Q1. Explain the differences between Python 2 and Python 3.

**Print Statement:** In Python 2, print is treated as a statement, whereas in Python 3, it is treated as a function (e.g., print "Hello" in Python 2 vs print("Hello") in Python 3).

**Division of Integers:** In Python 2, dividing two integers using / results in an integer (e.g., 5/2 results in 2), whereas in Python 3, it results in a float (e.g., 5/2 results in 2.5). Use // in Python 3 for integer division.

**Unicode:** Python 3 uses Unicode by default for strings (e.g., "hello" is a Unicode string in Python 3), while Python 2 uses ASCII by default.

**xrange() vs range():** In Python 2, xrange() is used for generating numbers in a range as it is more memory-efficient than range(). In Python 3, range() behaves like xrange() in Python 2.

Q2.What is the Global Interpreter Lock (GIL) in Python, and how does it impact multi-threading?

The Global Interpreter Lock (GIL) is a mutex that protects access to Python objects, preventing multiple native threads from executing Python bytecodes at once. This means that even in a multi-threaded Python program, only one thread can execute Python code at a time. The GIL impacts multi-threading by limiting the effectiveness of CPU-bound threads, making Python less efficient for parallel processing in such cases. For I/O-bound operations, however, the GIL is less of an issue as threads spend a lot of time waiting for external events, allowing other threads to run.

Q3.How does Python's memory management work?

Python uses a private heap space to manage memory, where all Python objects and data structures are stored. The memory manager in Python handles the allocation of heap space for Python objects. Python also has an inbuilt garbage collector, which recycles all the unused memory to make it available for heap space. The garbage collector primarily uses reference counting and generational garbage collection to manage memory

Q4.Explain the concept of decorators in Python and provide an example of their usage.

Ans:-

**Decorators**: Decorators are a powerful tool in Python that allows you to modify the behavior of a function or method. They are functions that take another function and extend its behavior without explicitly modifying it.

**Example:-**

def my\_decorator(func):

def wrapper():

print("Something is happening before the function is called.")

func()

print("Something is happening after the function is called.")

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

say\_hello()

Q5.What is the purpose of virtual environments in Python, and how do you create and activate one?

Ans:-

**Purpose**: Virtual environments allow you to create isolated Python environments for different projects. This helps avoid conflicts between dependencies required by different projects.

**Creation**:

python3 -m venv myenv

myenv\Scripts\activate

source myenv/bin/activate

SQL

Q1..Explain the differences between INNER JOIN, LEFT JOIN, and RIGHT JOIN. Provide examples.

ANs:-

**INNER JOIN**: Returns only the rows that have matching values in both tables.

**LEFT JOIN**: Returns all rows from the left table and the matched rows from the right table. If no match is found, NULLs are returned for columns from the right table.

**RIGHT JOIN**: Returns all rows from the right table and the matched rows from the left table. If no match is found, NULLs are returned for columns from the left table.

**Example**:

SELECT a.name, b.order\_id

FROM customers a

INNER JOIN orders b

ON a.customer\_id = b.customer\_id;

Q2..What is normalization, and why is it important in database design?

Ans:-

**Normalization**: Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

**Importance**: It ensures that the database is free from insert, update, and delete anomalies and enhances query performance by minimizing redundancy and ensuring data consistency.

Q3.Write a SQL query to retrieve all records from a "Customers" table where the "City" is

Ans:-

SELECT \* FROM Customers WHERE City = 'New York';

Q4.Explain the ACID properties in the context of database transactions.

Ans:-

**ACID**:

* **Atomicity**: Ensures that all operations within a transaction are completed successfully; if not, the transaction is aborted.
* **Consistency**: Guarantees that a transaction brings the database from one valid state to another.
* **Isolation**: Ensures that transactions are executed in isolation from each other.
* **Durability**: Ensures that once a transaction has been committed, it will remain committed even in the case of a system failure.

Q5.What is the primary key in a relational database, and why is it important?

Ans:-

**Primary Key**: A primary key is a unique identifier for each record in a database table.

**Importance**: It ensures that each record can be uniquely identified, which is crucial for establishing relationships between tables and for performing fast lookups.

Q1. Define text mining and its purpose in data analysis.

Ans:-

**Text Mining**: Text mining involves extracting meaningful information and insights from unstructured text data. It involves processes such as text classification, clustering, sentiment analysis, and keyword extraction.

**Purpose**: The purpose is to convert unstructured text into structured data for analysis, enabling businesses to derive actionable insights, identify patterns, and make informed decisions.

Q2.Describe TF-IDF (Term Frequency-Inverse Document Frequency) and its significance in text mining.

Ans:-

**TF-IDF**: TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents (corpus).

* **Term Frequency (TF)**: Measures how frequently a word appears in a document.
* **Inverse Document Frequency (IDF)**: Measures how important a word is by considering its rarity across the entire corpus.

**Significance**: TF-IDF is widely used in text mining and information retrieval to evaluate the relevance of a word in a document, helping to filter out common words that may not be meaningful.

Q3.Discuss the process of sentiment analysis and its applications in analysing opinions or emotions from text.

Ans:-

 **Sentiment Analysis**: Sentiment analysis involves determining the sentiment expressed in a piece of text, usually categorizing it as positive, negative, or neutral.

 **Process**:

1. **Text Preprocessing**: Cleaning and preparing the text data (e.g., removing stop words, tokenization).
2. **Feature Extraction**: Extracting relevant features from the text (e.g., TF-IDF, word embeddings).
3. **Modeling**: Applying machine learning models or lexicon-based approaches to classify the sentiment.

 **Applications**:

* **Business**: Understanding customer feedback, monitoring brand reputation.
* **Politics**: Analyzing public opinion on policy matters.
* **Marketing**: Gauging the effectiveness of marketing campaigns.