Q1. How do you define a custom function in Python? Provide a simple example of a function definition.

Ans:-

**Definition**: A custom function in Python is defined using the def keyword followed by the function name and parentheses. The body of the function contains the code that defines what the function does.

**Example**:

def greet(name):

print(f"Hello, {name}!")

Q2. Explain the concept of function parameters and arguments. How can you define functions that accept one or more input values?

Ans:-

**Function Parameters**: Parameters are the variables listed inside the parentheses in the function definition. They act as placeholders for the values that will be passed to the function when it is called.

**Arguments**: Arguments are the actual values passed to the function when it is called.

**Example:-**

**def add\_numbers(a, b):**

**return a + b**

Q3. What is the purpose of the 'return' statement in a Python function?

Ans:-

The return statement is used to exit a function and return a value to the caller. It can be used to pass a result back to the code that called the function.

def square(x):

return x \* x

Q4. .Describe how to call (or invoke) a function in Python. Provide an example of calling a custom function.

Ans:-

**Calling a Function**: To call a function, you simply use its name followed by parentheses. If the function requires arguments, you pass them inside the parentheses.

**Example:-**

def multiply(a, b):

return a \* b

result = multiply(3, 4)

print(result) # Outputs: 12

Q5. .How can you set default values for function parameters in Python? When and why might you use default parameters?

Ans:- **Setting Default Values**: Default values can be assigned to parameters by using the assignment operator = in the function definition.

**Usage**: Default parameters are used when you want to allow a function to be called with fewer argudef greet(name, message="Hello"):

print(f"{message}, {name}!")

greet("Alice") # Outputs: Hello, Alice!

greet("Bob", "Good morning") # Outputs: Good morning, Bob!ments than it is defined to accept. They provide a default value if no argument is passed.

SQL

Q1. What is an SQL trigger, and why is it used in database management? Provide a basic definition of triggers.

Ans:-

Definition: An SQL trigger is a predefined action that is automatically executed (or "triggered") in response to a specific event on a particular table or view in a database.

Purpose: It is used to enforce business rules, maintain data integrity, and automate certain tasks without requiring explicit intervention from the application or user.

Q2. Describe the two main types of SQL triggers: "before" triggers and "after" triggers. How do they differ in their execution timing?

Ans:- **Before Triggers**: Execute before the triggering event (such as an INSERT, UPDATE, or DELETE operation) occurs. They are often used to validate data or modify it before it is saved to the database.

**After Triggers**: Execute after the triggering event has occurred. They are used for actions like logging changes or updating related tables after the main operation is completed.

Q3. What types of database events can trigger the execution of trigger functions? Provide examples of events that can be associated with triggers.

Ans:-

**Types of Events**:

* **INSERT**: Triggered when a new record is added to a table.
* **UPDATE**: Triggered when an existing record is modified.
* **DELETE**: Triggered when a record is removed from a table.

**Examples**:

* Automatically log changes to an employee's salary after an UPDATE operation.
* Ensure that certain fields are not empty before allowing an INSERT.

Q4. Explain the SQL syntax for creating a trigger. Include the necessary components like the trigger name, timing, event, and the SQL statement to execute.

Ans:-

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE | DELETE}

ON table\_name

FOR EACH ROW

BEGIN

-- SQL statements

END;

**Components**:

* trigger\_name: The name of the trigger.
* {BEFORE | AFTER}: Specifies when the trigger should execute.
* {INSERT | UPDATE | DELETE}: The event that will cause the trigger to execute.
* table\_name: The table on which the trigger is defined.
* FOR EACH ROW: Specifies that the trigger should execute for each row affected by the event.
* BEGIN ... END: The block containing the SQL statements to execute when the trigger is fired.

Q5. How can you specify whether a trigger should execute before or after the associated event? What keywords are used to control the timing of trigger execution?

Ans:-

**Specifying Timing**: You specify the timing of a trigger using the keywords BEFORE or AFTER in the trigger definition.

**Keywords**:

* BEFORE: The trigger runs before the event.
* AFTER: The trigger runs after the event.