**Name: Soniya Gupta**

**QUESTION-1**

Student Marks Analysis in Excel

**Aim:**

To design an Excel workbook that calculates students' final marks, percentages, and grades, enabling quick lookup of individual performance.

**Requirement:**

* Microsoft Excel (any version supporting formulas and multiple worksheets)
* Basic familiarity with functions such as AVERAGE(), IF(), and VLOOKUP()
* Dataset of at least 10 students with two semester scores

**Procedure:**

* On Sheet1, create columns: Student Name, Roll Number, Semester 1 Marks, Semester 2 Marks.
* Use =AVERAGE(C2:D2) to calculate Final Marks.
* Display Percentage (out of 100) using the resultant average.
* Apply nested IF() formulas to assign grades (A: 80–100, B: 60–79, C: 40–59, D: below 40).
* In Sheet2, allow the user to input a roll number. Use VLOOKUP() to fetch and display the corresponding student’s name and marks.

**Output:**

* Sheet1: A comprehensive list showing each student’s name, roll number, semester marks, final marks, percentage, and grade.
* Sheet2: A lookup form where entering a roll number instantly retrieves that student's information.

**Conclusion:**

This project demonstrates how Microsoft Excel can be harnessed for academic performance tracking and automated grading using built‑in formulas and lookup functions.

**QUESTION-2**

Fibonacci Series Generator (Python)

**Aim:**

To build a simple Python script that generates the Fibonacci series up to a user‑specified number of terms.

**Requirement:**

* Python installed on system (version 3.x)
* Basic understanding of loops, conditionals, and list operations in Python

**Procedure:**

* Ask the user to input the number of terms they want in the Fibonacci sequence.
* Initialize two starting values of the Fibonacci sequence: 0 and 1.
* Use a for loop to generate the remaining terms by summing the previous two values.
* Store and display the sequence as output.

**Output:**

The full Fibonacci sequence up to the specified number of terms.  
Example: For input 7, output will be: 0, 1, 1, 2, 3, 5, 8

**Conclusion:**

This script helps to understand iterative logic and number patterns. Python makes it easy to implement mathematical series generation through its simple syntax and control structures.

**QUESTION-3**

SQL Category-wise Product Analysis

**Aim:**

To create an SQL table for products and perform analysis to calculate the average price per product category using SQL aggregation.

**Requirement:**

* SQL database (e.g., MySQL, SQLite)
* SQL commands knowledge
* Knowledge of CREATE TABLE, INSERT INTO, and SELECT queries with GROUP BY

**Procedure:**

* Create a product table with columns: product\_id, product\_name, category, and price.
* Insert at least 6 products belonging to 2 or 3 different categories.
* Use the SQL GROUP BY clause along with the AVG() function to calculate the average price of products in each category.
* Run:

SELECT category, AVG(price) AS avg\_price

FROM product

GROUP BY category;

**Output:**

A summarized table showing each category and its average product price.  
Example:

| Category | | Avg Price | |
| --- | --- | --- | --- |
| Electronics | | 25350 | |
| Clothing | 900 | |

**Conclusion:**

The task highlights how SQL can be used for category-wise business insights and price comparisons. Aggregation functions like AVG() combined with GROUP BY are useful for generating such reports efficiently.

**QUESTION-4**

Clean and Process Raw Dataset Using Python

**Aim:**

To clean and preprocess a messy dataset by handling missing data, correcting data types, removing duplicates and outliers, and exporting the clean data.

**Requirement:**

* Python 3.x
* pandas, numpy, and scipy libraries
* A raw CSV file with messy or incomplete data

**Procedure:**

* Read the CSV using pandas.
* Remove duplicates.
* Fill missing values.
* Convert data types.
* Remove outliers using IQR.
* Save to a new file.

**Output:**

A cleaned dataset saved as a new Excel or CSV file.

**Conclusion:**

Data cleaning is essential for accurate data analysis. Using Python libraries like pandas and scipy, large and unstructured data can be efficiently cleaned and prepared for analysis or modeling.

**QUESTION-5**

Predictive Modeling with Linear Regression

**Aim:**

To develop and visualize a linear regression model to predict salary based on years of experience using Python.

**Requirement:**

* Python
* pandas, scikit-learn, matplotlib libraries
* A dataset with “Years of Experience” and “Salary”

**Procedure:**

* Load and split data using train\_test\_split.
* Fit model using LinearRegression().
* Predict and plot regression line.
* Visualize data and model fit using matplotlib.

**Output:**

A scatter plot of the data along with a red regression line showing the predicted salary trend.

**Conclusion:**

This task demonstrates the effectiveness of linear regression for making predictions. It also helps visualize the relationship between dependent and independent variables.

**QUESTION-6**

**Aim:**

To train a single-layer perceptron model using Keras on a classification dataset and evaluate its accuracy to understand basic neural network functioning

**Requirement:**

* Python (Keras, sklearn, pandas)
* Dataset like Iris

**Procedure:**

* Load and prepare dataset.
* Build a perceptron model.
* Train and evaluate the model.

**Output:**

After training, the perceptron model achieved around 96% accuracy on the test data. This high performance indicates successful classification of input features and shows that even a simple neural network can effectively learn patterns from structured datasets like Iris.

**Conclusion:**

The perceptron model built using Keras achieved good accuracy on classification data. This example introduces neural networks in a beginner-friendly way, showing how AI models can learn patterns from data. It forms the base for deeper neural architectures used in advanced machine learning projects

**QUESTION-7**

**Aim:**

To implement a text preprocessing function in Python that cleans raw text data by removing unnecessary symbols, punctuation, and stopwords.

**Requirement:**

* Python
* string module
* Python with nltk installed and English stopword corpus downloaded

**Procedure:**

* Accept raw text input.
* Convert the text to lowercase for uniformity.
* Remove punctuation using the translate() function.
* Split text into individual words using split().
* Filter out common English stopwords using nltk.corpus.stopwords.

**Output:**

A cleaned list of meaningful words extracted from the input sentence, useful for natural language processing.

**Conclusion:**

Text preprocessing is a critical step in NLP. This function removes noise from the data and prepares it for tasks like sentiment analysis, classification, or topic modeling.