise result
          movies.isnull().sum(axis=0)

Out[111]: color                         19
          director_name                104
          num_critic_for_reviews        50
          duration                      15
          director_facebook_likes      104
          actor_3_facebook_likes        23
          actor_2_name                  13
          actor_1_facebook_likes         7
          gross                        884
          genres                         0
          actor_1_name                   7
          movie_title                    0
          num_voted_users                0
          cast_total_facebook_likes      0
          actor_3_name                  23
          facenumber_in_poster          13
          plot_keywords                153
          movie_imdb_link                0
          num_user_for_reviews          21
          language                      12
          country                        5
          content_rating               303
          budget                       492
          title_year                   108
          actor_2_facebook_likes        13
          imdb_score                     0
          aspect_ratio                 329
          movie_facebook_likes           0
          dtype: int64
```



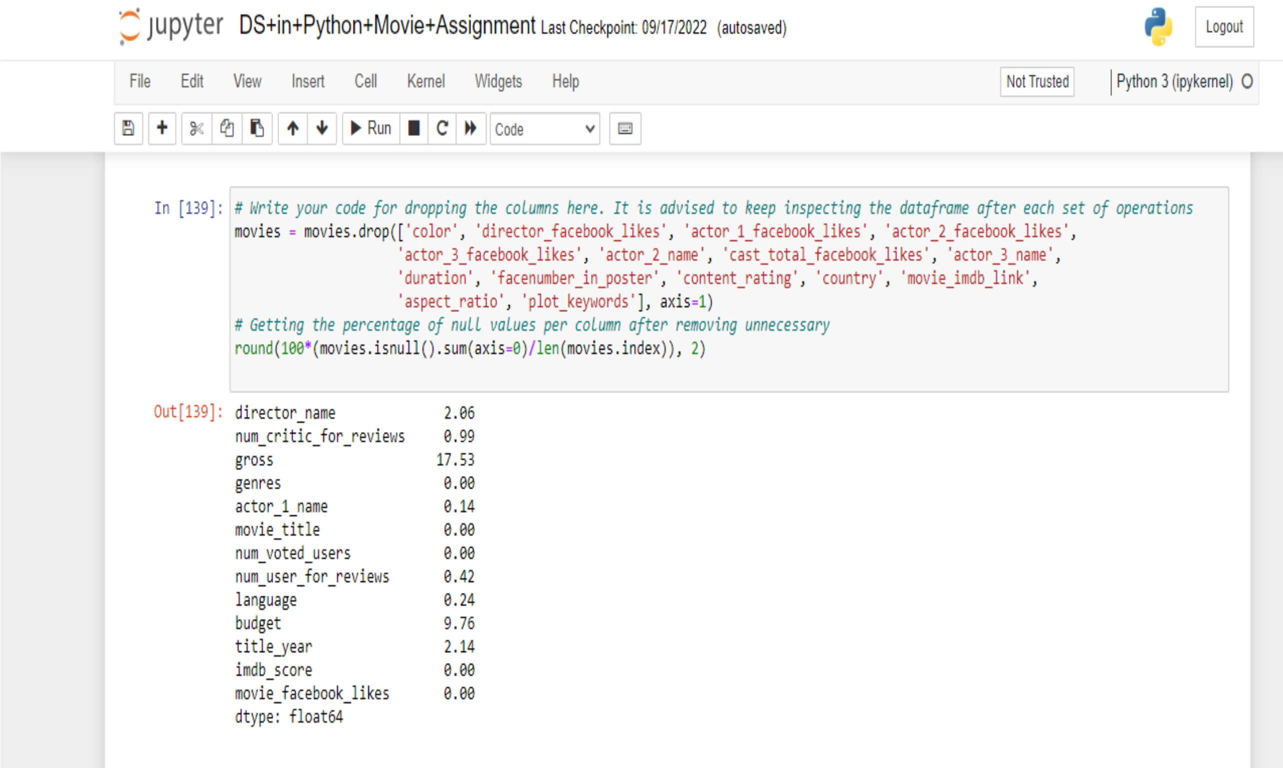
```
In [112]: # Write your code for row-wise null count here
          # axis=1 for row-wise resul
          movies.isnull().sum(axis=1)

Out[112]: 0        0
          1        0
          2        0
          3        0
          4       14
                  ..
          5038     4
          5039     5
          5040     4
          5041     2
          5042     0
          Length: 5043, dtype: int64
```



JioNet
DS+in+Python+Movie+Assignm
localhost:8888/notebooks/Downloads/DS%2Bin%2BPython%2BMovie%2BAssignment.ipynb
jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

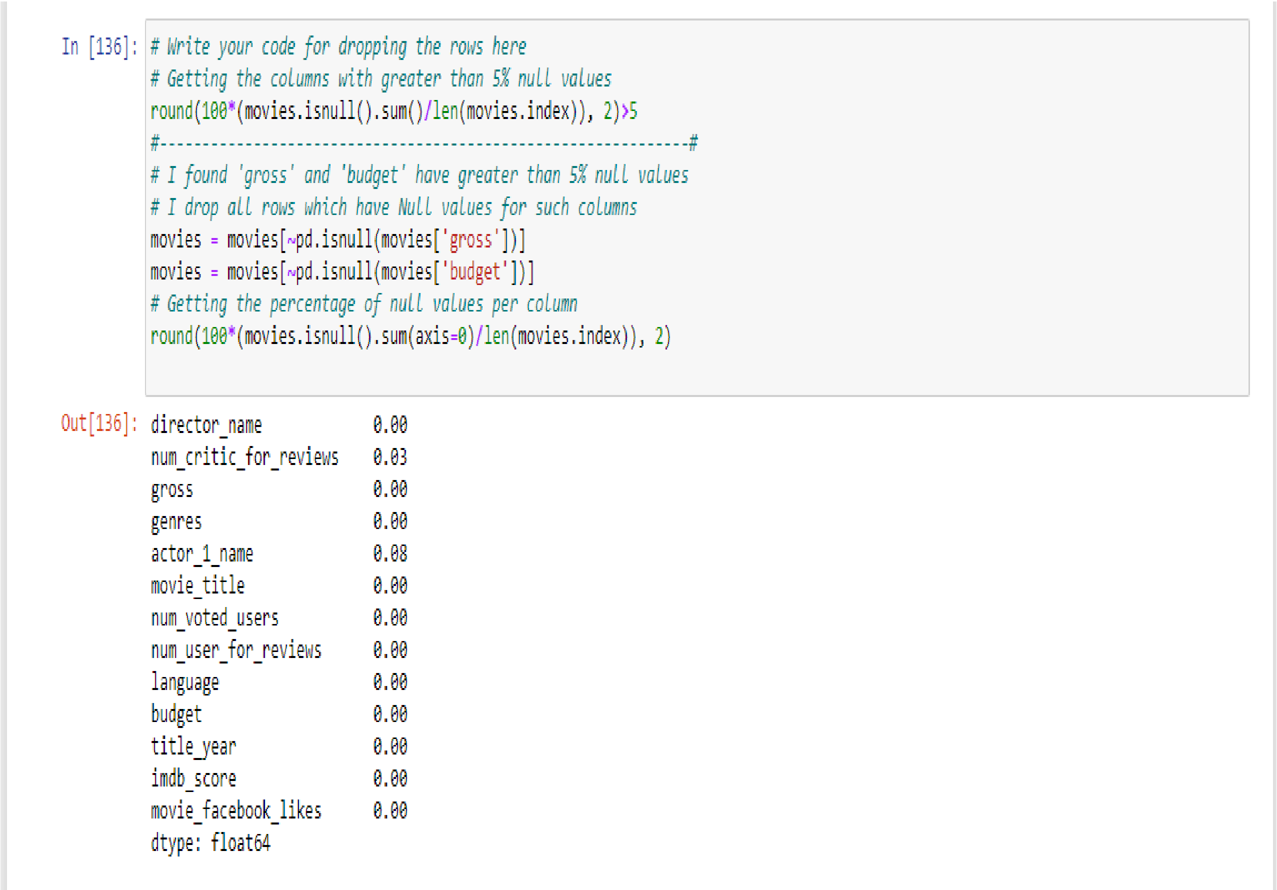
```
In [113]: # Write your code for column-wise null percentages here
          # axis=0 for column-wise and len(movies.index) gives total number of rows
          round(100*(movies.isnull().sum(axis=0)/len(movies.index)), 2)

Out[113]: color                         0.38
          director_name                 2.06
          num_critic_for_reviews        0.99
          duration                      0.30
          director_facebook_likes       2.06
          actor_3_facebook_likes        0.46
          actor_2_name                  0.26
          actor_1_facebook_likes        0.14
          gross                        17.53
          genres                        0.00
          actor_1_name                  0.14
          movie_title                   0.00
          num_voted_users               0.00
          cast_total_facebook_likes     0.00
          actor_3_name                  0.46
          facenumber_in_poster          0.26
          plot_keywords                 3.03
          movie_imdb_link               0.00
          num_user_for_reviews          0.42
          language                      0.24
          country                       0.10
          content_rating                6.01
          budget                        9.76
          title_year                    2.14
          actor_2_facebook_likes        0.26
          imdb_score                    0.00
          aspect_ratio                  6.52
          movie_facebook_likes          0.00
          dtype: float64
```

Dropping Unecessary columns


jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [139]: # Write your code for dropping the columns here. It is advised to keep inspecting the dataframe after each set of operations
          movies = movies.drop(['color', 'director_facebook_likes', 'actor_1_facebook_likes', 'actor_2_facebook_likes',
                                'actor_3_facebook_likes', 'actor_2_name', 'cast_total_facebook_likes', 'actor_3_name',
                                'duration', 'facenumber_in_poster', 'content_rating', 'country', 'movie_imdb_link',
                                'aspect_ratio', 'plot_keywords'], axis=1)
          # Getting the percentage of null values per column after removing unnecessary
          round(100*(movies.isnull().sum(axis=0)/len(movies.index)), 2)
```

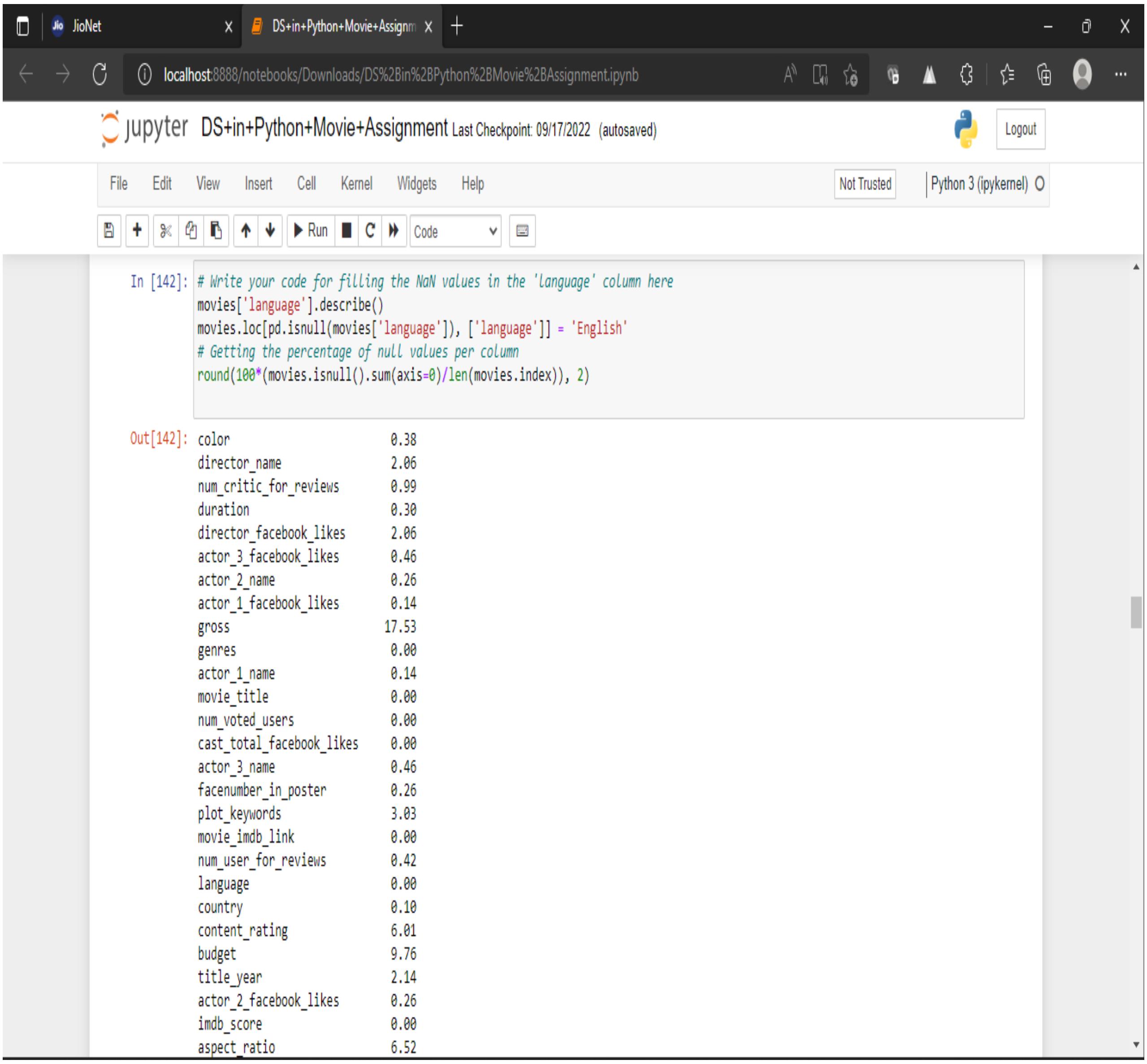
```
Out[139]: director_name              2.06
          num_critic_for_reviews     0.99
          gross                     17.53
          genres                     0.00
          actor_1_name               0.14
          movie_title                0.00
          num_voted_users            0.00
          num_user_for_reviews       0.42
          language                   0.24
          budget                     9.76
          title_year                 2.14
          imdb_score                 0.00
          movie_facebook_likes       0.00
          dtype: float64
```



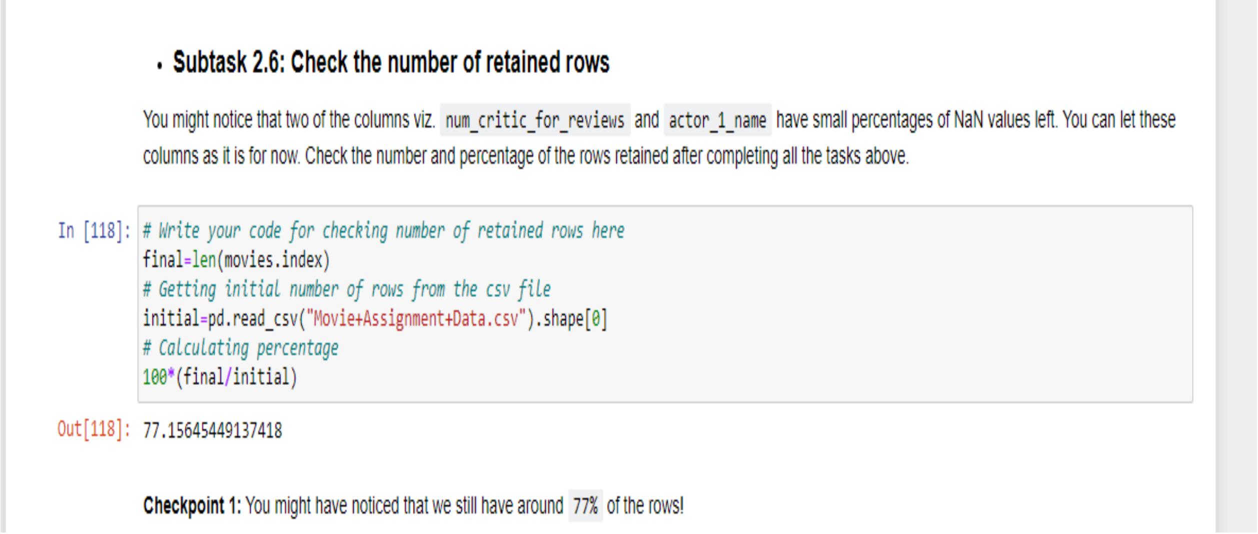
```
In [136]: # Write your code for dropping the rows here
          # Getting the columns with greater than 5% null values
          round(100*(movies.isnull().sum()/len(movies.index)), 2)>5
          #-------------------------------------------------------------#
          # I found 'gross' and 'budget' have greater than 5% null values
          # I drop all rows which have Null values for such columns
          movies = movies[~pd.isnull(movies['gross'])]
          movies = movies[~pd.isnull(movies['budget'])]
          # Getting the percentage of null values per column
          round(100*(movies.isnull().sum(axis=0)/len(movies.index)), 2)
```

```
Out[136]: director_name             0.00
          num_critic_for_reviews    0.03
          gross                     0.00
          genres                    0.00
          actor_1_name              0.08
          movie_title               0.00
          num_voted_users           0.00
          num_user_for_reviews      0.00
          language                  0.00
          budget                    0.00
          title_year                0.00
          imdb_score                0.00
          movie_facebook_likes      0.00
          dtype: float64
```

Filling NaN Values


JioNet
DS+in+Python+Movie+Assignm
localhost:8888/notebooks/Downloads/DS%2Bin%2BPython%2BMovie%2BAssignment.ipynb
jupyter
DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [142]: # Write your code for filling the NaN values in the 'language' column here
          movies['language'].describe()
          movies.loc[pd.isnull(movies['language']), ['language']] = 'English'
          # Getting the percentage of null values per column
          round(100*(movies.isnull().sum(axis=0)/len(movies.index)), 2)

Out[142]: color                       0.38
          director_name               2.06
          num_critic_for_reviews      0.99
          duration                    0.30
          director_facebook_likes     2.06
          actor_3_facebook_likes      0.46
          actor_2_name                0.26
          actor_1_facebook_likes      0.14
          gross                      17.53
          genres                      0.00
          actor_1_name                0.14
          movie_title                 0.00
          num_voted_users             0.00
          cast_total_facebook_likes   0.00
          actor_3_name                0.46
          facenumber_in_poster        0.26
          plot_keywords               3.03
          movie_imdb_link             0.00
          num_user_for_reviews        0.42
          language                    0.00
          country                     0.10
          content_rating              6.01
          budget                      9.76
          title_year                  2.14
          actor_2_facebook_likes      0.26
          imdb_score                  0.00
          aspect_ratio                6.52
```



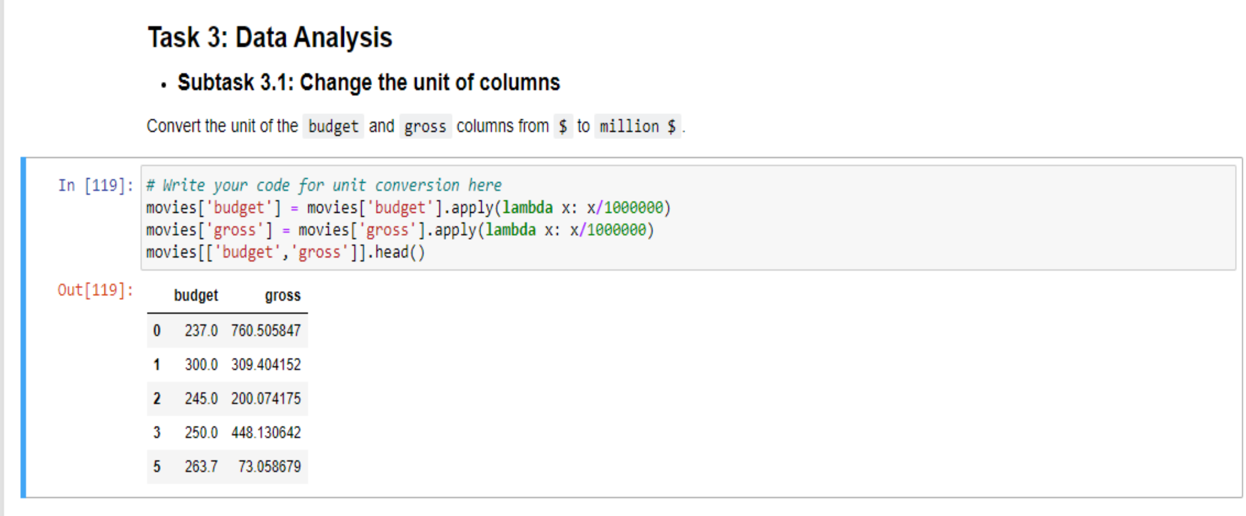
- **Subtask 2.6: Check the number of retained rows**

You might notice that two of the columns viz. `num_critic_for_reviews` and `actor_1_name` have small percentages of NaN values left. You can let these columns as it is for now. Check the number and percentage of the rows retained after completing all the tasks above.

```
In [118]: # Write your code for checking number of retained rows here
          final=len(movies.index)
          # Getting initial number of rows from the csv file
          initial=pd.read_csv("Movie+Assignment+Data.csv").shape[0]
          # Calculating percentage
          100*(final/initial)

Out[118]: 77.15645449137418
```

Checkpoint 1: You might have noticed that we still have around `77%` of the rows!

Task 3- Analysis of Data



Task 3: Data Analysis

- **Subtask 3.1: Change the unit of columns**

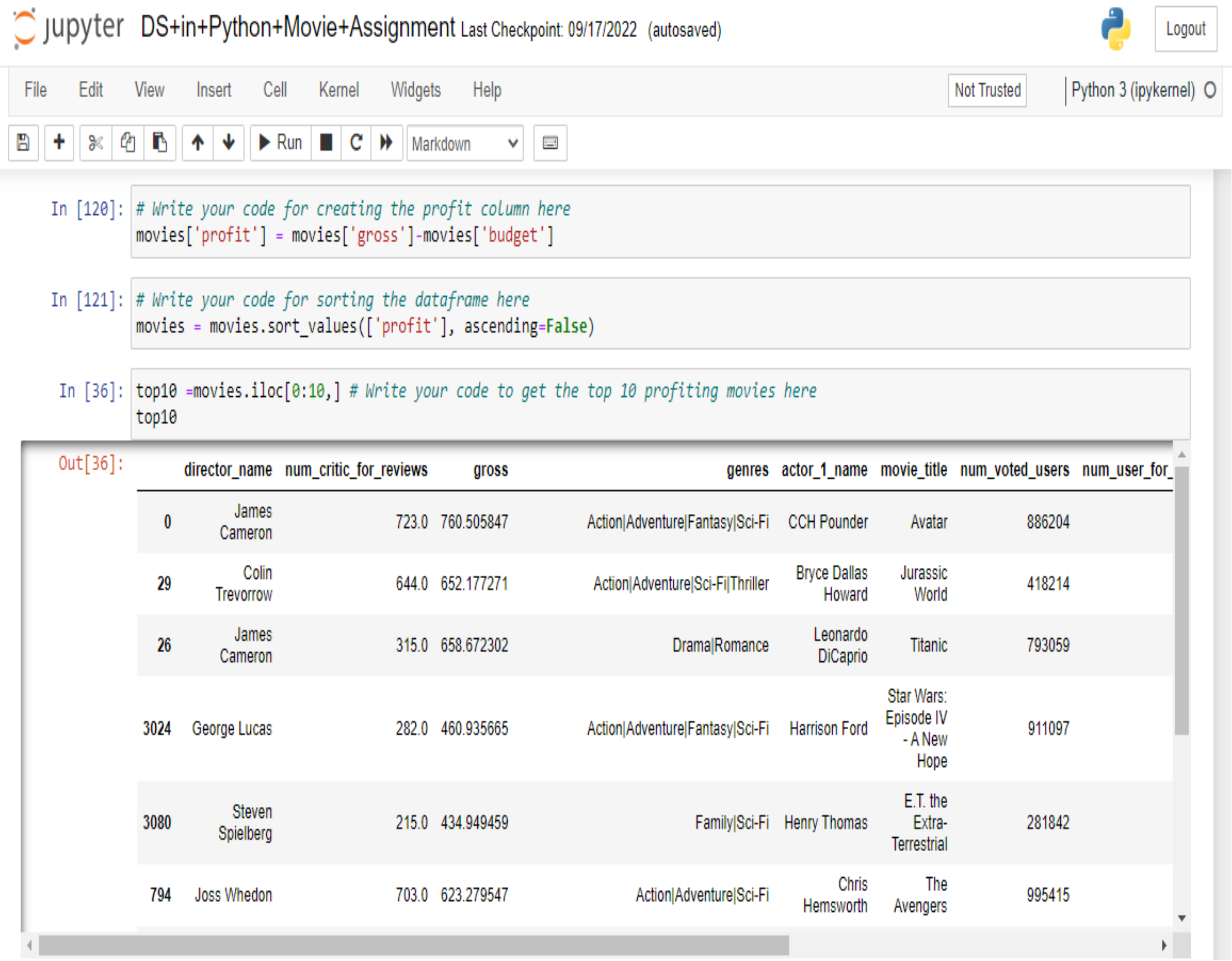
Convert the unit of the `budget` and `gross` columns from `$` to `million $`.

In [119]:
```
# Write your code for unit conversion here
movies['budget'] = movies['budget'].apply(lambda x: x/1000000)
movies['gross'] = movies['gross'].apply(lambda x: x/1000000)
movies[['budget','gross']].head()
```

Out[119]:

	budget	gross
0	237.0	760.505847
1	300.0	309.404152
2	245.0	200.074175
3	250.0	448.130642
5	263.7	73.058679

Finding the movies with highest Profit



jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

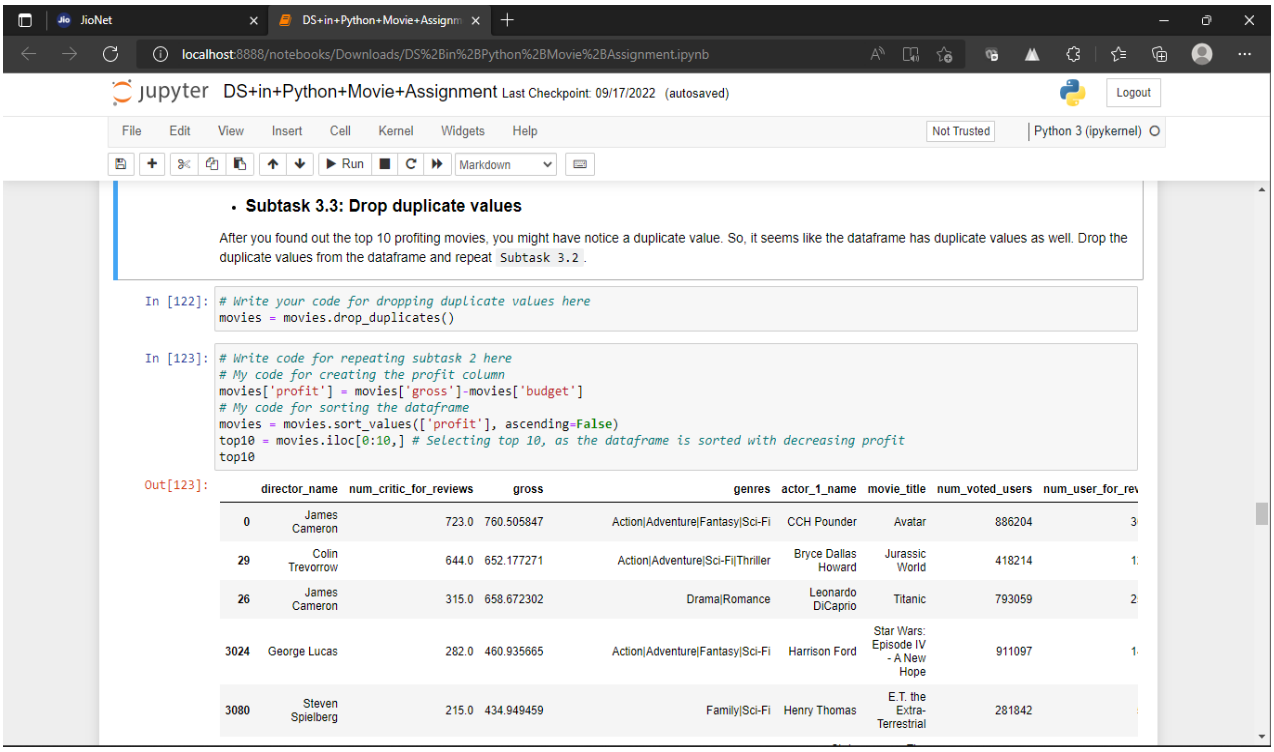
In [120]:
```
# Write your code for creating the profit column here
movies['profit'] = movies['gross']-movies['budget']
```

In [121]:
```
# Write your code for sorting the dataframe here
movies = movies.sort_values(['profit'], ascending=False)
```

In [36]:
```
top10 =movies.iloc[0:10,] # Write your code to get the top 10 profiting movies here
top10
```

Out[36]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	


JioNet
DS+in+Python+Movie+Assignm
localhost:8888/notebooks/Downloads/DS%2Bin%2BPython%2BMovie%2BAssignment.ipynb
jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

- **Subtask 3.3: Drop duplicate values**

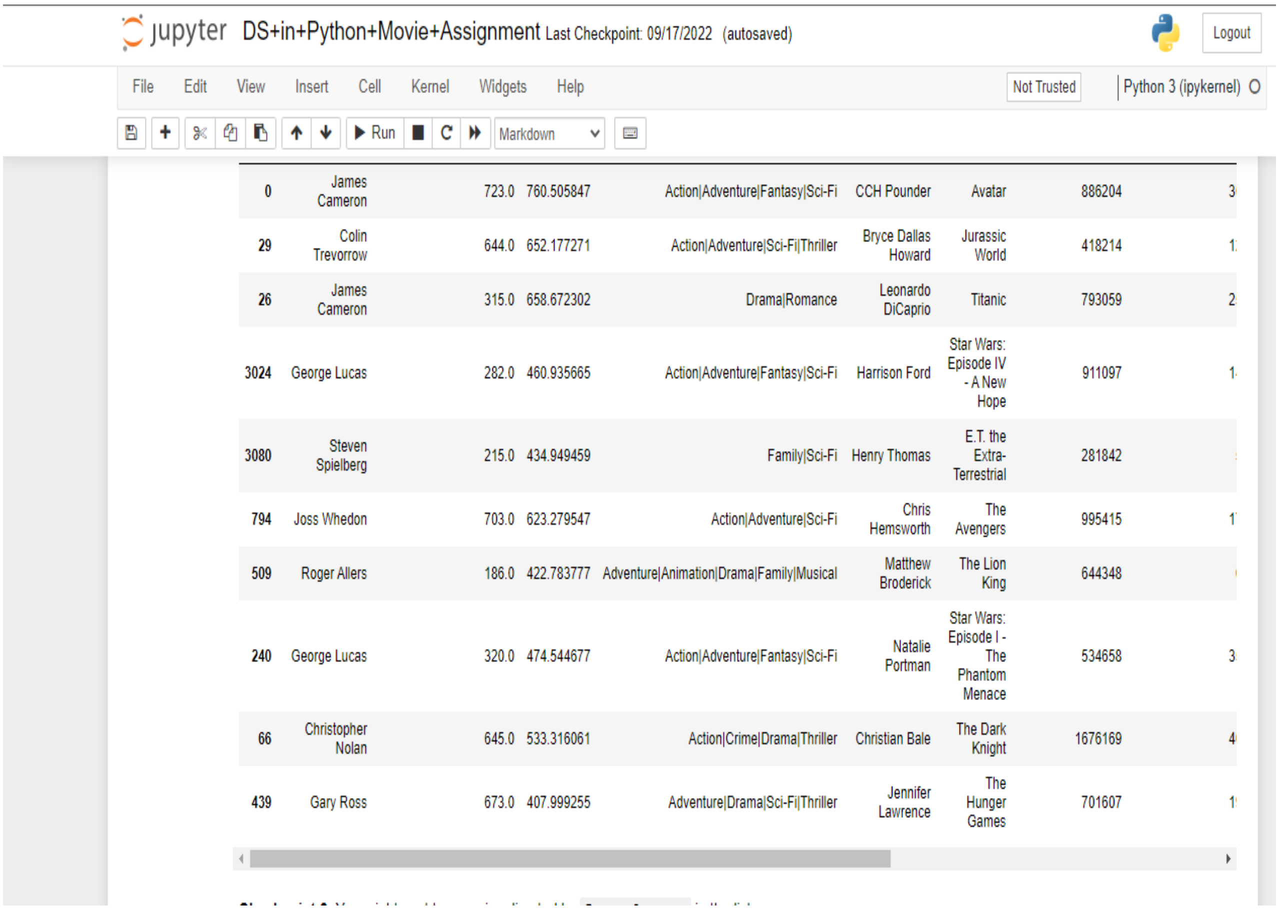
After you found out the top 10 profiting movies, you might have notice a duplicate value. So, it seems like the dataframe has duplicate values as well. Drop the duplicate values from the dataframe and repeat `Subtask 3.2` .

```
In [122]: # Write your code for dropping duplicate values here
          movies = movies.drop_duplicates()
```

```
In [123]: # Write code for repeating subtask 2 here
          # My code for creating the profit column
          movies['profit'] = movies['gross']-movies['budget']
          # My code for sorting the dataframe
          movies = movies.sort_values(['profit'], ascending=False)
          top10 = movies.iloc[0:10,] # Selecting top 10, as the dataframe is sorted with decreasing profit
          top10
```

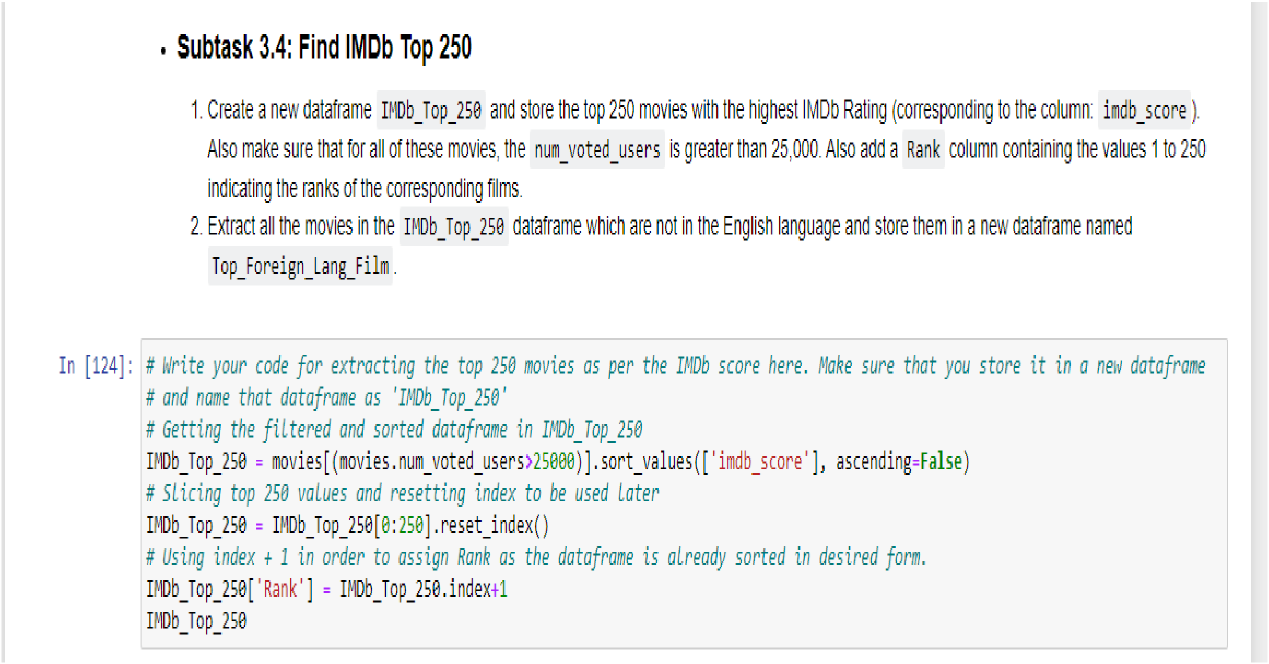
Out[123]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_rev
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	1
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	2
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	1
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	


jupyter DS+in+Python+Movie+Assignment Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	1
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	2
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	1
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	
794	Joss Whedon	703.0	623.279547	Action|Adventure|Sci-Fi	Chris Hemsworth	The Avengers	995415	1
509	Roger Allers	186.0	422.783777	Adventure|Animation|Drama|Family|Musical	Matthew Broderick	The Lion King	644348	
240	George Lucas	320.0	474.544677	Action|Adventure|Fantasy|Sci-Fi	Natalie Portman	Star Wars: Episode I - The Phantom Menace	534658	3
66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	4
439	Gary Ross	673.0	407.999255	Adventure|Drama|Sci-Fi|Thriller	Jennifer Lawrence	The Hunger Games	701607	1

Finding IMDB Top 250

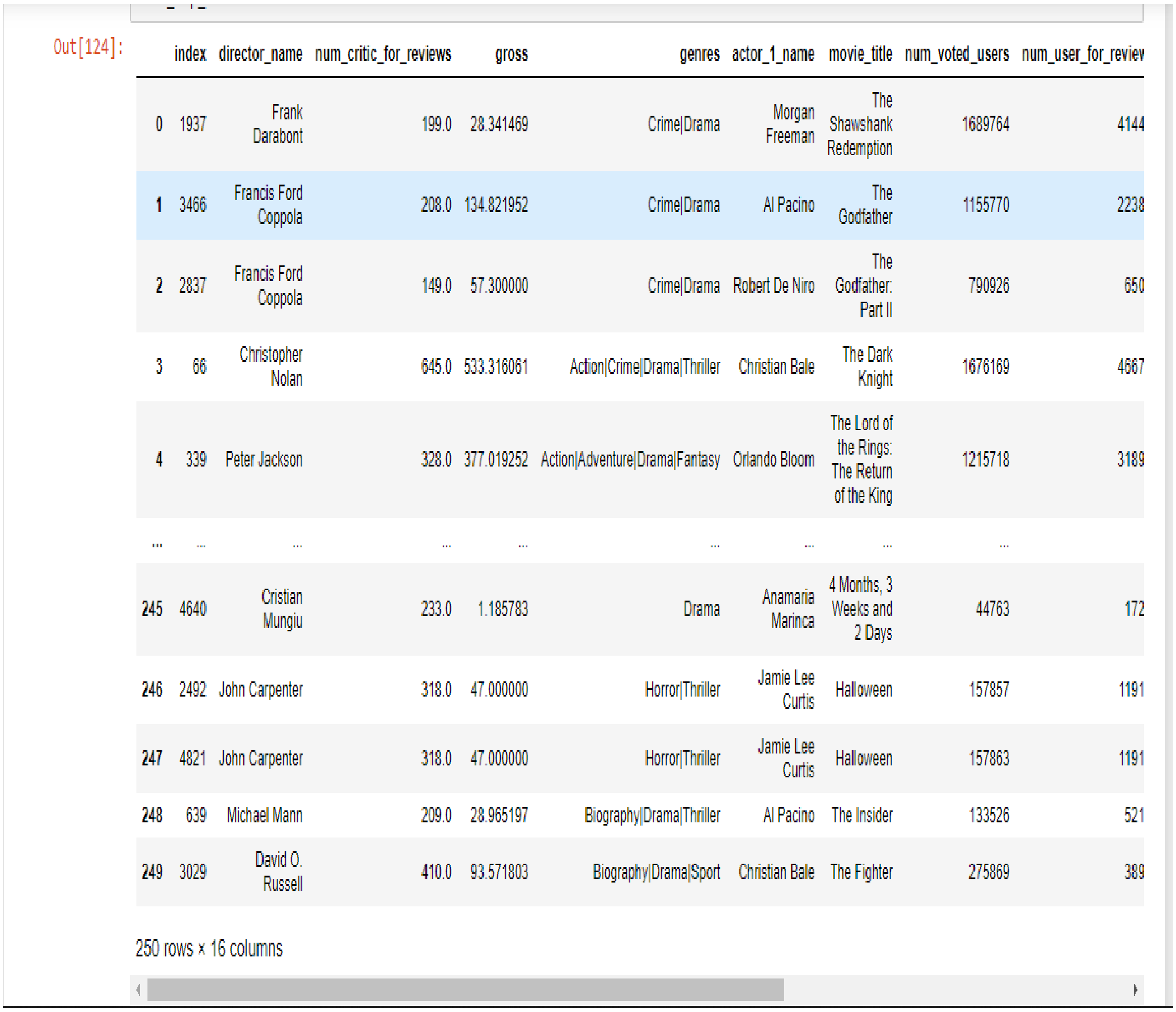


- **Subtask 3.4: Find IMDb Top 250**

1. Create a new dataframe `IMDb_Top_250` and store the top 250 movies with the highest IMDb Rating (corresponding to the column: `imdb_score`). Also make sure that for all of these movies, the `num_voted_users` is greater than 25,000. Also add a `Rank` column containing the values 1 to 250 indicating the ranks of the corresponding films.
2. Extract all the movies in the `IMDb_Top_250` dataframe which are not in the English language and store them in a new dataframe named `Top_Foreign_Lang_Film`.

In [124]:

```
# Write your code for extracting the top 250 movies as per the IMDb score here. Make sure that you store it in a new dataframe
# and name that dataframe as 'IMDb_Top_250'
# Getting the filtered and sorted dataframe in IMDb_Top_250
IMDb_Top_250 = movies[(movies.num_voted_users>25000)].sort_values(['imdb_score'], ascending=False)
# Slicing top 250 values and resetting index to be used later
IMDb_Top_250 = IMDb_Top_250[0:250].reset_index()
# Using index + 1 in order to assign Rank as the dataframe is already sorted in desired form.
IMDb_Top_250['Rank'] = IMDb_Top_250.index+1
IMDb_Top_250
```

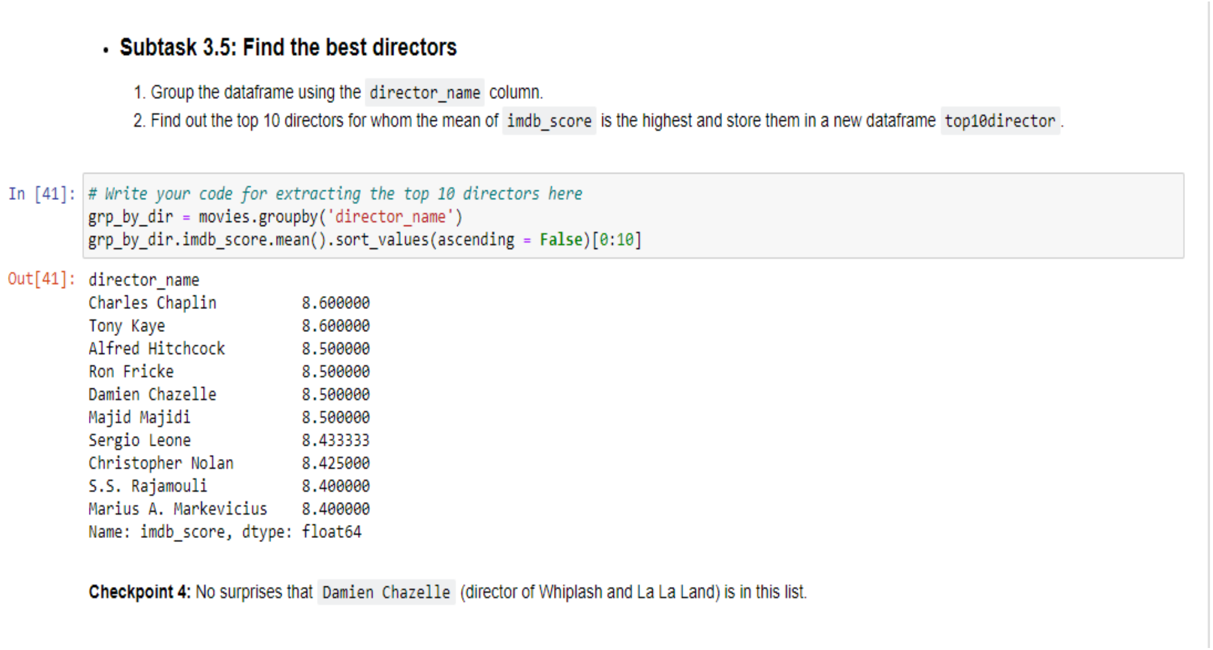


Out[124]:

	index	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviev
0	1937	Frank Darabont	199.0	28.341469	Crime|Drama	Morgan Freeman	The Shawshank Redemption	1689764	4144
1	3466	Francis Ford Coppola	208.0	134.821952	Crime|Drama	Al Pacino	The Godfather	1155770	2238
2	2837	Francis Ford Coppola	149.0	57.300000	Crime|Drama	Robert De Niro	The Godfather: Part II	790926	650
3	66	Christopher Nolan	645.0	533.316061	Action|Crime|Drama|Thriller	Christian Bale	The Dark Knight	1676169	4667
4	339	Peter Jackson	328.0	377.019252	Action|Adventure|Drama|Fantasy	Orlando Bloom	The Lord of the Rings: The Return of the King	1215718	3189
...	...	...	...	...	...	...	...	...	
245	4640	Cristian Mungiu	233.0	1.185783	Drama	Anamaria Marinca	4 Months, 3 Weeks and 2 Days	44763	172
246	2492	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157857	1191
247	4821	John Carpenter	318.0	47.000000	Horror|Thriller	Jamie Lee Curtis	Halloween	157863	1191
248	639	Michael Mann	209.0	28.965197	Biography|Drama|Thriller	Al Pacino	The Insider	133526	521
249	3029	David O. Russell	410.0	93.571803	Biography|Drama|Sport	Christian Bale	The Fighter	275869	389

250 rows × 16 columns

Finding the best Directors



- **Subtask 3.5: Find the best directors**

1. Group the dataframe using the `director_name` column.
2. Find out the top 10 directors for whom the mean of `imdb_score` is the highest and store them in a new dataframe `top10director`.

In [41]:

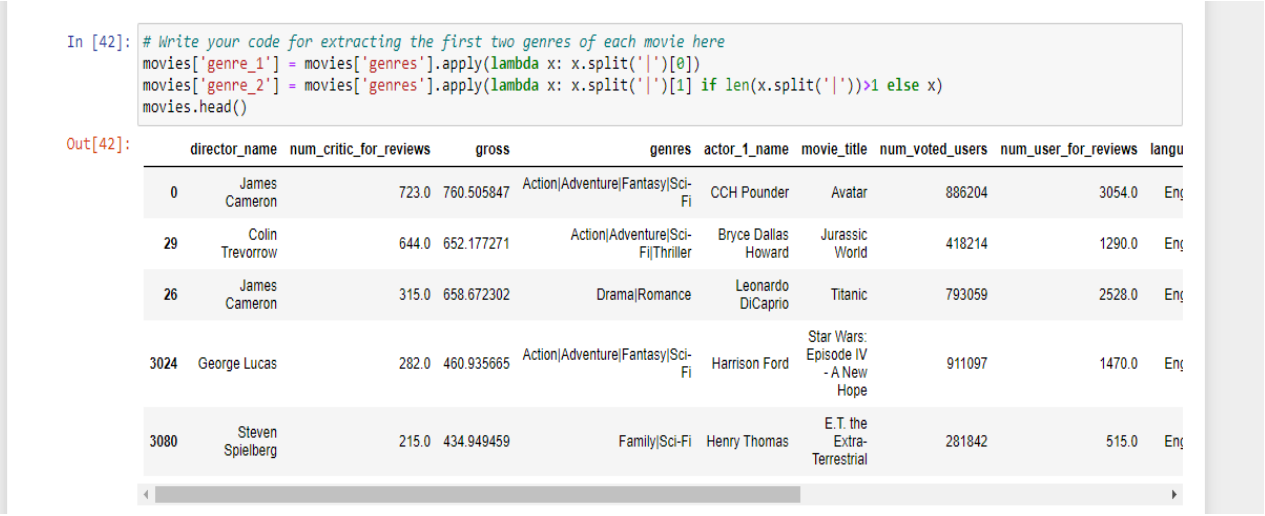
```
# Write your code for extracting the top 10 directors here
grp_by_dir = movies.groupby('director_name')
grp_by_dir.imdb_score.mean().sort_values(ascending = False)[0:10]
```

Out[41]:

```
director_name
Charles Chaplin         8.600000
Tony Kaye               8.600000
Alfred Hitchcock        8.500000
Ron Fricke              8.500000
Damien Chazelle         8.500000
Majid Majidi            8.500000
Sergio Leone            8.433333
Christopher Nolan       8.425000
S.S. Rajamouli          8.400000
Marius A. Markevicius   8.400000
Name: imdb_score, dtype: float64
```

Checkpoint 4: No surprises that `Damien Chazelle` (director of Whiplash and La La Land) is in this list.

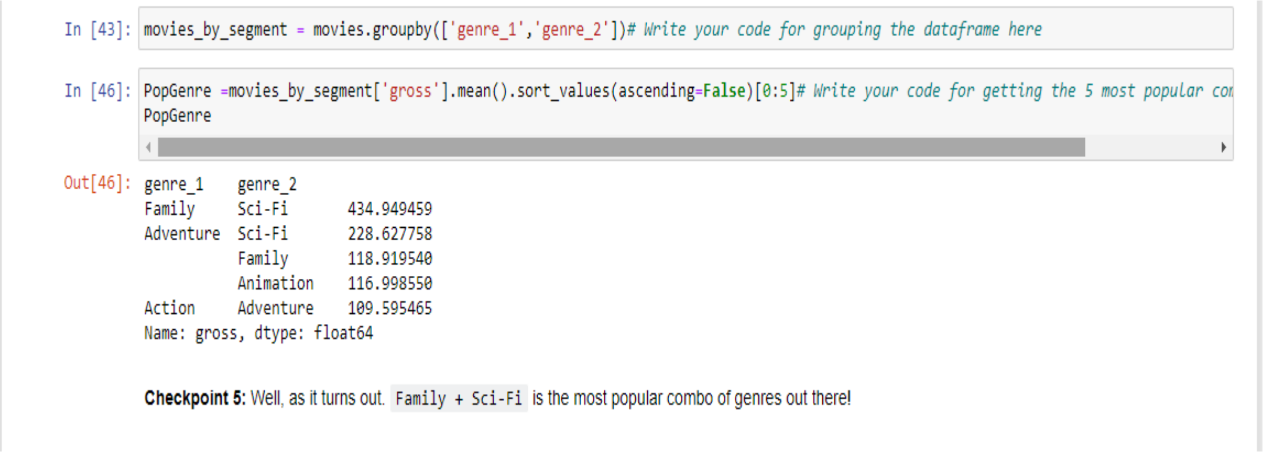
Finding Popular Genres



```
In [42]: # Write your code for extracting the first two genres of each movie here
         movies['genre_1'] = movies['genres'].apply(lambda x: x.split('|')[0])
         movies['genre_2'] = movies['genres'].apply(lambda x: x.split('|')[1] if len(x.split('|'))>1 else x)
         movies.head()
```

Out[42]:

	director_name	num_critic_for_reviews	gross	genres	actor_1_name	movie_title	num_voted_users	num_user_for_reviews	langu
0	James Cameron	723.0	760.505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder	Avatar	886204	3054.0	En
29	Colin Trevorrow	644.0	652.177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas Howard	Jurassic World	418214	1290.0	En
26	James Cameron	315.0	658.672302	Drama|Romance	Leonardo DiCaprio	Titanic	793059	2528.0	En
3024	George Lucas	282.0	460.935665	Action|Adventure|Fantasy|Sci-Fi	Harrison Ford	Star Wars: Episode IV - A New Hope	911097	1470.0	En
3080	Steven Spielberg	215.0	434.949459	Family|Sci-Fi	Henry Thomas	E.T. the Extra-Terrestrial	281842	515.0	En



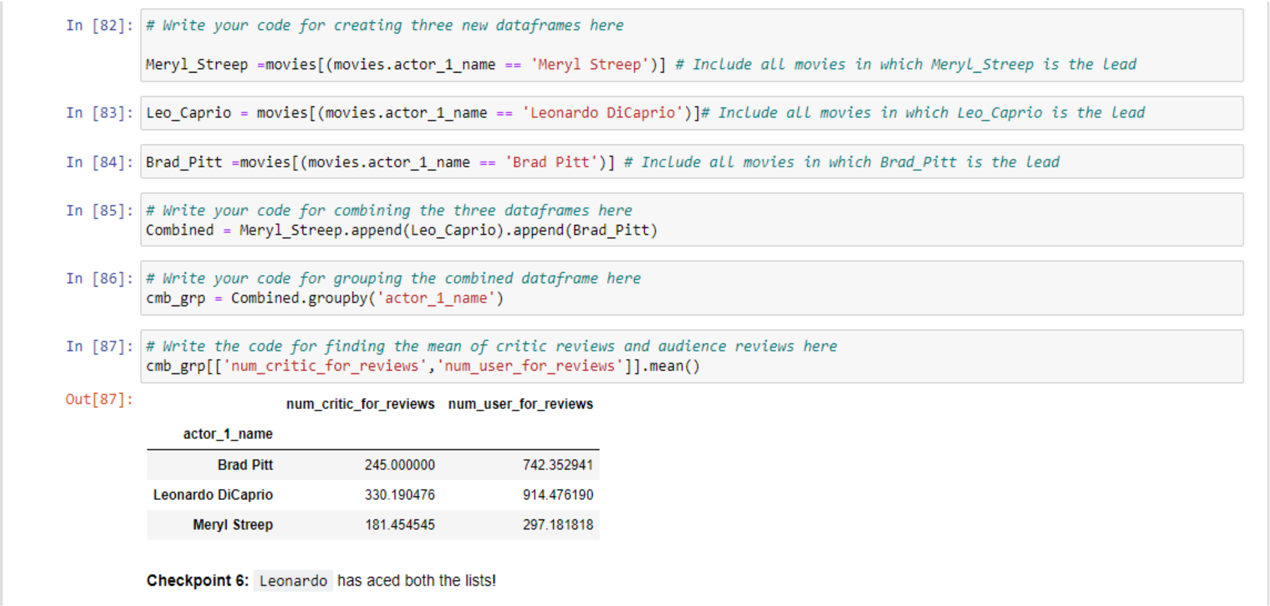
```
In [43]: movies_by_segment = movies.groupby(['genre_1','genre_2'])# Write your code for grouping the dataframe here
```

```
In [46]: PopGenre =movies_by_segment['gross'].mean().sort_values(ascending=False)[0:5]# Write your code for getting the 5 most popular cor
         PopGenre
```

```
Out[46]: genre_1    genre_2
         Family     Sci-Fi       434.949459
         Adventure  Sci-Fi       228.627758
                    Family       118.919540
                    Animation    116.998550
         Action     Adventure    109.595465
         Name: gross, dtype: float64
```

Checkpoint 5: Well, as it turns out. `Family + Sci-Fi` is the most popular combo of genres out there!

Finding the Critic-Favourite and Audience-Favourite actors



```
In [82]: # Write your code for creating three new dataframes here

         Meryl_Streep =movies[(movies.actor_1_name == 'Meryl Streep')] # Include all movies in which Meryl_Streep is the lead
```

```
In [83]: Leo_Caprio = movies[(movies.actor_1_name == 'Leonardo DiCaprio')]# Include all movies in which Leo_Caprio is the lead
```

```
In [84]: Brad_Pitt =movies[(movies.actor_1_name == 'Brad Pitt')] # Include all movies in which Brad_Pitt is the lead
```

```
In [85]: # Write your code for combining the three dataframes here
         Combined = Meryl_Streep.append(Leo_Caprio).append(Brad_Pitt)
```

```
In [86]: # Write your code for grouping the combined dataframe here
         cmb_grp = Combined.groupby('actor_1_name')
```

```
In [87]: # Write the code for finding the mean of critic reviews and audience reviews here
         cmb_grp[['num_critic_for_reviews','num_user_for_reviews']].mean()
```

Out[87]:

actor_1_name	num_critic_for_reviews	num_user_for_reviews
Brad Pitt	245.000000	742.352941
Leonardo DiCaprio	330.190476	914.476190
Meryl Streep	181.454545	297.181818

Checkpoint 6: `Leonardo` has aced both the lists!

Conclusion and Future of Data Science

Highly specialized knowledge and skills are necessary, so there's a gap between the supply and demand of data scientists. Everybody who’s in data science recognizes this. If you look to the US, for example, there's a need right now for more than 150,000 data scientists. There's also a global shortage of data science skills in Europe and Asia.

If you’re a company that really needs experts in this field, it can be very difficult because of the complexity of the data and each of the company's specific data practices. So, if you’re a data scientist, you have to be able to handle the technical and be good at communicating, and also, there’s the business aspect of the role—and that's only at the beginning.

It's also interesting to cite research showing that 94 percent of data scientists and graduates have gotten jobs since 2011. Ninety-four percent, so you can feel very comfortable if you're either moving into this direction or you’re already a data scientist that you have very good job potential. This indicates how reliable a career option in data science is now but also moving into the future.

It's a career path that runs parallel to all the digital disruption that's on the horizon. You can see how this is evolving very quickly, capable of growing alongside the changing landscape of technological progress. If we look at this growth in data science right now, it’s also connected to other important factors. You can think about the data increase from IoT or from social data at the edge.

If we look a little bit more ahead, the US Bureau of Labour Statistics predicts that by 2026—so around six years from now—there will be 11.5 million jobs in data science and analytics.

REFERENCES

upGrad | Learning Platform

Data Science With Python Tutorial - GeeksforGeeks

Data Science Tutorial (w3schools.com)

Overview of Data Science - GeeksforGeeks