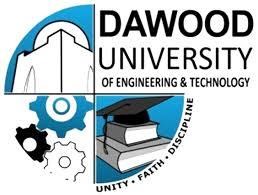
**Artificial Intelligence**

**(Practical Manual)**



**4th Semester, 2nd Year BATCH -2023**

# BS ARTIFICIAL INTELLIGENCE

**DAWOOD UNIVERSITY OF ENGINEERING & TECHNOLOGY, KARACHI**



## CERTIFICATE

This is to certify that Mr./Ms. \_\_\_\_\_\_\_\_\_\_\_Kainat Moin\_\_\_\_\_\_\_with Roll # 23-AI-48 of Batch 2023 has successfully completed all the labs prescribed for the course “Artificial Intelligence”.

**Engr. Hamza Farooqui**

Lecturer

Department of AI

|  |  |
| --- | --- |
| **S. No.** | **Title of Experiment** |
| **1** | Introduction to Programming in Python |
| **2** | Working with NumPy Arrays |
| **3** | Data Manipulation Using Pandas |
| **4** | Implementing Breadth First Search (BFS) |
| **5** | Open Ended Lab - 1 |
| **6** | Implementing Depth First Search (DFS) |
| **7** | Implementing Best First Search (Without Heuristics) |
| **8** | A\* Search Algorithm |
| **9** | Simple Linear Regression |
| **10** | Multivariate Linear Regression |
| **11** | Open Ended Lab – 2 |
| **12** | Binary Classification using Logistic Regression |

## Lab No: 1

**Objective:** To introduce students to **Python programming** and develop their ability to write, understand, and execute basic Python code for data handling and problem solving.

**Why Python?**

* Python is a high-level, interpreted language widely used in AI, data science, and software development.
* It is known for its simple syntax, large community, and rich set of libraries.

**Core Concepts: -**

| **Concept** | **Description** |
| --- | --- |
| Variables & Data Types | int, float, str, bool, list, tuple, dict |
| Operators | Arithmetic (+, -, \*, /), Comparison (==, !=) |
| Control Structures | if, elif, else, for, while |
| Functions | Using def to define reusable code blocks |
| Input/Output | input(), print() |
| Basic Libraries | math, random, datetime, etc. |

**🔹 Simple Example Code**

name = input("Enter your name: ")

print("Hello,", name)

num = int(input("Enter a number: "))

print("Square is:", num \*\* 2)

**Why It Matters in AI:**

* Python is the primary language for AI frameworks like TensorFlow, PyTorch, and scikit-learn.
* Understanding Python is essential for implementing AI algorithms, preprocessing data, and building models.

**Task:**

Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

**Example 1:**

**Input:** haystack = "sadbutsad", needle = "sad"

**Output:** 0

**Explanation:** "sad" occurs at index 0 and 6.

The first occurrence is at index 0, so we return 0.

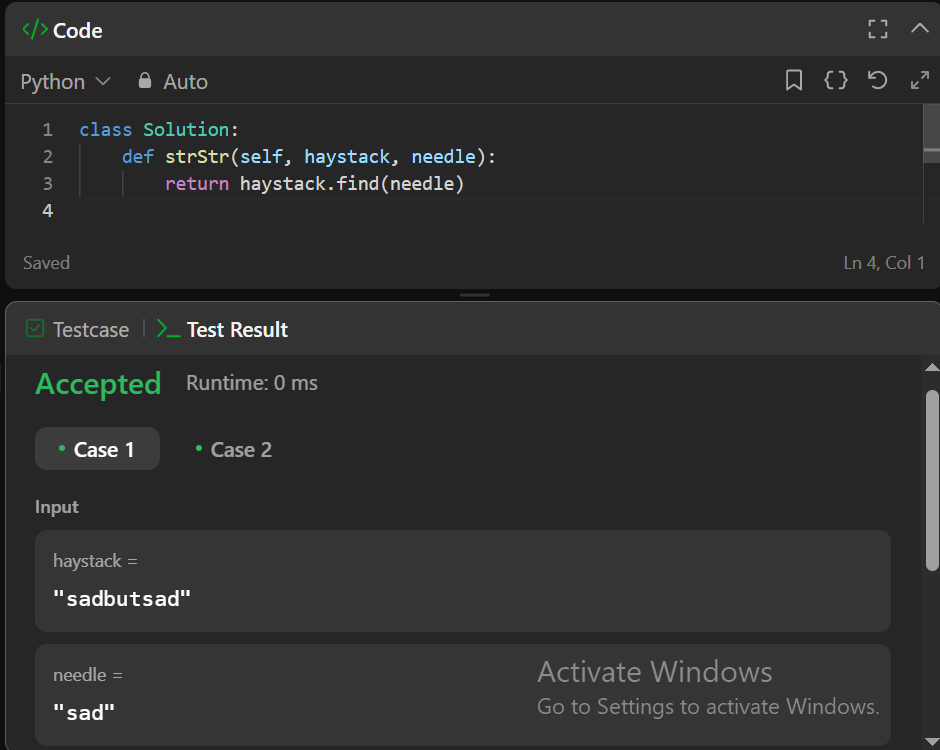
**Example 2:**

**Input:** haystack = "leetcode", needle = "leeto"

**Output:** -1

**Explanation:** "leeto" did not occur in "leetcode", so we return -1.

**Output:**



## Lab No: 2

**Objective:** Write Python program to demonstrate use of **Numpy**

**Practical Significance: -**

Though Python is simple to learn language but it also very strong with its features. As mentioned earlier Python supports various built-in packages. Apart from built-in package user can also make their own packages i.e. User Defined Packages. **Numpy** is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. This practical will allow students to write a code.

**Minimum Theoretical Background: -**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Steps for Installing numpy in windows OS

1. goto Command prompt

2. run command pip install numpy

3. open IDLE Python Interpreter

4. Check numpy is working or not

>>> import numpy

>>> import numpy as np

>>> a=np.array([10,20,30,40,50])

>>> print(a)

[10 20 30 40 50]

### **Example: -**

>>> student=np.dtype([('name','S20'),('age','i1'),('marks','f4')])

>>> a=np.array([('Hamza',43,90),('Asad',38,80)],dtype=student)

>>> print(a)

[('Hamza', 43, 90.) ('Asad', 38, 80.)]

### **Example: -**

>>> print(a)

[10 20 30 40 50 60]

>>> a.shape=(2,3)

>>> print(a) [[10 20 30]

[40 50 60]]

>>> a.shape=(3,2)

>>> print(a) [[10 20]

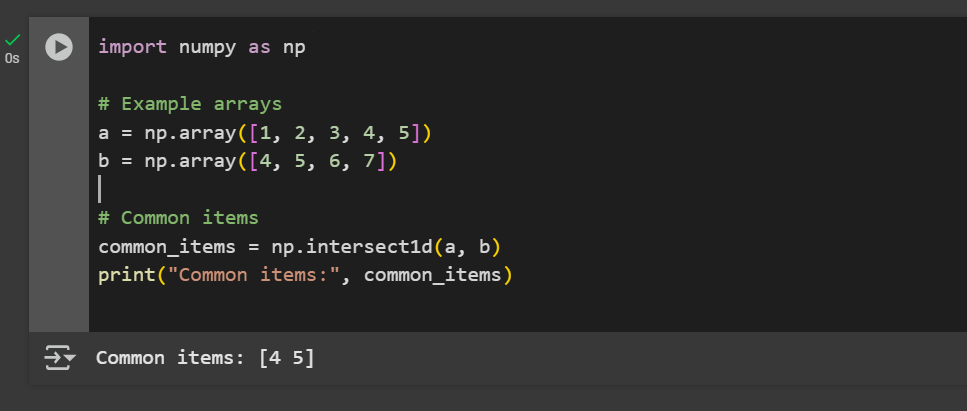
[30 40]

[50 60]]

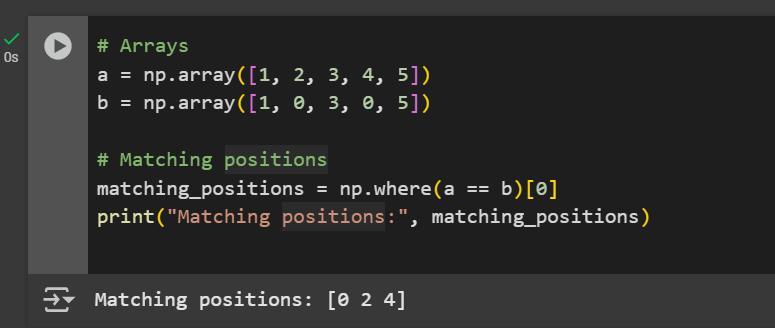
**Tasks: -**

Write Python Code for the following:

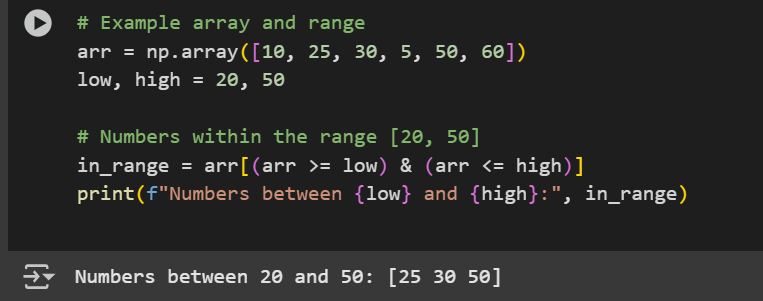
1. How to get the common items between two python numpy arrays?



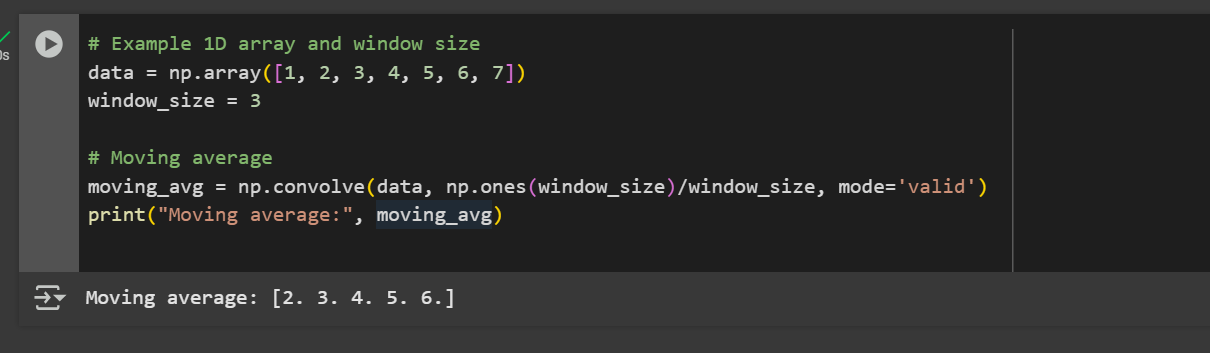
1. How to get the positions where elements of two arrays match?



1. How to extract all numbers between a given range from a numpy array?



1. Implement the moving average for the 1D array in NumPy.



## Lab No: 3

**Objective:** To equip students with the skills to manipulate, clean, analyze, and preprocess structured datasets using the **Pandas library** in Python, preparing data for use in AI models and algorithms.

**Introduction to Pandas: -**

**Pandas** is a powerful Python library used for data manipulation and analysis. It provides two main data structures:

1. **Series** – One-dimensional labeled array.
2. **DataFrame** – Two-dimensional labeled data structure, similar to a table in a database or an Excel sheet.

Pandas is widely used in **AI and Machine Learning** pipelines for preprocessing, analyzing, and cleaning data before feeding it into models.

**Loading Data: -**

You can read structured data from various file formats:

# Load CSV file

df = pd.read\_csv('data.csv')

# Load Excel file

df\_excel = pd.read\_excel('data.xlsx')

# Load from dictionary

data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}

df\_dict = pd.DataFrame(data)

**Exploring Data: -**

df.head() # First 5 rows

df.tail() # Last 5 rows

df.info() # Data types and non-null values

df.describe() # Statistical summary of numeric columns

**Data Selection: -**

df['Age'] # Select a single column

df[['Name', 'Age']] # Select multiple columns

df.iloc[0] # Row by index position

df.loc[0] # Row by index label

**Filtering Data: -**

# Filter rows where Age > 25

df[df['Age'] > 25]

# Filter rows with multiple conditions

df[(df['Age'] > 25) & (df['Gender'] == 'Male')]

**Adding/Modifying Columns: -**

# Add a new column

df['Is\_Adult'] = df['Age'] >= 18

# Modify an existing column

df['Age'] = df['Age'] + 1

**Handling Missing Values: -**

df.isnull().sum() # Count missing values

df.dropna() # Drop rows with any missing values

df.fillna(0) # Fill missing values with 0

df.fillna(df.mean()) # Fill with mean of the column

**Grouping and Aggregation: -**

# Group by Gender and calculate mean age

df.groupby('Gender')['Age'].mean()

# Count entries per category

df['Gender'].value\_counts()

**Sorting and Reordering: -**

df.sort\_values(by='Age', ascending=False) # Sort by Age descending

df.reset\_index(drop=True, inplace=True) # Reset index after sorting

**Dropping Columns and Rows: -**

df.drop(columns=['Is\_Adult'], inplace=True) # Drop a column

df.drop(index=[0], inplace=True) # Drop a row

**Merging and Joining DataFrames: -**

# Merge on a common column

merged\_df = pd.merge(df1, df2, on='ID')

# Concatenate along rows or columns

pd.concat([df1, df2], axis=0) # Row-wise

pd.concat([df1, df2], axis=1) # Column-wise

**Saving Data: -**

df.to\_csv('cleaned\_data.csv', index=False)

df.to\_excel('output.xlsx', index=False)

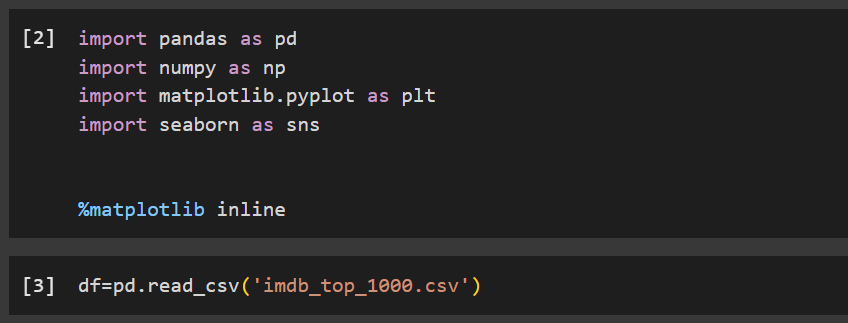
**Why Pandas is Important in AI: -**

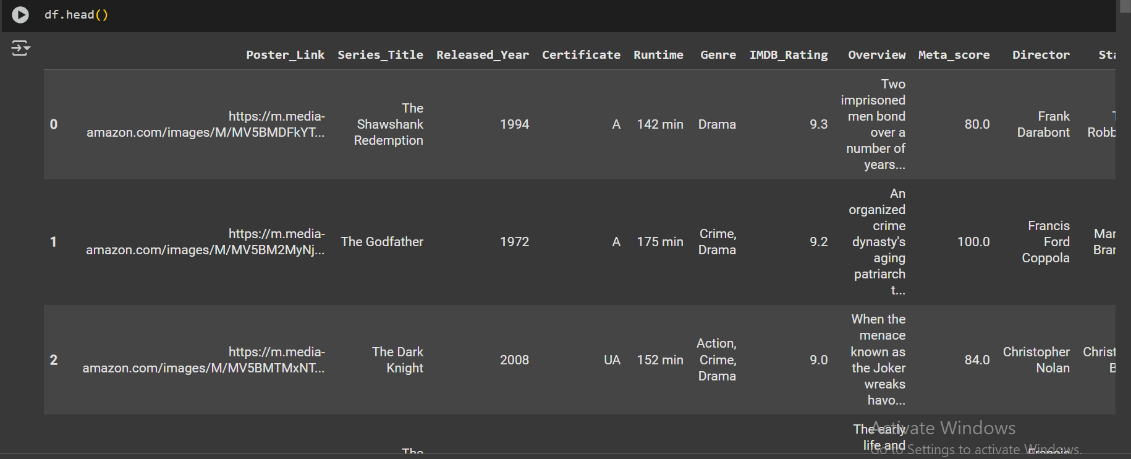
* Prepares raw data for ML models.
* Enables feature engineering.
* Helps detect and handle missing or inconsistent values.
* Supports exploratory data analysis (EDA) and data cleaning.

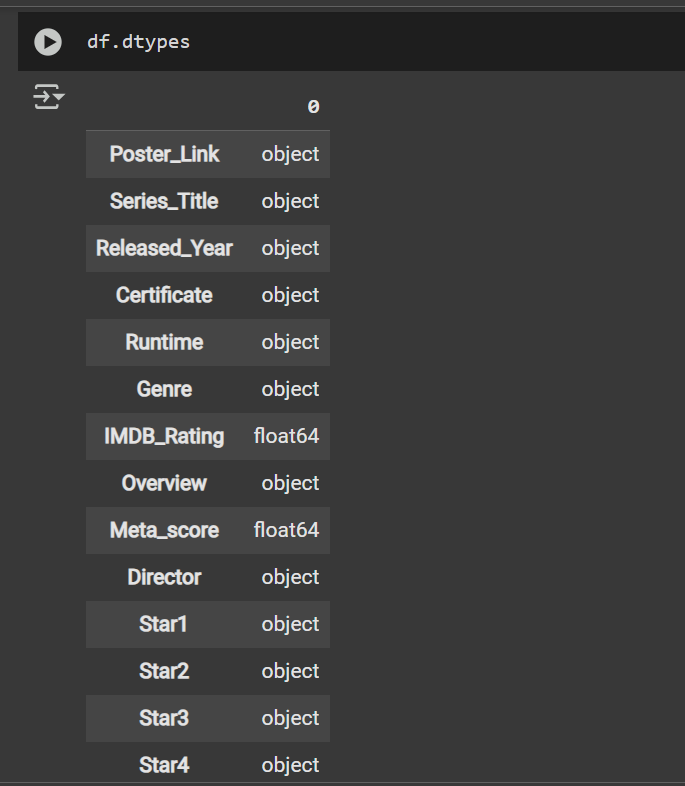
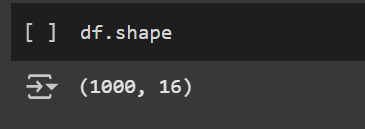
**Tasks: -**

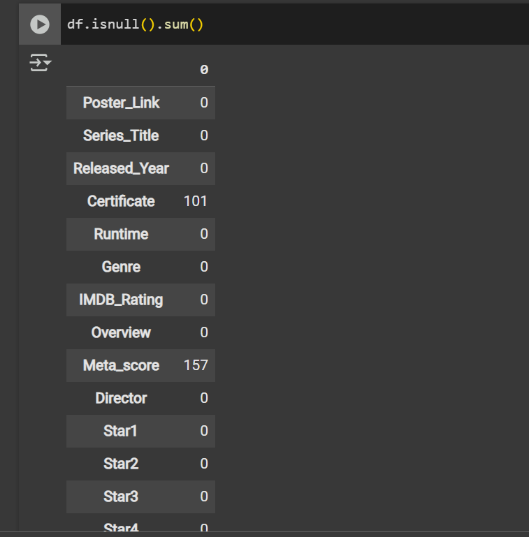
**Kaggle** IMDb Top 1000 Movies dataset

**Task 1: Load and Explore the Dataset**

1. Load the dataset.
2. Display the first 5 rows.

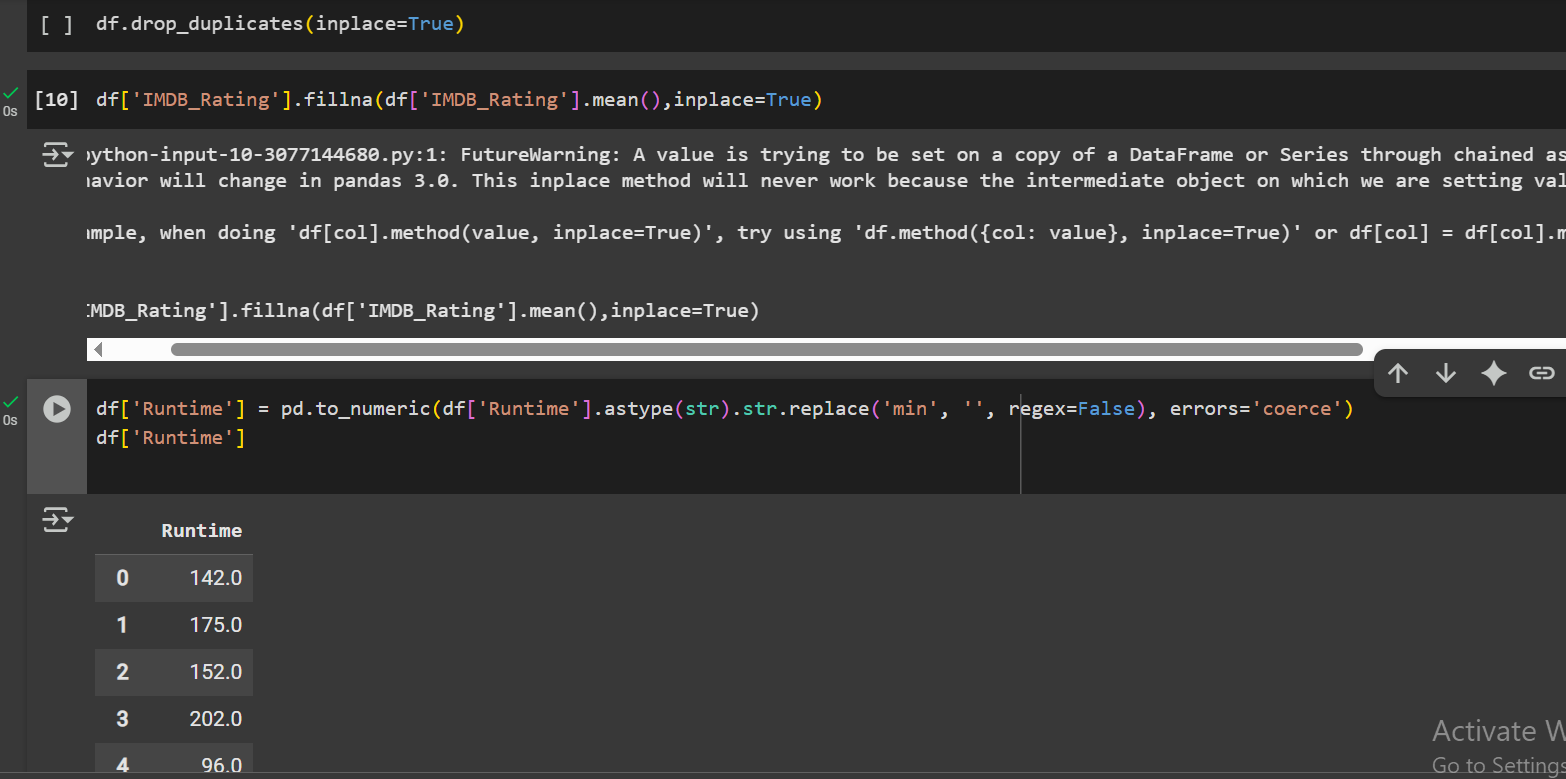


1. Check the data types of each column.
2. Find the number of rows and columns.
3. Check for missing values.



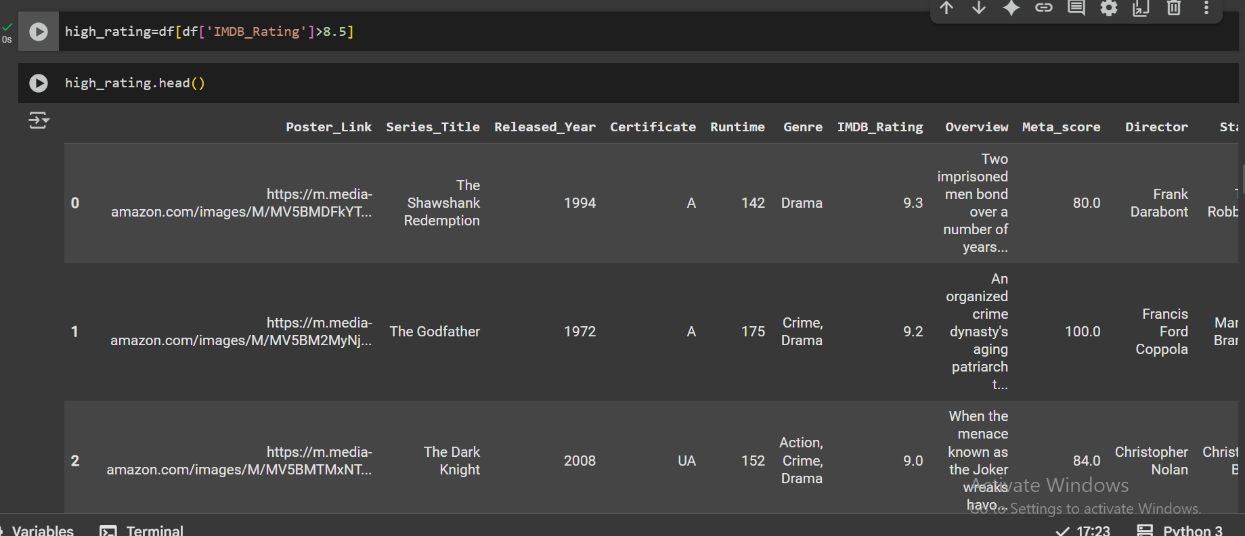
**Task 2: Data Cleaning**

1. Remove any duplicate rows if present.
2. Fill missing values in the dataset (e.g., replace missing ratings with the mean rating).
3. Convert the Runtime column (which is in minutes as a string, e.g., "120 min") to an integer.

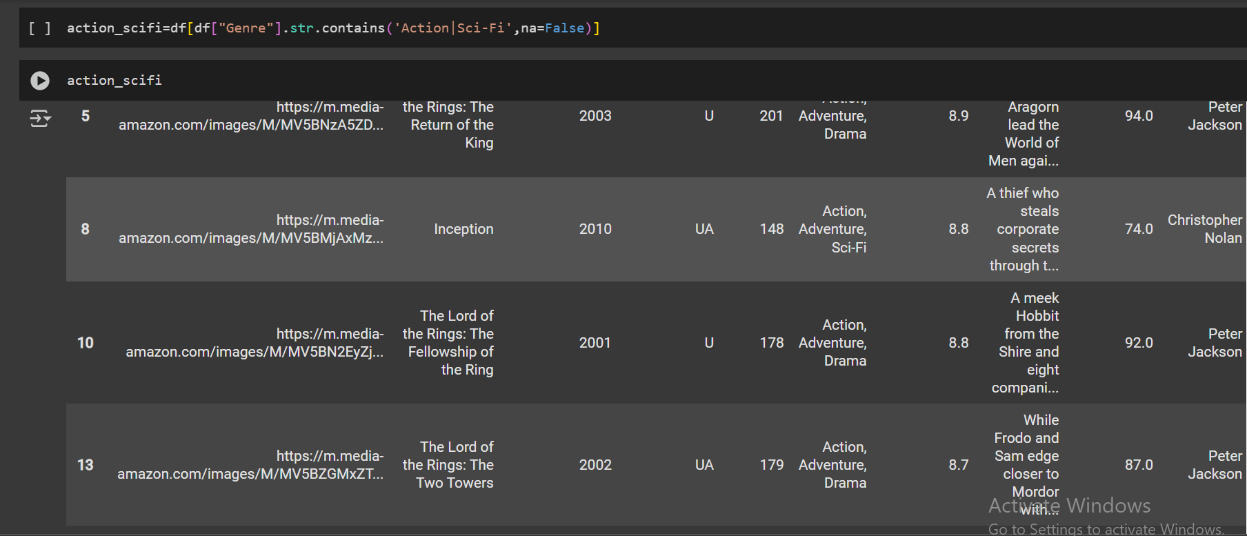


**Task 3: Data Filtering & Sorting**

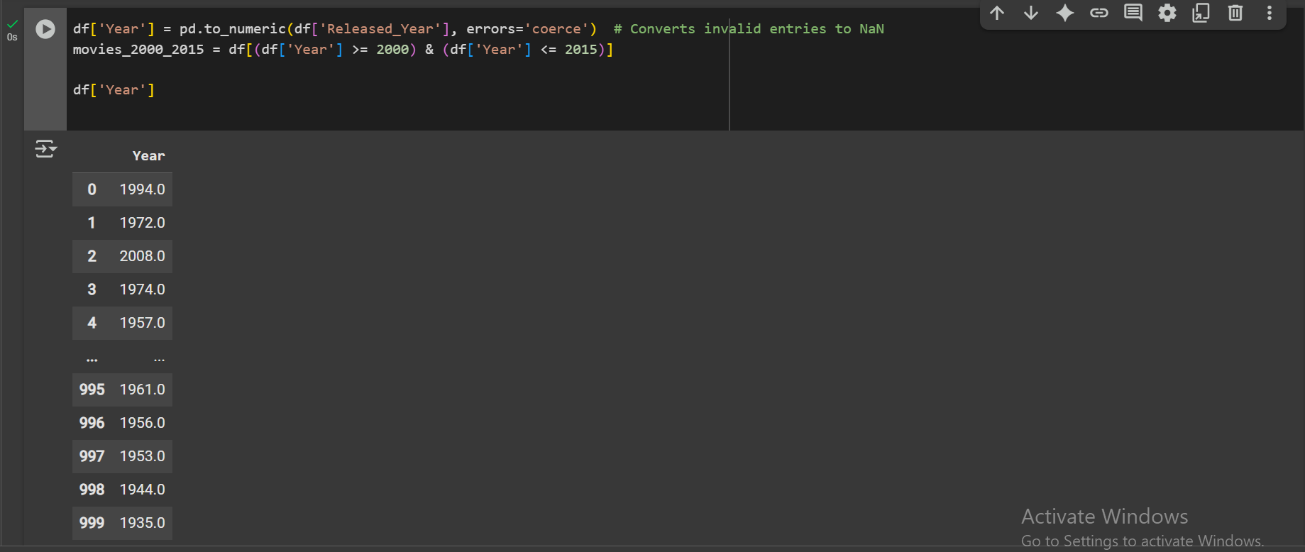
1. Find all movies with an **IMDb rating greater than 8.5**.



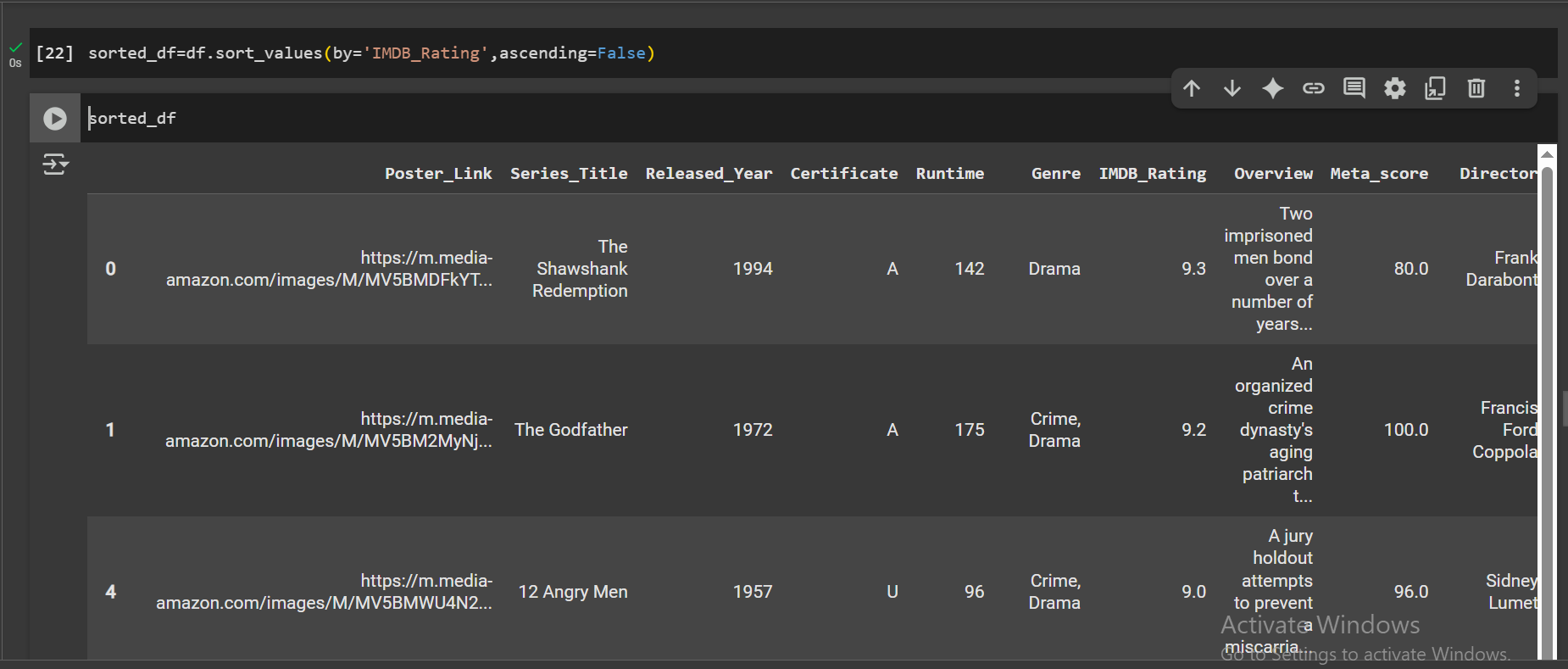
1. List movies that belong to the **Action or Sci-Fi genre**.



1. Find movies that were released **between 2000 and 2015**.

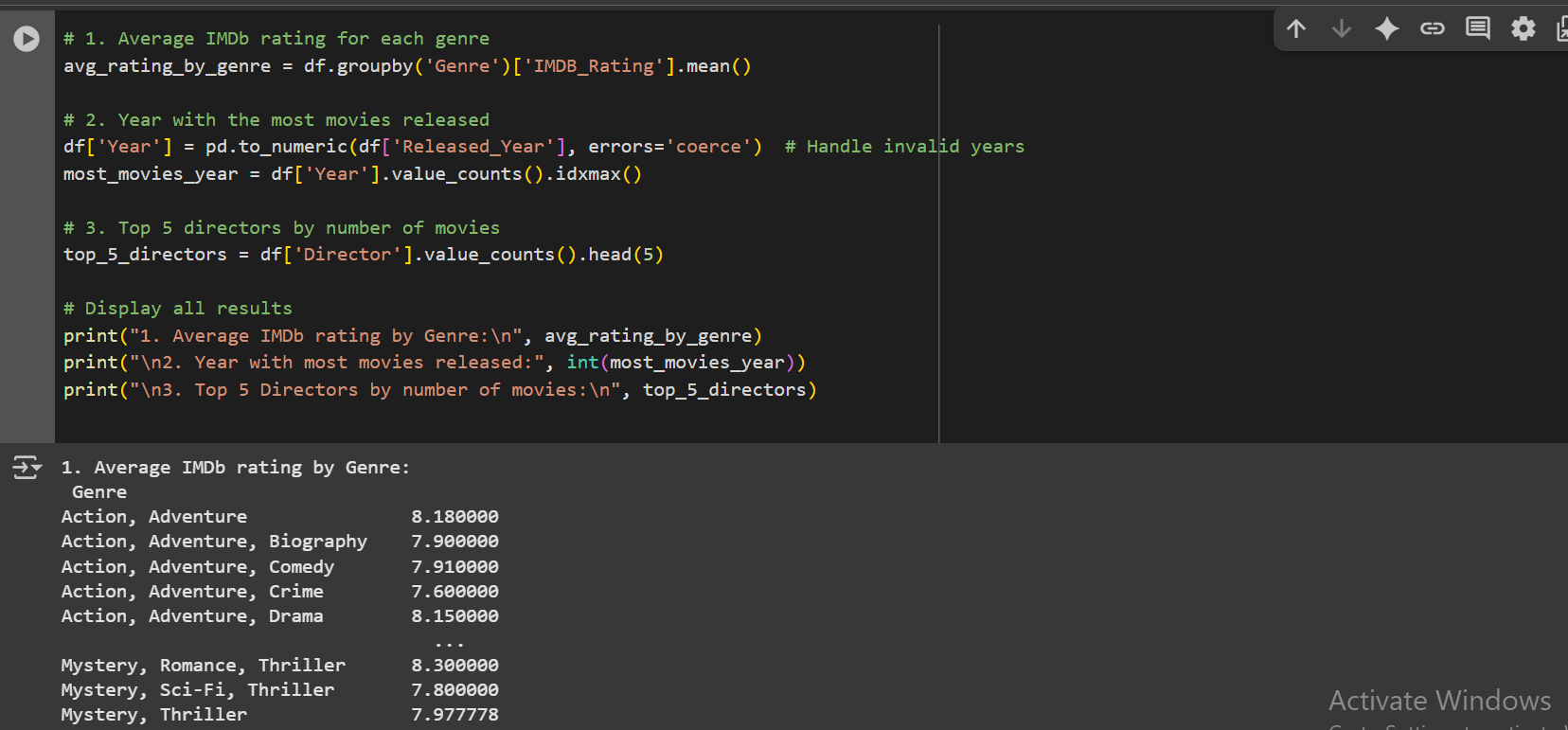


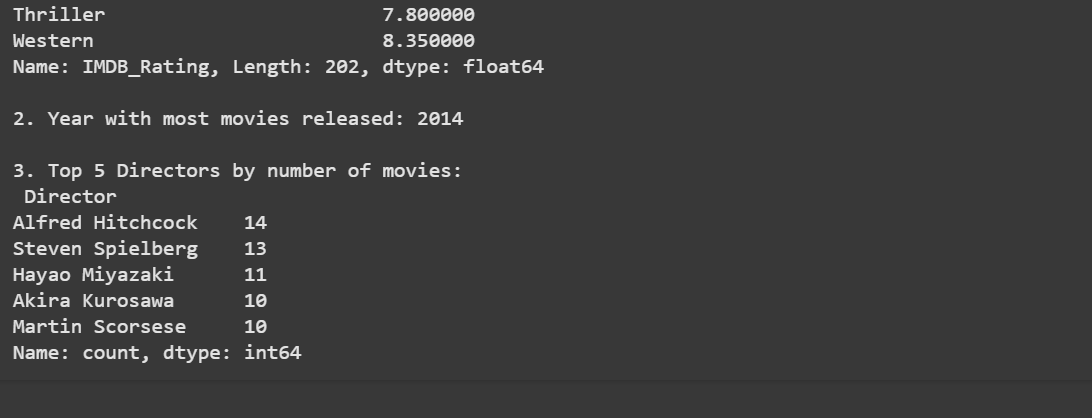
1. Sort the dataset based on **IMDb rating in descending order**.



**Data Aggregation & Grouping**

1. Find the **average IMDb rating** for each genre.
2. Determine which **year had the most movies released**.
3. Find the **top 5 directors** who have directed the most movies in the dataset.



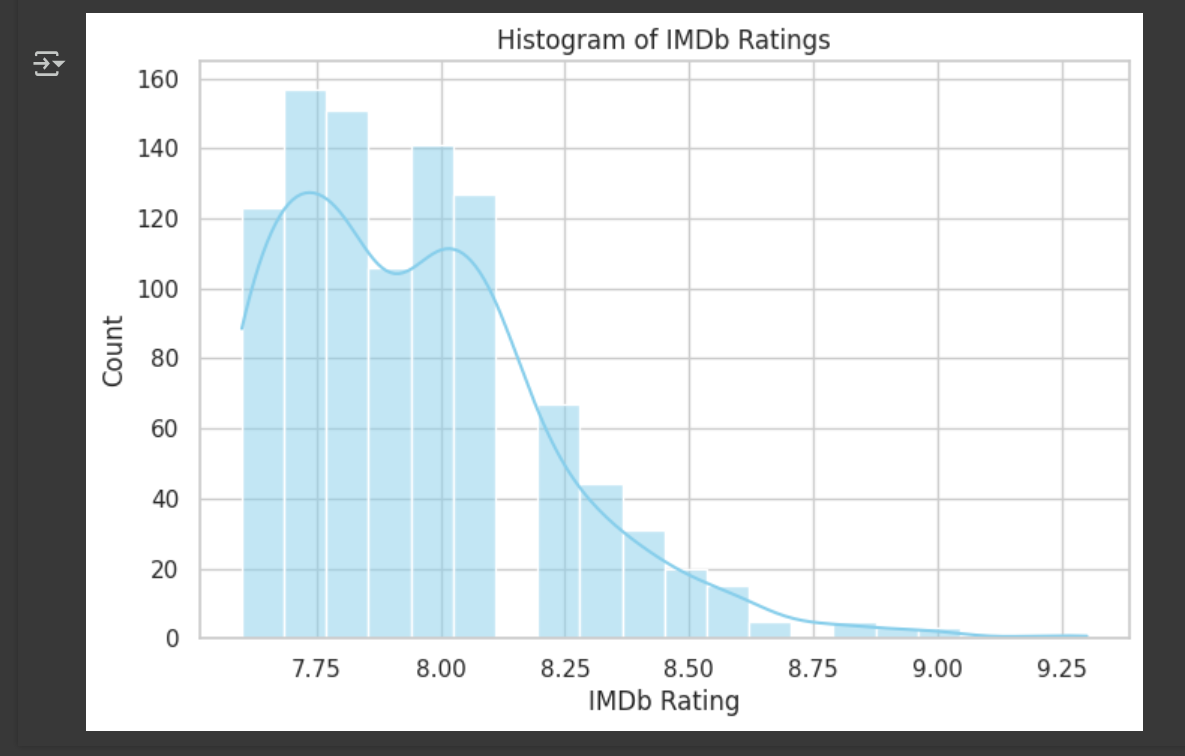


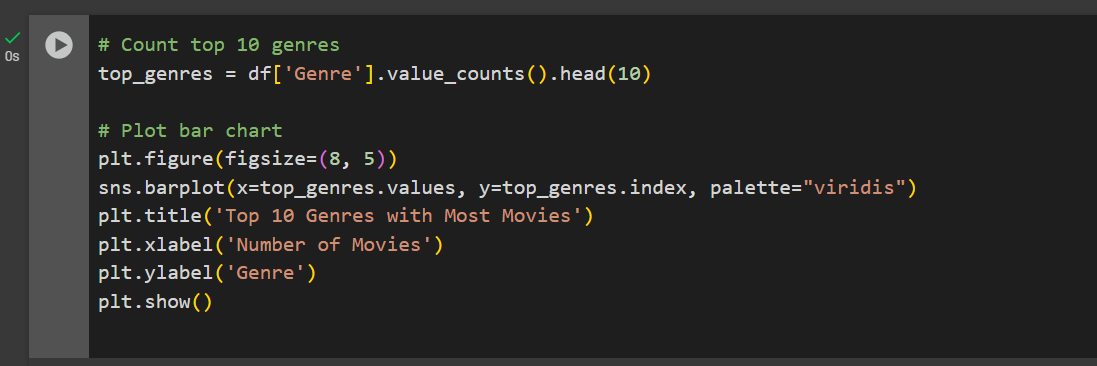
**Visualization (Optional)**

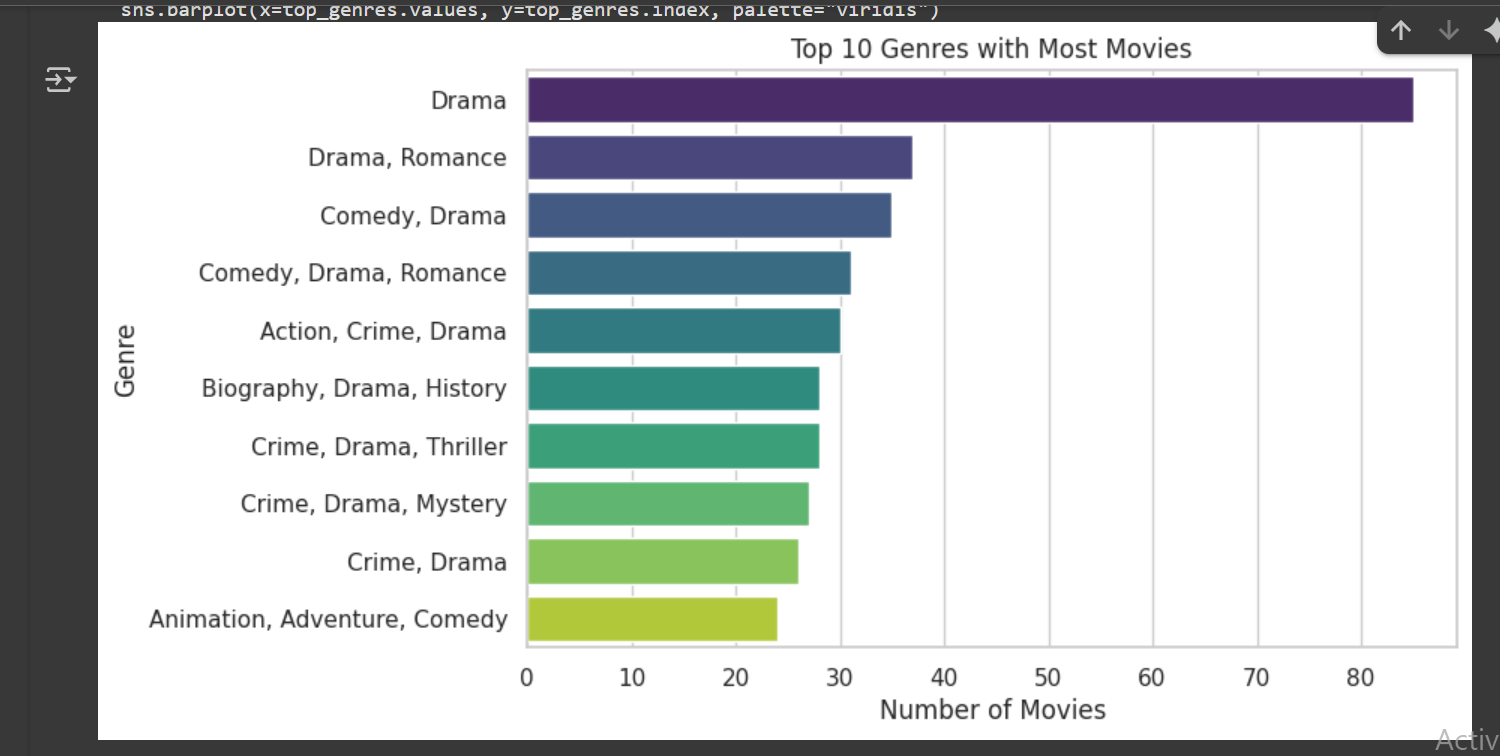
**Matplotlib** or **Seaborn**

1. Plot a histogram of IMDb ratings.

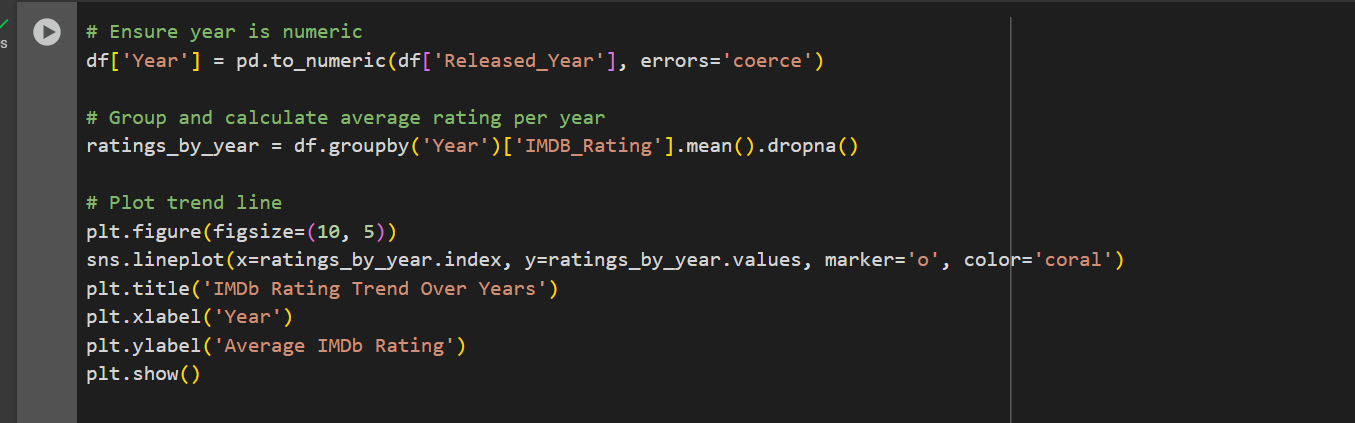


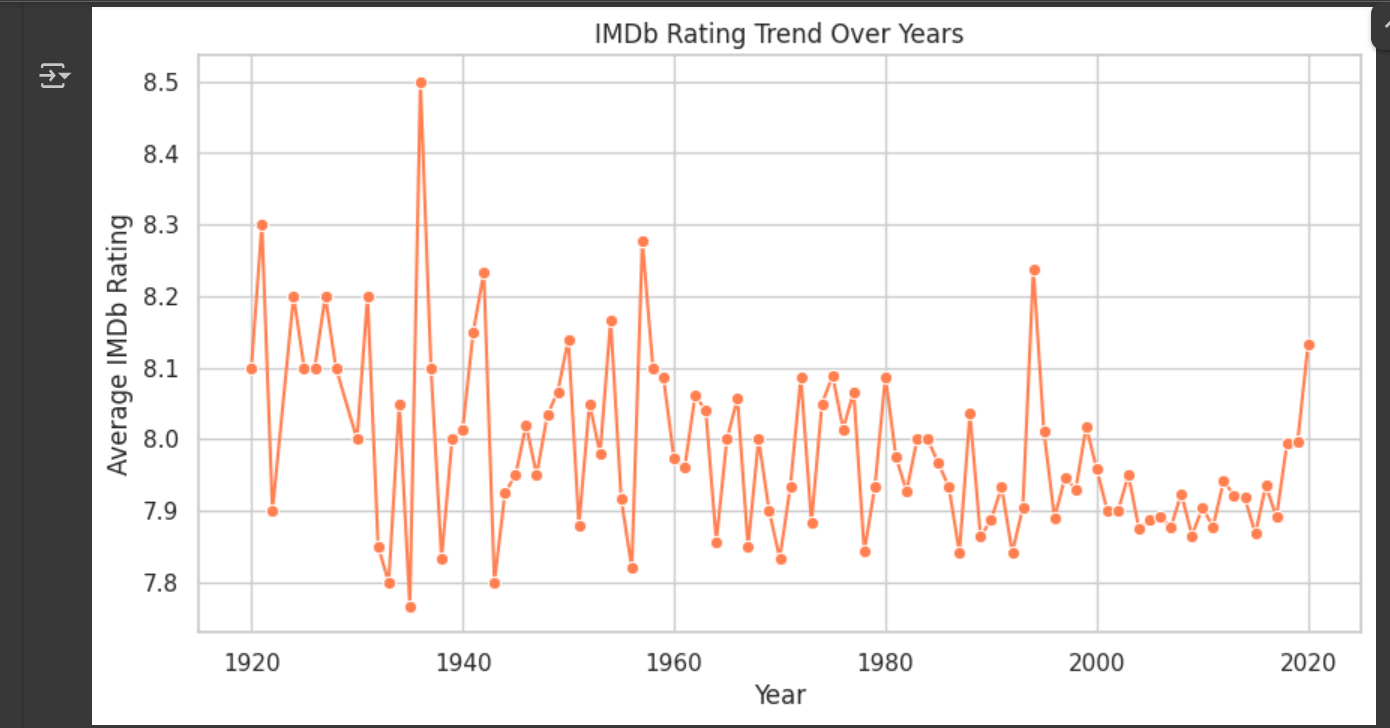


1. Create a bar chart showing the **top 10 genres** with the most movies.



1. Visualize the **trend of IMDb ratings over the years**.

****

****

## Lab No: 4

**Objective:** To enable students to understand and implement the **Breadth-First Search algorithm** for solving graph traversal and pathfinding problems in artificial intelligence applications.

Breadth-First Search (BFS) is an **uninformed search algorithm** that explores a graph level by level. It begins at a selected node (called the root or source) and explores all neighbouring nodes at the current depth before moving on to nodes at the next depth level.

It uses a **queue** data structure (FIFO) to keep track of the nodes to be visited.

**Practical Significance: -**

Breadth-First Search (BFS) has practical significance in various fields and applications due to its unique characteristics. Here are some practical applications:

Network Routing and Broadcasting:

In computer networks, BFS is often used to discover neighboring nodes and determine the shortest path for routing.

It is also employed in broadcasting information across a network efficiently.

Web Crawling:

Search engines use BFS to crawl the web and index pages. Starting from a seed page, BFS explores links level by level, ensuring a systematic and comprehensive traversal.

Puzzle Solving:

BFS is used in puzzle-solving scenarios, such as the famous "Eight Puzzle" or "Fifteen Puzzle," to find the shortest sequence of moves to reach the goal state.

Maze Solving:

BFS can be applied to solve mazes by finding the shortest path from the start to the exit. It guarantees the discovery of the shortest path when the maze has uniform edge weights.

Robotics and Autonomous Vehicles:

BFS is employed in robotics and autonomous vehicle navigation to explore and map unknown environments systematically.

Optimizing Data Structures:

BFS is often used in optimizing data structures like trees and graphs, ensuring efficient access and retrieval of information.

Game Development:

BFS can be applied in game development for tasks such as pathfinding, where it helps in finding the shortest path for characters or objects.

Database Querying:

BFS is used in certain database querying scenarios to explore relationships and dependencies between different entities.

In summary, BFS is a versatile algorithm with practical applications across various domains, providing an efficient way to explore and analyze relationships in interconnected systems.

**BFS Algorithm: -**

Input:

Graph G represented as an adjacency list, starting vertex start, and goal vertex goal.

Initialization:

Create an empty set visited to keep track of visited vertices. Create a deque queue and enqueue the start vertex.

Add the start vertex to the visited set.

BFS Loop:

While the queue is not empty:

Dequeue a vertex current\_vertex from the front of the queue. Print or process current\_vertex.

If current\_vertex is equal to the goal vertex:

Print a message indicating that the goal state is reached. Return, indicating that the goal state is reached.

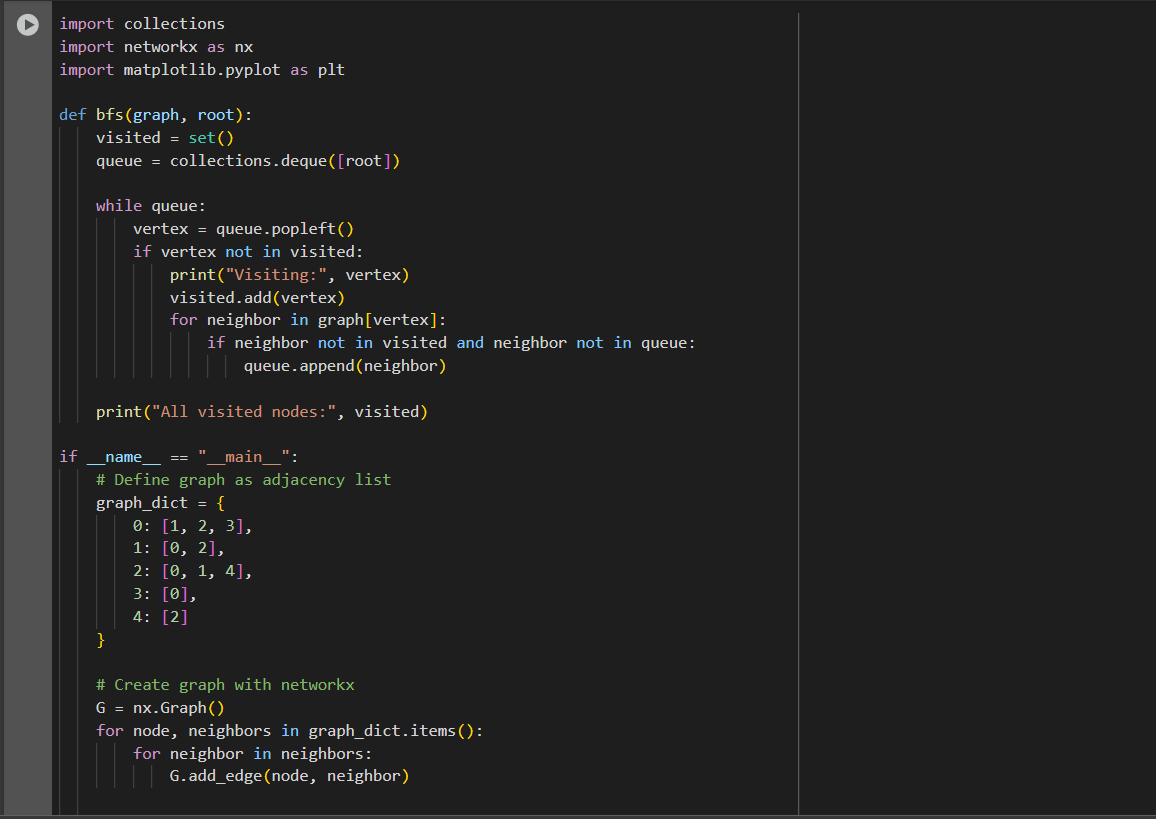
For each neighbor neighbor of current\_vertex in the graph: If neighbor is not in the visited set:

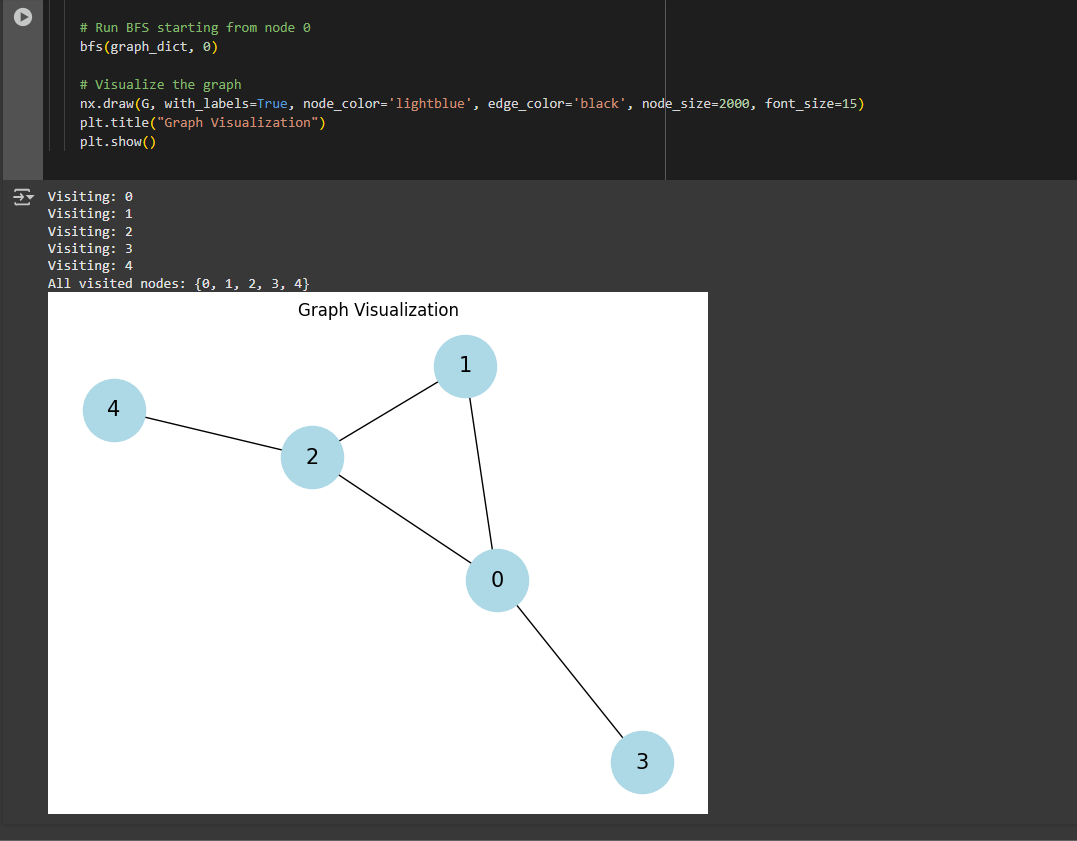
Enqueue neighbor to the back of the queue. Add neighbor to the visited set.

Output: Print a message indicating that the goal state is not reached if the loop completes without returning.

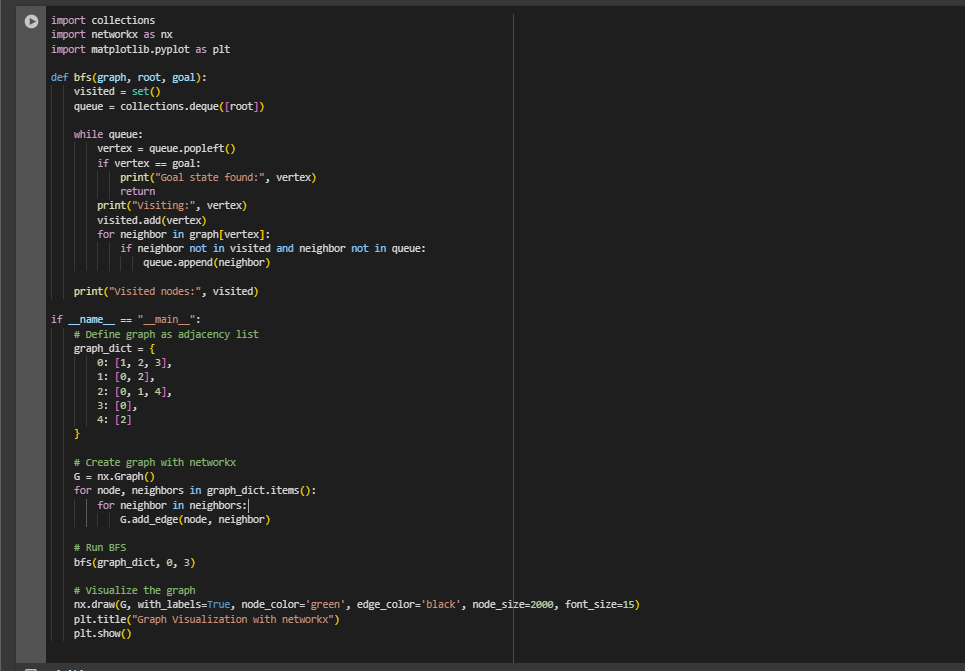
**Tasks: -**

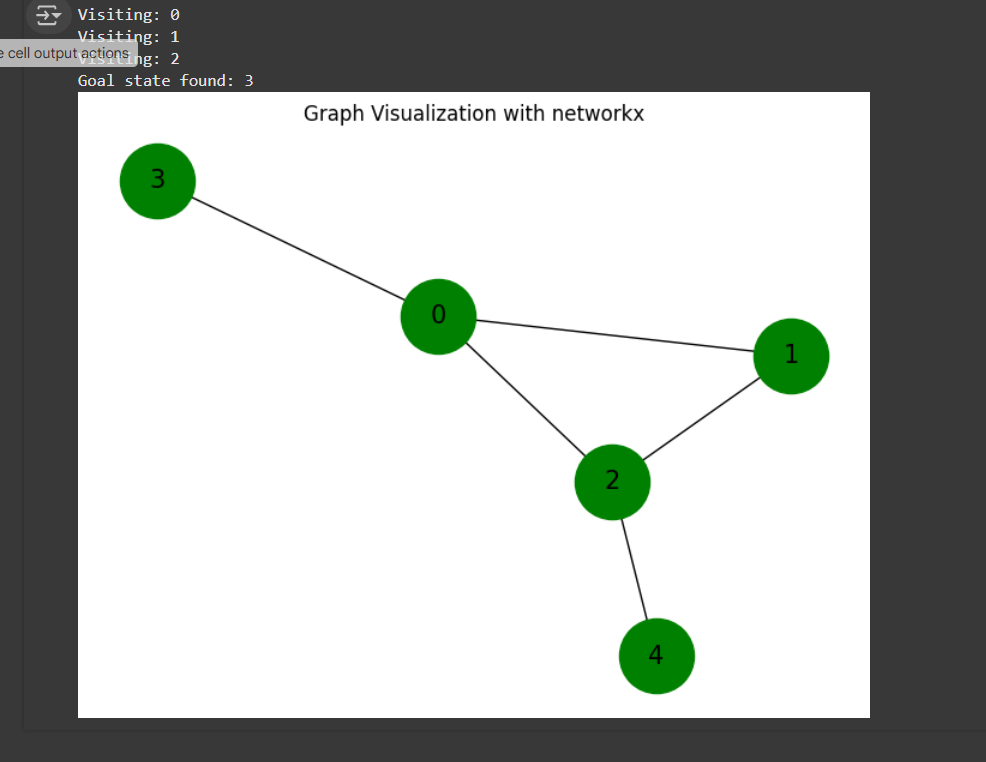
* 1. Write a Program to Implement Breadth First Search without goal state using Python.





* 1. Write a Program to Implement Breadth First Search with goal state using Python.

****

****

## Lab No: 5

**Objective:** To enable students to understand and implement the **Depth-First Search algorithm** for exploring graphs or state spaces

### **Practical Significance: -**

Depth-First Search (DFS) is a versatile algorithm with practical significance in various domains. Here are some practical applications and use cases of DFS:

Pathfinding and Maze Solving:

DFS is commonly used to find paths and solve mazes. Its recursive nature makes it efficient in exploring paths until a solution is found.

Cycle Detection:

DFS can be applied to detect cycles in a graph. This is useful in dependency analysis, resource allocation, and preventing deadlocks in concurrent systems.

Graph Traversal:

DFS is fundamental for graph traversal and exploration. It is used in applications such as network analysis, social network mapping, and web crawling.

Puzzle Solving:

DFS is employed in solving puzzles, such as the N-Queens problem and the Tower of Hanoi. It systematically explores possible states until a solution is found.

Artificial Intelligence:

DFS is applied in AI algorithms, particularly in decision tree traversal, game playing (e.g., chess, tic-tac-toe), and state space exploration.

Anomaly Detection:

DFS can be employed in anomaly detection systems to identify unusual patterns or behaviors in data.

The practical significance of DFS lies in its ability to systematically explore and analyze complex structures, making it a valuable tool in a wide range of applications across computer science, mathematics, engineering, and artificial intelligence.

**DFS Algorithm: -**

Input:

Graph G represented as an adjacency list, starting vertex start, and goal vertex goal.

Initialization:

Create an empty set visited to keep track of visited vertices. Create a deque stack and push the start vertex onto it.

Add the start vertex to the visited set.

DFS Loop:

While the stack is not empty:

Pop a vertex current\_vertex from the front of the stack. Print or process current\_vertex.

If current\_vertex is equal to the goal vertex:

Print a message indicating that the goal state is reached. Return, indicating that the goal state is reached.

For each neighbor neighbor of current\_vertex in the graph: If neighbor is not in the visited set:

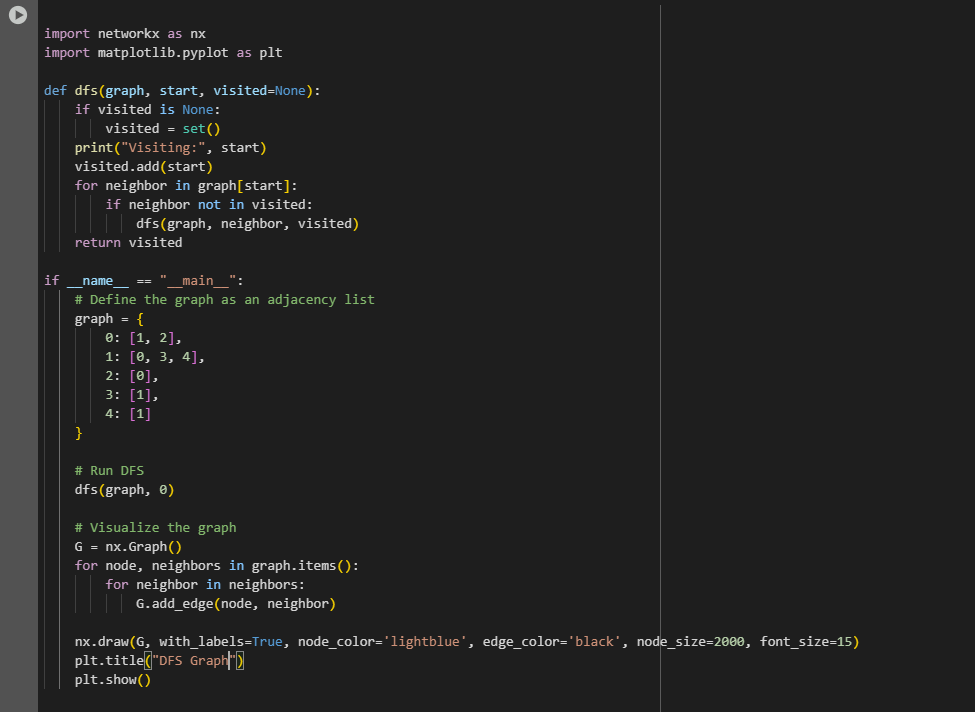
Push neighbor onto the front of the stack. Add neighbor to the visited set.

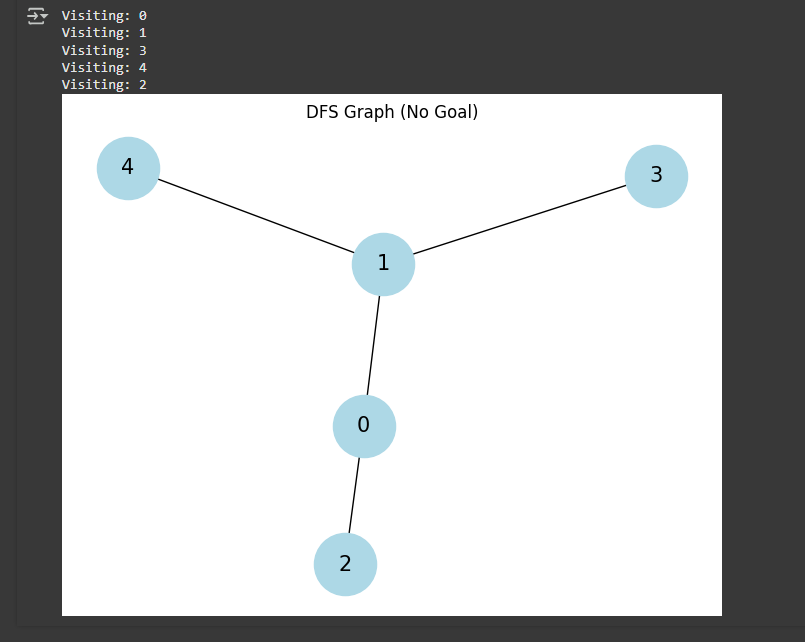
Output:

Print a message indicating that the goal state is not reached if the loop completes without returning.

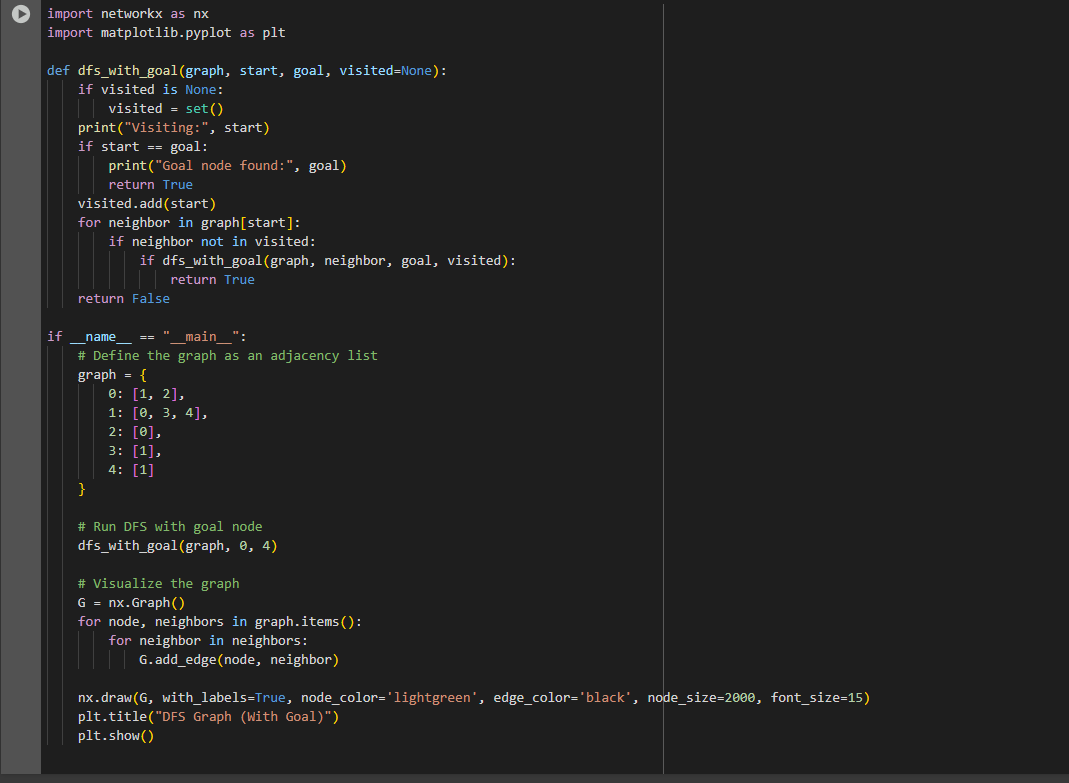
**Tasks: -**

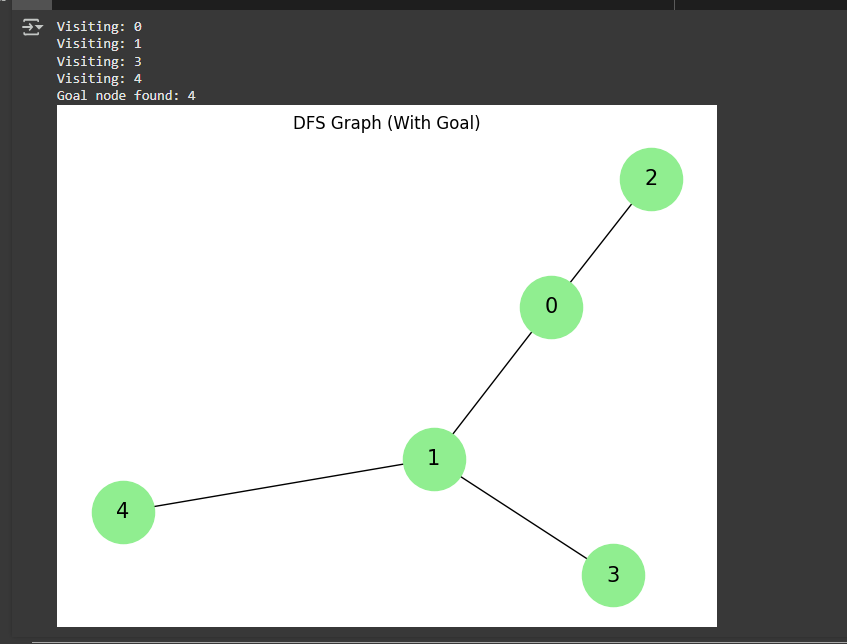
1. Write a Program to Implement Depth First Search without goal state using Python.





1. Write a Program to Implement Depth First Search with goal state using Python.





**Lab No : 06**

**Objective:** To introduce students to the concept of **Best-First Search** and enable them to implement it using basic priority-based exploration

**What is Best-First Search?**

Best-First Search (BFS) is a search algorithm that explores a graph by selecting the most promising node based on a specific criterion. It uses a priority queue to decide the order in which nodes are explored.

When implemented without heuristics, Best-First Search can behave similarly to other uninformed search algorithms—like Breadth-First Search or Uniform Cost Search—depending on how the priority is defined.

| **Feature** | **Description** |
| --- | --- |
| **Search Type** | Informed |
| **Data Structure** | Priority Queue |
| **Goal** | To reach the goal node by expanding the least costly or earliest node |
| **Priority Basis** | May use path cost (g(n)) or simple order of discovery |

**Example Use Case: -**

A basic priority-based search where the algorithm always chooses the next node alphabetically or based on node depth (depending on the implementation) is an example of Best-First Search.

**BFS Algorithm:**

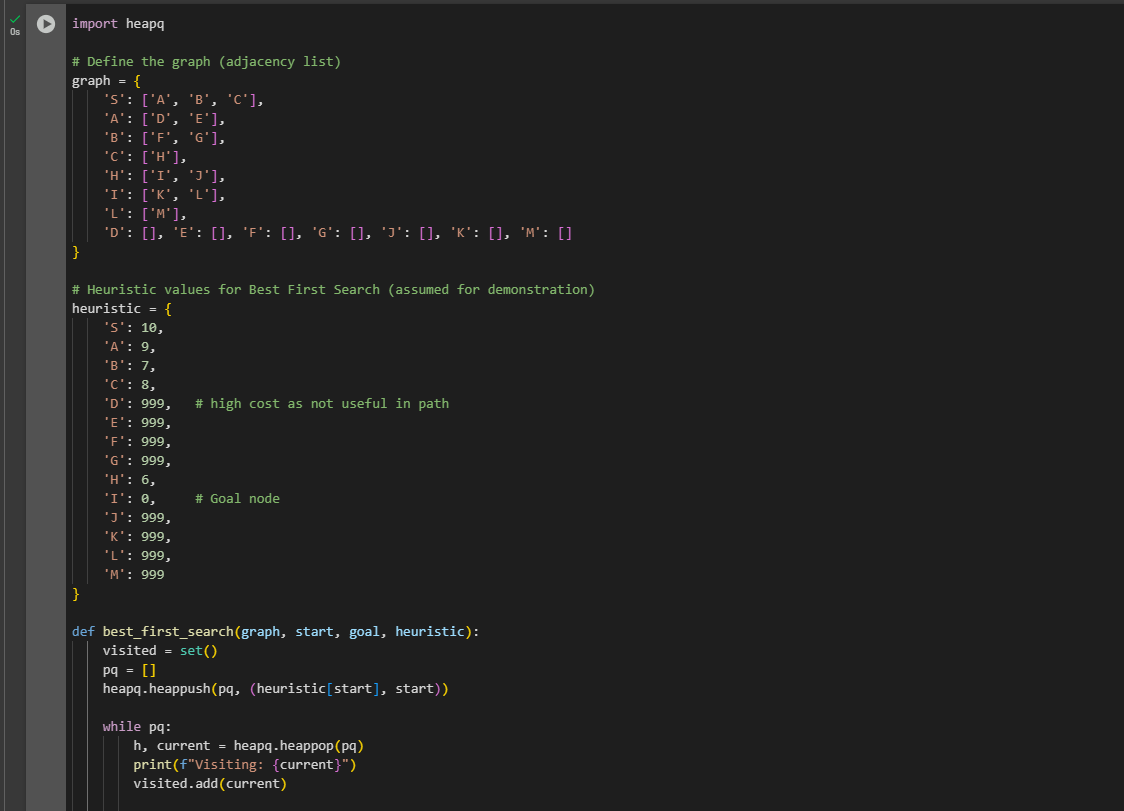
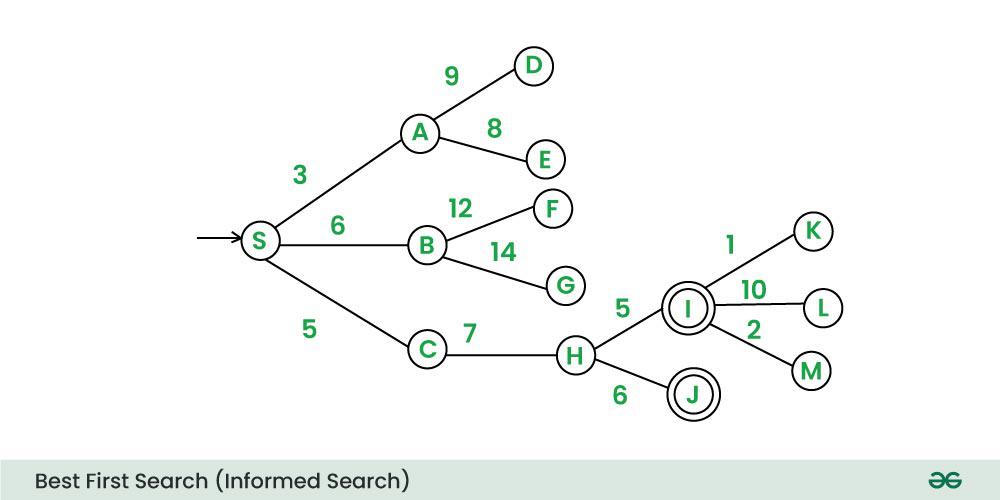
If we are given an edge list of a graph where every edge is represented as (u, v, w). Here u, v and w represent source, destination and weight of the edges respectively. We need to do Best First Search of the graph (Pick the minimum cost edge next).

* Initialize an empty Priority Queue named **pq**.
* Insert the starting node into **pq**.
* While **pq** is not empty:
  + Remove the node **u** with the lowest evaluation value from **pq**.
  + If **u** is the goal node, terminate the search.
  + Otherwise, for each neighbor **v** of **u**: If **v** has not been visited, Mark **v** as visited and Insert **v** into **pq**.
  + Mark **u** as examined.
* End the procedure when the goal is reached or **pq** becomes empty.

**Task:**

Write a Program to Implement Best First Search of the following graph from starting node “S” to goal node “I” using Python. To help with writing the program following steps are provided for guidance:

* We start from source "S" and search for goal "I" using given costs and Best First search.
* pq initially contains S
  + We remove S from pq and process unvisited neighbors of S to pq.
  + pq now contains {A, C, B} (C is put before B because C has lesser cost)
* We remove A from pq and process unvisited neighbors of A to pq.
  + pq now contains {C, B, E, D}
* We remove C from pq and process unvisited neighbors of C to pq.
* pq now contains {B, H, E, D}
* We remove B from pq and process unvisited neighbors of B to pq.
  + pq now contains {H, E, D, F, G}
* We remove H from pq.
* Since our goal "I" is a neighbor of H, we return*.*

****

****

## Lab No: 7

**Objective:** To implement the **A\* algorithm** for finding the shortest path using both actual and heuristic costs in intelligent search problems.

**What is A\* Search?**

A\* is an informed search algorithm that finds the shortest path from a start node to a goal node by combining:

* g(n): Actual cost from the start node to the current node.
* h(n): Heuristic estimate of the cost from the current node to the goal.
* f(n) = g(n) + h(n): Total estimated cost of the cheapest solution through node n.

**Key Properties of A\* Search: -**

| **Property** | **Description** |  |
| --- | --- | --- |
| Informed? | Yes – uses heuristics |  |
| Optimal? | Yes – if the heuristic is admissible (never overestimates) |  |
| Complete? | Yes |  |
| Time Complexity | Can be high depending on heuristic accuracy |  |
| Data Structure | Priority Queue based on f(n) |  |

**Use Cases in AI: -**

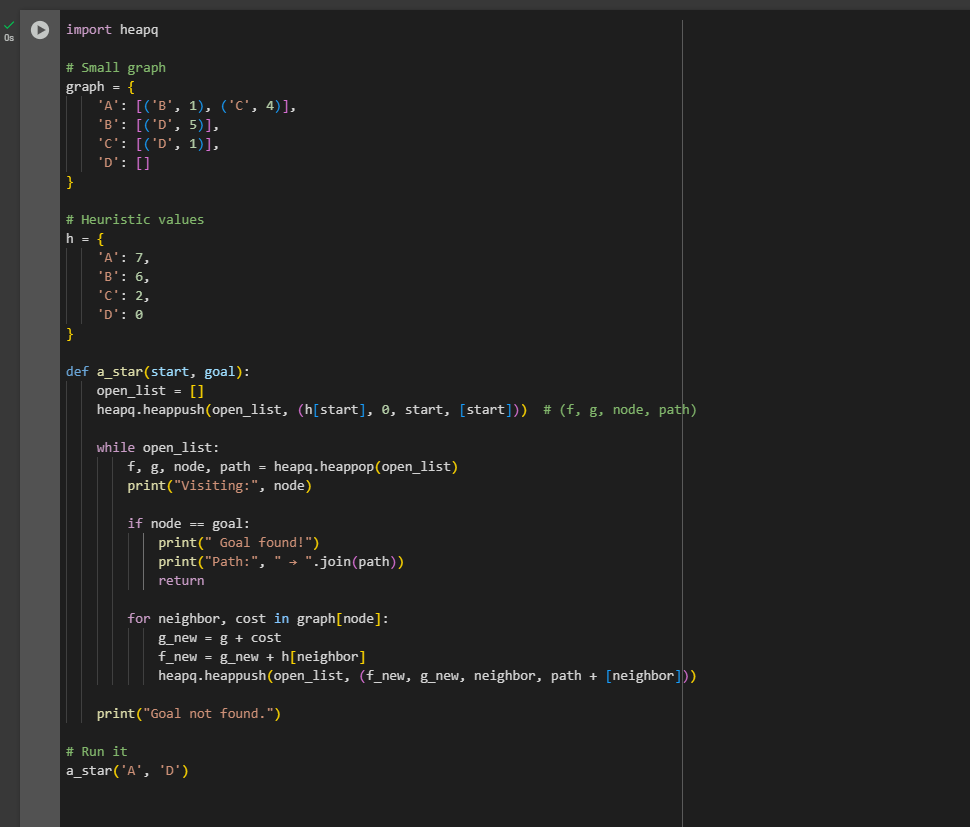
* Pathfinding in maps or games.
* Puzzle solvers (e.g., 8-puzzle, sliding tiles).
* Planning and robotics.

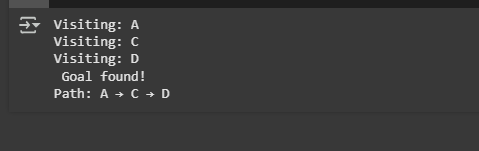
**A\* Algorithm Steps: -**

1. Initialize the open list (priority queue) with the start node.
2. Loop until the open list is empty:
   * Remove the node with the lowest f(n) from the open list.
   * If it is the goal, return the path.
   * Else, generate its neighbors.
   * For each neighbor:
     + Calculate g(n), h(n), and f(n).
     + Add to open list if not visited or if a better f(n) is found.

**Task:**

Write a Program to Implement A\* Algorithm with goal state using Python.



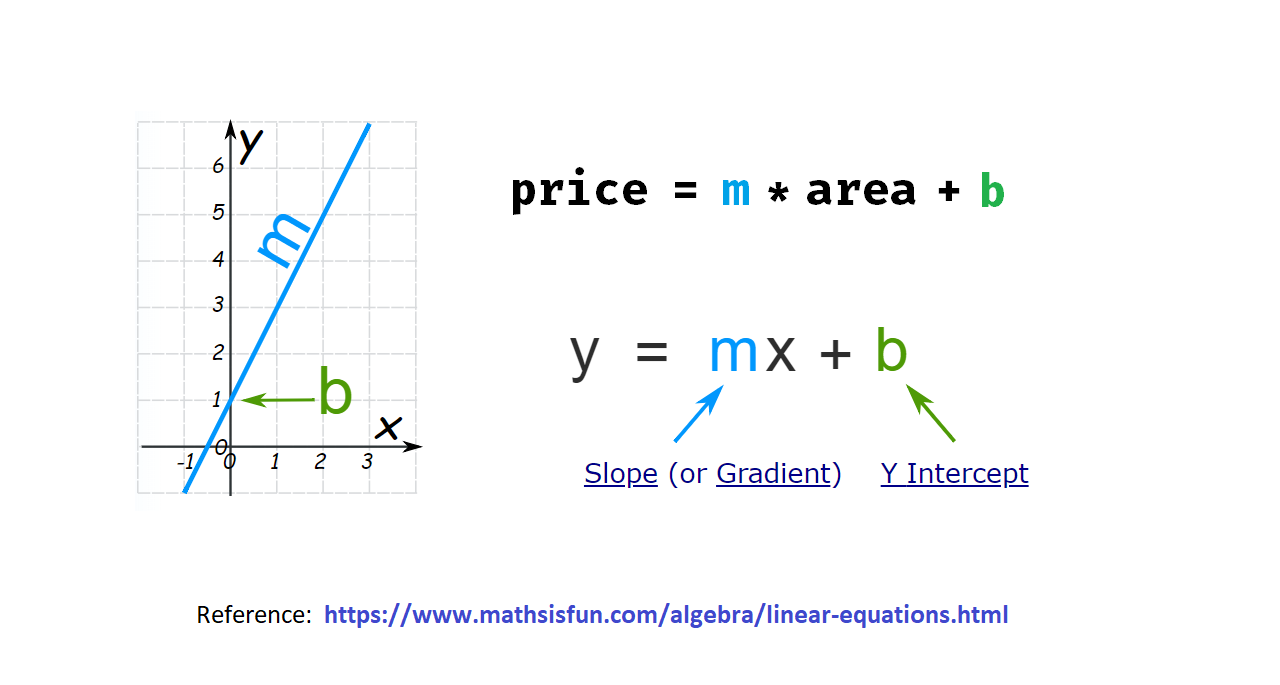
**Output:**

## Lab No: 8

**Objective:** To implement **simple linear regression** and understand how it models the relationship between two variables for predictive analysis.

**What is Simple Linear Regression?**

Simple Linear Regression is a supervised learning algorithm that models the relationship between a dependent variable (Y) and a single independent variable (X) using a straight line.



**The model has the form:**

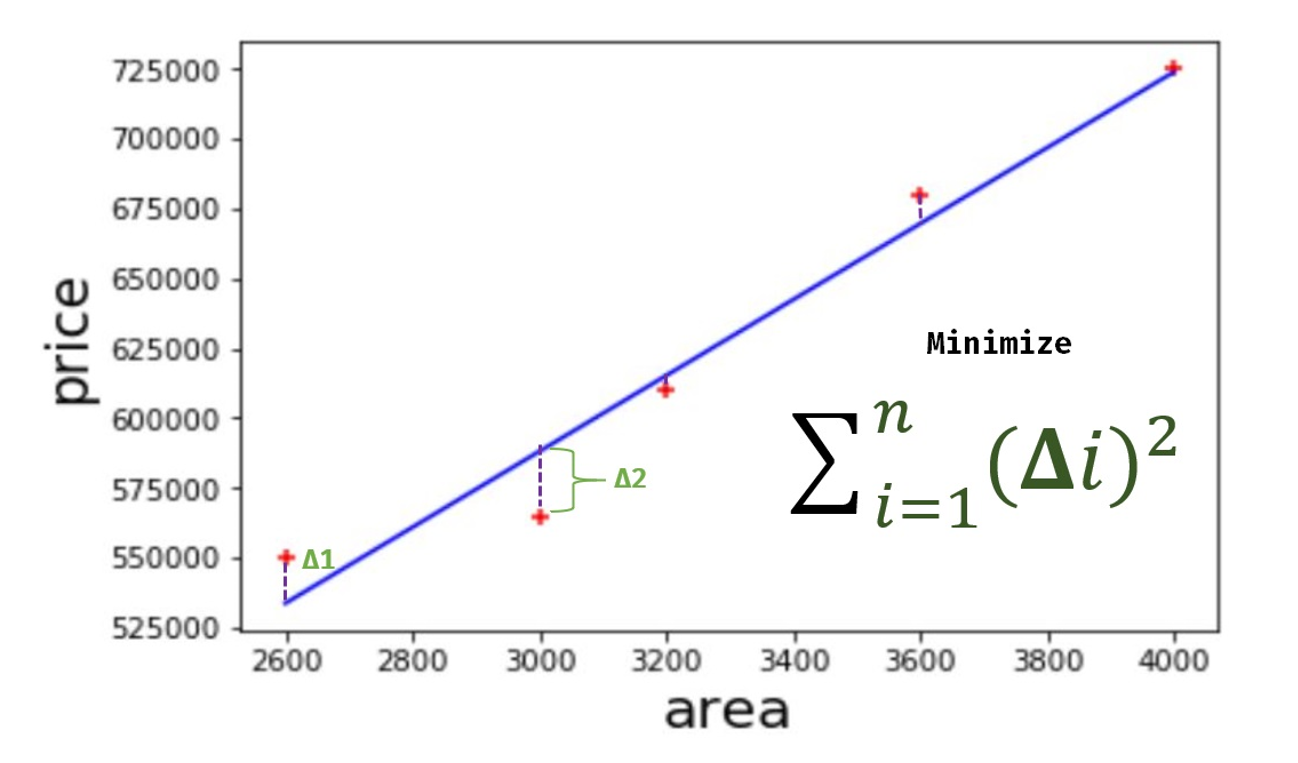
Y=mX+b

Where:

* Y = Predicted value
* X = Input feature
* m = Slope (coefficient)
* b = Intercept (bias)

**Goal of the Algorithm: -**

To find the best-fitting line (regression line) that minimizes the error between actual and predicted values (usually using Mean Squared Error).

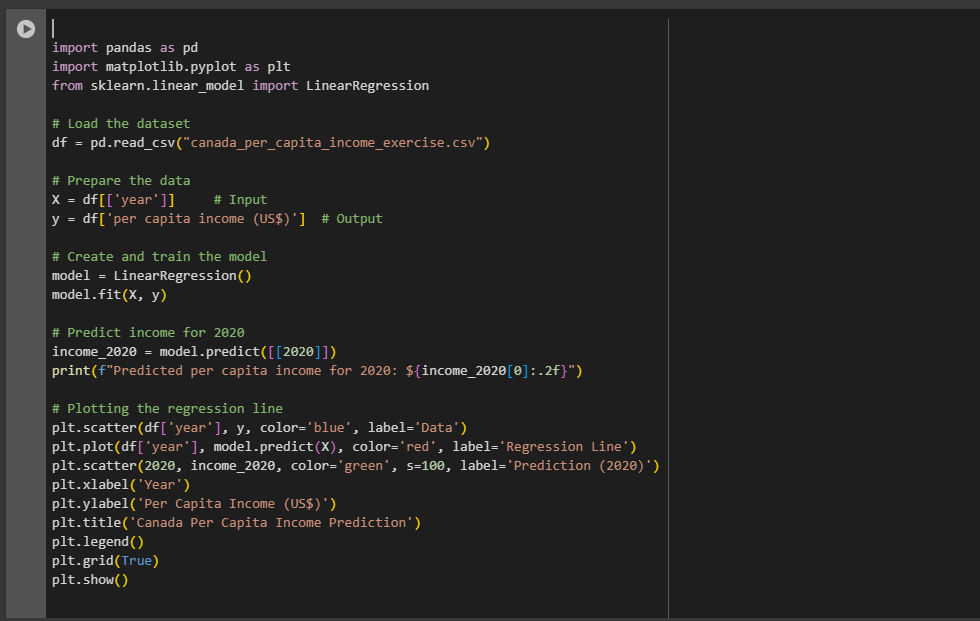


**Key Terms: -**

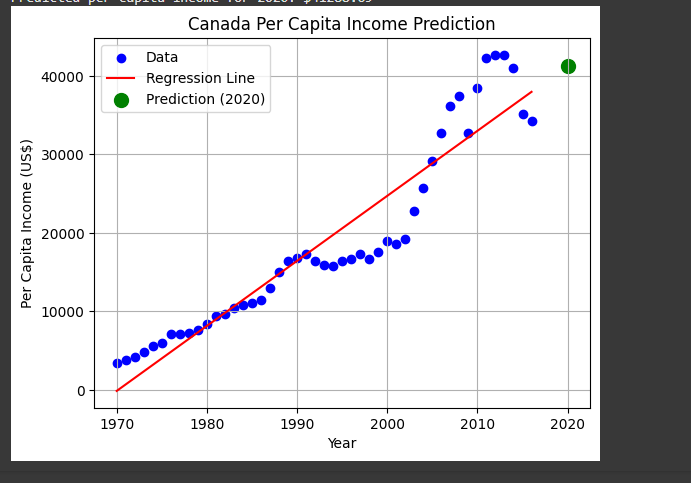
* Independent variable (X) – The input or feature.
* Dependent variable (Y) – The output or label.
* Loss Function – Measures prediction error (commonly MSE).

**Tasks:**

Predict Canada’s per capita income in year 2020. Using this build a regression model and predict the per capita income for Canadian citizens in year 2020. canada\_per\_capita\_income\_exercise.csv file has been provided for dataset.

****

**Output:**

****

## Lab No: 09

**Objective:** To implement **multivariate linear regression** and understand how multiple features can be used to predict a continuous output variable.

**What is Multivariate Linear Regression?**

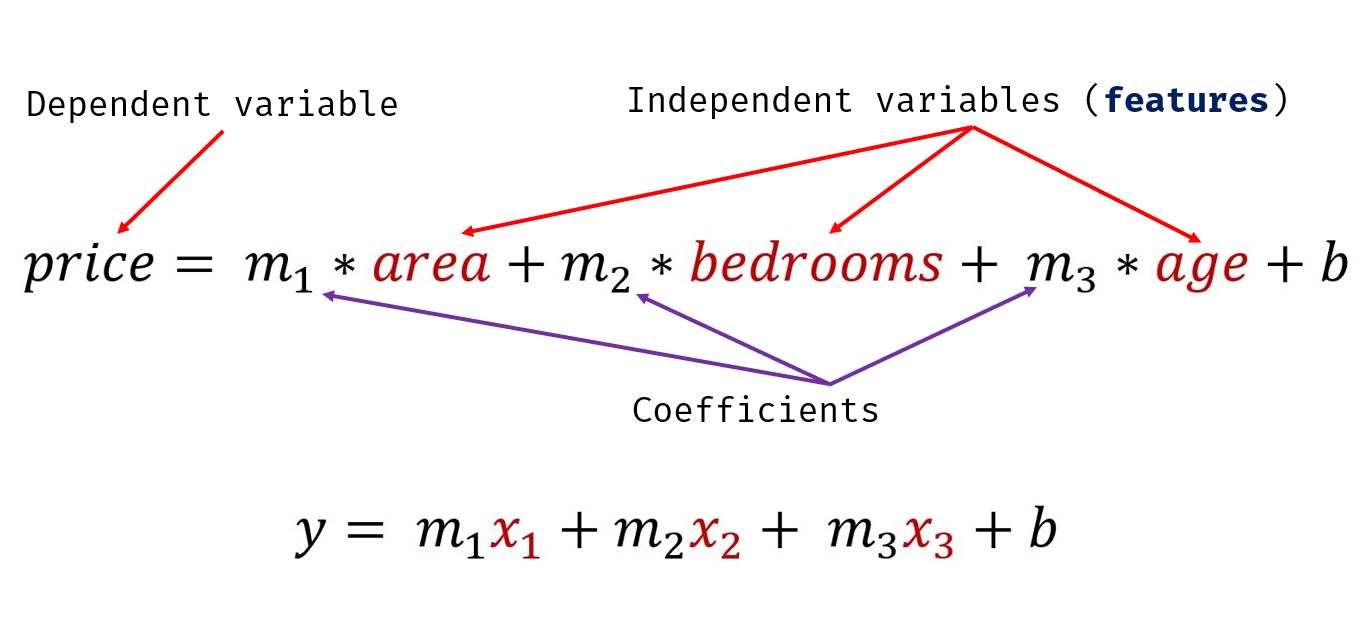
Multivariate Linear Regression extends simple linear regression by modeling the relationship between a dependent variable (Y) and multiple independent variables (X₁, X₂, ..., Xₙ).

**The model takes the form:**

Y=b0+b1X1+b2X2+⋯+bnXn ​

Where:

* Y = Output (dependent variable)
* X₁ to Xₙ = Input features (independent variables)
* b₀ = Intercept
* b₁ to bₙ = Coefficients (slopes)



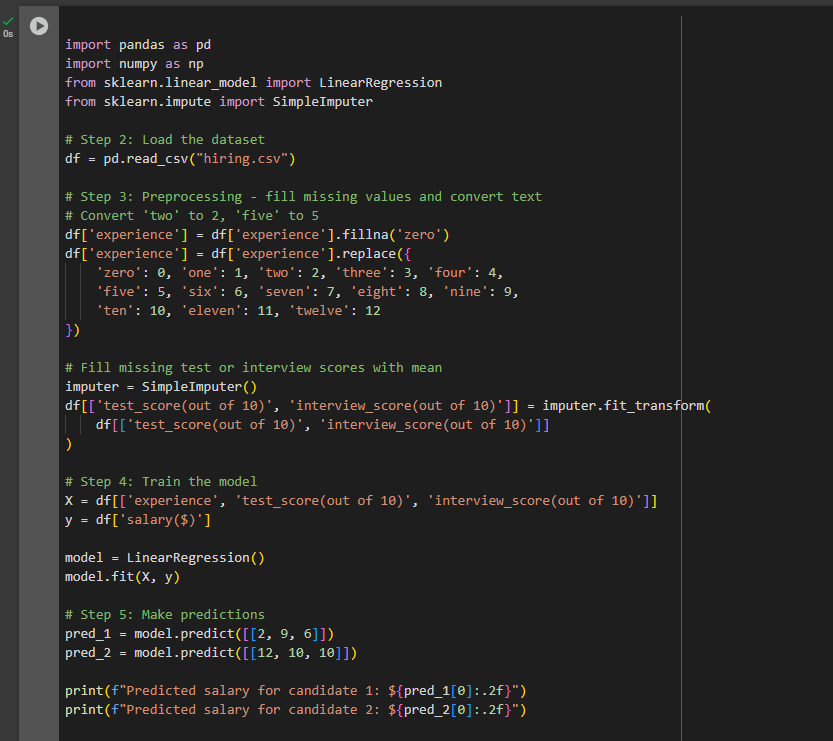
**Key Concepts: -**

* Multiple features are used to improve prediction accuracy.
* The model learns coefficients that best fit the training data.
* Error minimization is usually done using Mean Squared Error (MSE).

**Task:**

There is **hiring.csv**. This file contains hiring statics for a firm such as experience of candidate, his written test score and personal interview score. Based on these 3 factors, HR will decide the salary. Given this data, you need to build a machine learning model for HR department that can help them decide salaries for future candidates. Using this predict salaries for following candidates:

* 2 yr experience, 9 test score, 6 interview score
* 12 yr experience, 10 test score, 10 interview score



**Output:**

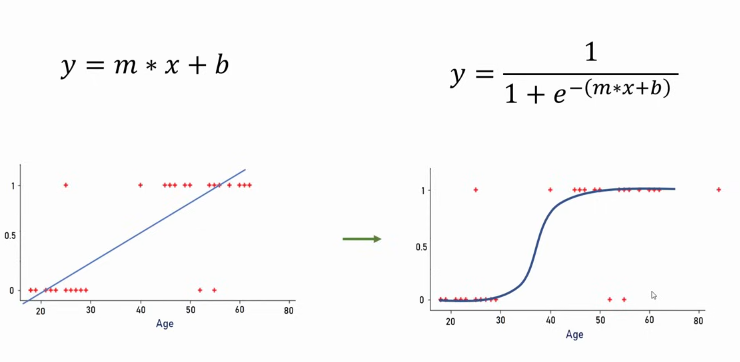
## Lab No: 10

**Objective:** To implement **logistic regression** for binary classification tasks and understand how it models the probability of class membership.

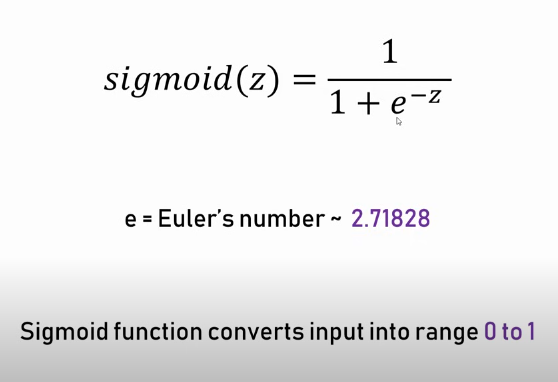
**What is Logistic Regression?**

Logistic Regression is a supervised learning algorithm used for binary classification. It predicts the probability that a given input belongs to a particular class (typically 0 or 1).

Unlike linear regression, it uses the sigmoid (logistic) function to map predicted values to a probability between 0 and 1.

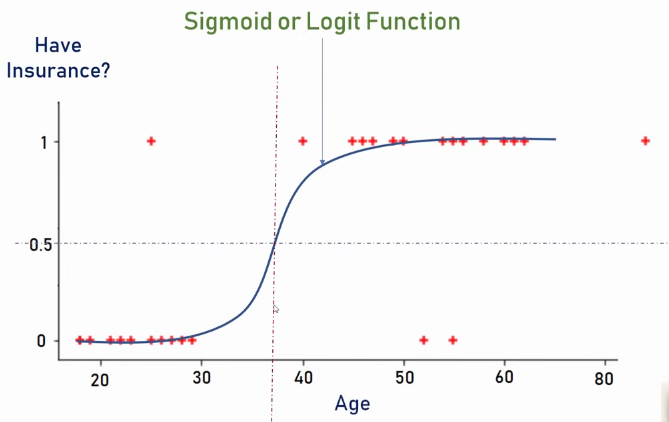


**Sigmoid Function: -**

****

Where:

* z = w₀ + w₁x₁ + w₂x₂ + ... + wₙxₙ (linear combination of features)
* Output: probability (e.g., if > 0.5 → class 1, else class 0)

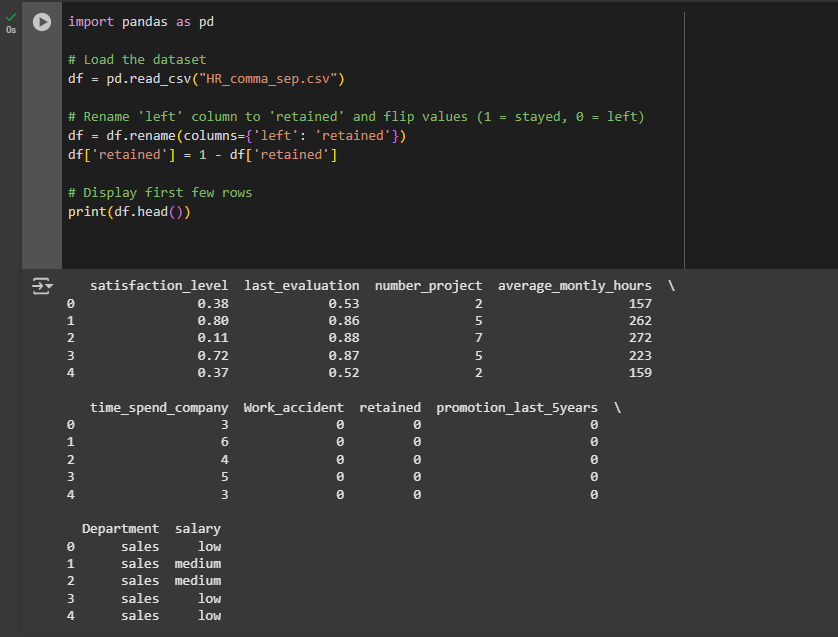


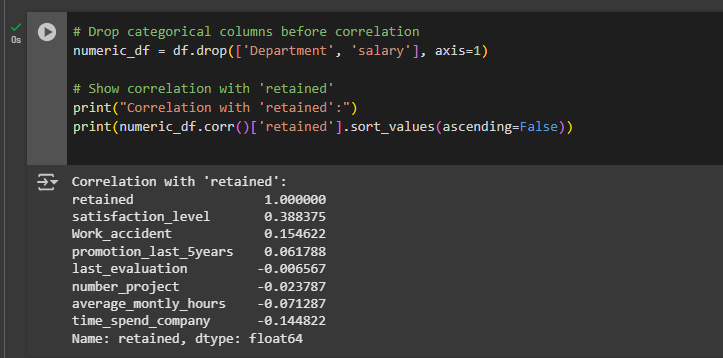
**Key Concepts**

* Output is a probability score.
* Decision boundary separates the two classes (e.g., at 0.5).
* Loss function used is log loss or binary cross-entropy.

**Tasks:**

Download employee retention dataset from here: https://www.kaggle.com/giripujar/hr-analytics.

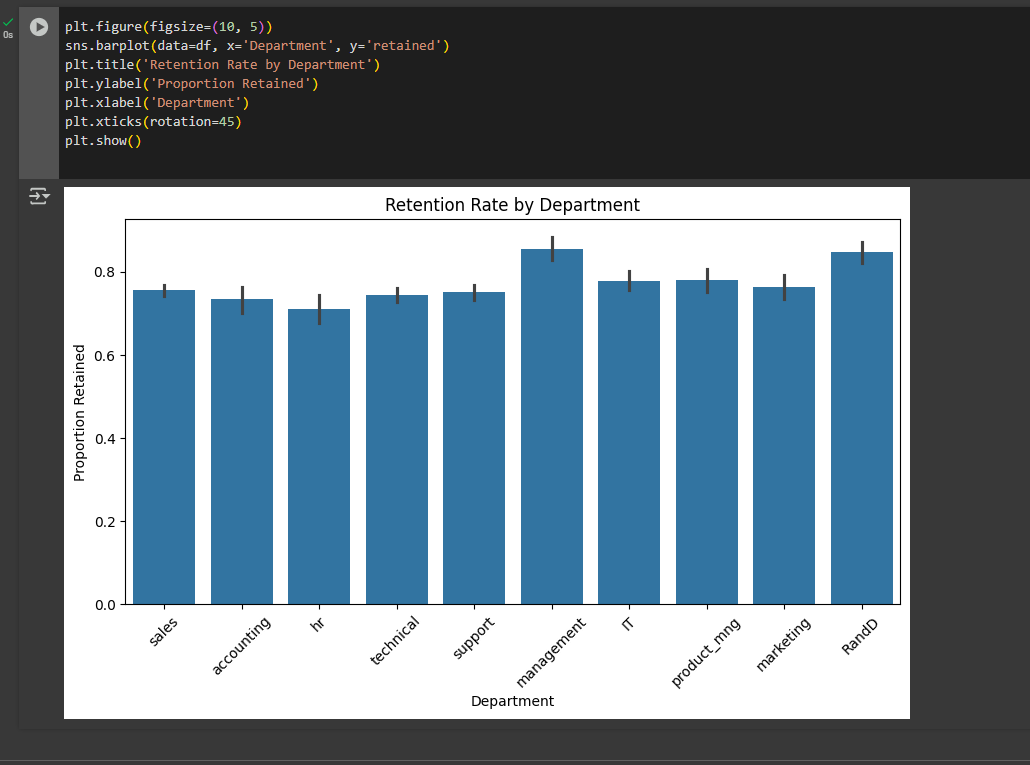
1. Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)

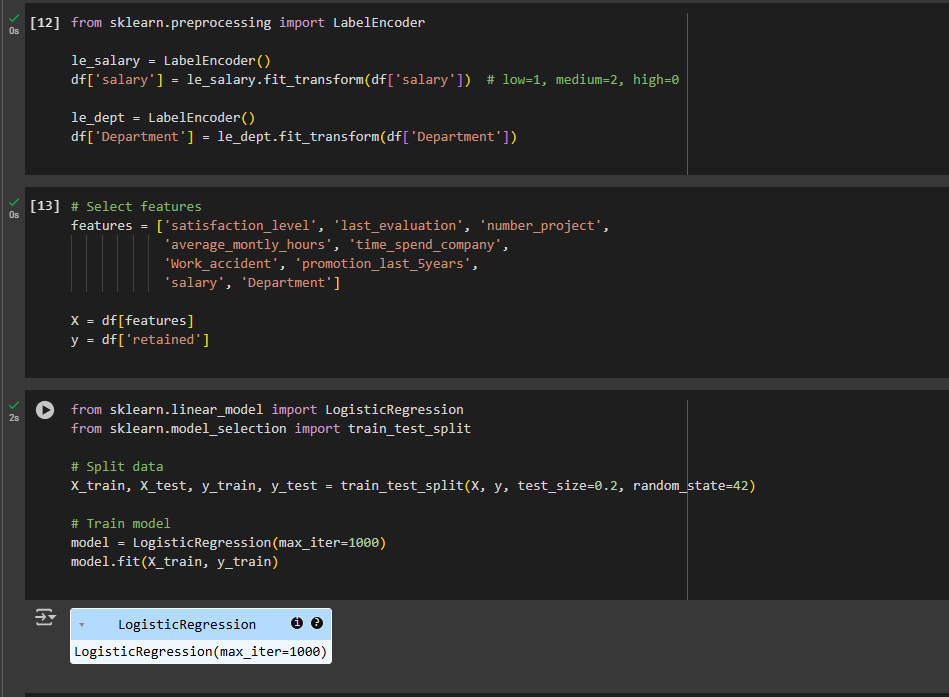


1. Plot bar charts showing impact of employee salaries on retention



3.Plot bar charts showing correlation between department and employee retention



4.Now build logistic regression model using variables that were narrowed down in step 1

5.Measure the accuracy of the model