MODULE 5 PYTHON ASSIGNMENT

1. publisher, year\_of\_publication, and price. Insert five records into the table.

Ans. CREATE TABLE books (

book\_id INT PRIMARY KEY,

title VARCHAR(255),

author VARCHAR(255),

publisher VARCHAR(255),

year\_of\_publication INT,

price DECIMAL(10, 2)

);

1. Create a table members in library\_db with columns: member\_id, member\_name, date\_of\_membership, and email. Insert five records into this table.

Ans. CREATE TABLE members (

member\_id INT PRIMARY KEY,

member\_name VARCHAR(255),

date\_of\_membership DATE,

email VARCHAR(255)

);

1. Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.

Ans. SELECT \*

FROM members

WHERE date\_of\_membership < '2022-01-01'

ORDER BY date\_of\_membership;

1. Write SQL queries to display the titles of books published by a specific author. Sort the results by year\_of\_publication in descending order.

Ans. SELECT

title

FROM books

WHERE author = 'George Orwell'

ORDER BY year\_of\_publication DESC;

1. : Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.

Ans. CREATE TABLE books (

book\_id INT PRIMARY KEY,

title VARCHAR(255),

author VARCHAR(255),

publisher VARCHAR(255),

year\_of\_publication INT,

price DECIMAL(10, 2) CHECK (price > 0)

);

1. : Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.

Ans. ALTER TABLE members

ADD CONSTRAINT unique\_email UNIQUE (email);

1. : Create a table authors with the following columns: author\_id, first\_name, last\_name, and country. Set author\_id as the primary key.

And. : CREATE TABLE authors (

author\_id INT PRIMARY KEY,

first\_name VARCHAR(100),

last\_name VARCHAR(100),

country VARCHAR(100)

);

1. : Create a table publishers with columns: publisher\_id, publisher\_name, contact\_number, and address. Set publisher\_id as the primary key and contact\_number as unique.

Ans. CREATE TABLE publishers (

publisher\_id INT PRIMARY KEY,

publisher\_name VARCHAR(255),

contact\_number VARCHAR(20) UNIQUE,

address VARCHAR(255)

);

1. : Add a new column genre to the books table. Update the genre for all existing records.

Ans. UPDATE books

SET genre = CASE

WHEN title = 'To Kill a Mockingbird' THEN 'Fiction'

WHEN title = '1984' THEN 'Dystopian'

WHEN title = 'The Great Gatsby' THEN 'Classic'

WHEN title = 'Pride and Prejudice' THEN 'Romance'

WHEN title = 'The Catcher in the Rye' THEN 'Coming-of-age'

ELSE 'Unknown'

END;

1. : Modify the members table to increase the length of the email column to 100 characters.

Ans. ALTER TABLE members

MODIFY COLUMN email VARCHAR(100);

1. Drop the publishers table from the database after verifying its structure.

Ans. DESCRIBE publishers;

1. Create a backup of the members table and then drop the original members table.

Ans. CREATE TABLE members\_backup AS

SELECT \* FROM members;

DROP TABLE members;

1. Insert three new authors into the authors table, then update the last name of one of the authors.

And. INSERT INTO authors (author\_id, first\_name, last\_name, country)

VALUES

(1, 'George', 'Orwell', 'United Kingdom'),

(2, 'J.K.', 'Rowling', 'United Kingdom'),

(3, 'Harper', 'Lee', 'United States');

1. Delete a book from the books table where the price is higher than $100.

Ans. DELETE FROM books

WHERE price > 100;

1. : Update the year\_of\_publication of a book with a specific book\_id.

Ans. UPDATE books

SET year\_of\_publication = 2020

WHERE book\_id = 1;

1. Increase the price of all books published before 2015 by 10%.

Ans. UPDATE books

SET price = price \* 1.10

WHERE year\_of\_publication < 2015;

1. : Remove all members who joined before 2020 from the members table.

Ans. DELETE FROM members

WHERE date\_of\_membership < '2020-01-01';

1. Delete all books that have a NULL value in the author column.

Ans. DELETE FROM books

WHERE author IS NULL;

1. : Write a query to retrieve all books with price between $50 and $100.

Ans. SELECT \*

FROM books

WHERE price BETWEEN 50 AND 100;

1. : Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.

Ans. SELECT \*

FROM books

ORDER BY author ASC

LIMIT 3;

1. : Grant SELECT permission to a user named librarian on the books table.

Ans. GRANT SELECT ON books TO librarian;

1. Grant INSERT and UPDATE permissions to the user admin on the members table.

Ans. GRANT INSERT, UPDATE ON members TO admin;

1. Revoke the INSERT privilege from the user librarian on the books table.

Ans. REVOKE INSERT ON books FROM librarian;

1. Revoke all permissions from user admin on the members table.

Ans. REVOKE ALL PRIVILEGES ON members FROM admin;

1. Use COMMIT after inserting multiple records into the books table, then make another insertion and perform a ROLLBACK.

Ans. START TRANSACTION;

INSERT INTO books (book\_id, title, author, publisher, year\_of\_publication, price, genre)

VALUES

(6, 'Brave New World', 'Aldous Huxley', 'Chatto & Windus', 1932, 18.00, 'Dystopian'),

(7, 'Moby-Dick', 'Herman Melville', 'Harper & Brothers', 1851, 22.50, 'Adventure');

COMMIT;

1. Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

Ans. -- Start a transaction

START TRANSACTION;

-- Set a savepoint before making changes

SAVEPOINT before\_update;

-- Perform some updates

UPDATE members

SET email = 'updated.email@example.com'

WHERE member\_id = 2;

UPDATE members

SET member\_name = 'Updated Name'

WHERE member\_id = 3;

-- Roll back to the savepoint to undo the updates

ROLLBACK TO SAVEPOINT before\_update;

-- Commit the transaction (without the updates)

COMMIT;

1. Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.

Ans. SELECT

books.title,

CONCAT(authors.first\_name, ' ',

authors.last\_n

ame) AS author\_name

FROM books

INNER JOIN authors ON books.author\_id = authors.author\_id;

1. Group books by genre and display the total number of books in each genre.

Ans. SELECT genre, COUNT(\*) AS total\_books

FROM books

GROUP BY genre;

1. Group members by the year they joined and find the number of members who joined each year.

Ans. SELECT YEAR(date\_of\_membership) AS join\_year, COUNT(\*) AS total\_members

FROM members

GROUP BY YEAR(date\_of\_membership)

ORDER BY join\_year;

1. : Write a stored procedure to retrieve all books by a particular author.

Ans. DELIMITER //

CREATE PROCEDURE GetBooksByAuthor(IN author\_name VARCHAR(255))

BEGIN

SELECT \*

FROM books

WHERE author = author\_name;

END //

DELIMITER ;

1. Write a stored procedure that takes book\_id as an argument and returns the price of the book.

Ans. DELIMITER //

CREATE PROCEDURE GetBookPrice(IN input\_book\_id INT, OUT book\_price DECIMAL(10,2))

BEGIN

SELECT price INTO book\_price

FROM books

WHERE book\_id = input\_book\_id;

END //

DELIMITER ;

1. Create a view to show only the title, author, and price of books from the books table.

Ans CREATE VIEW books\_view AS

SELECT title, author, price

FROM books;

1. : Create a view to display members who joined before 2020.

Ans. CREATE VIEW members\_before\_2020 AS

SELECT member\_id, member\_name, date\_of\_membership, email

FROM members

WHERE date\_of\_membership < '2020-01-01';

1. Create a trigger to automatically update the last\_modified timestamp of the books table whenever a record is updated.

Ans. DELIMITER //

CREATE TRIGGER update\_last\_modified

BEFORE UPDATE ON books

FOR EACH ROW

BEGIN

SET NEW.last\_modified = NOW();

END //

DELIMITER ;

1. Create a trigger that inserts a log entry into a log\_changes table whenever a DELETE operation is performed on the books table.

ans.

CREATE TRIGGER log\_delete\_books

AFTER DELETE ON books

FOR EACH ROW

BEGIN

INSERT INTO log\_changes (action\_type, table\_name, record\_id, action\_timestamp)

VALUES ('DELETE', 'books', OLD.book\_id, NOW());

END //

DELIMITER ;

1. Write a PL/SQL block to insert a new book into the books table and display a confirmation message.

Ans. BEGIN

-- Insert a new book into the books table

INSERT INTO books (book\_id, title, author, publisher, year\_of\_publication, price, genre)

VALUES (9, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Scribner', 1925, 15.99, 'Classic');

-- Display a confirmation message

DBMS\_OUTPUT.PUT\_LINE('New book inserted successfully.');

EXCEPTION

-- Handle any potential exceptions

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

1. Write a PL/SQL block to display the total number of books in the books table.

Ans. DECLARE

total\_books INT; -- Declare a variable to hold the count

BEGIN

-- Retrieve the total number of books from the books table

SELECT COUNT(\*) INTO total\_books

FROM books;

-- Display the total number of books

DBMS\_OUTPUT.PUT\_LINE('Total number of books: ' || total\_books);

EXCEPTION

-- Handle any potential exceptions

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

1. : Write a PL/SQL block to declare variables for book\_id and price, assign values, and display the results.

Ans . DECLARE

book\_id INT := 101; -- Declare and assign value to book\_id

price DECIMAL(10,2) := 29.99; -- Declare and assign value to price

BEGIN

-- Display the values of book\_id and price

DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || book\_id);

DBMS\_OUTPUT.PUT\_LINE('Price: $' || price);

END;

/

1. Write a PL/SQL block using constants and perform arithmetic operations on book prices.

Ans . DECLARE

-- Declare constants for price adjustments

CONSTANT DISCOUNT\_PERCENTAGE CONSTANT DECIMAL(5,2) := 10; -- Discount percentage of 10%

CONSTANT TAX\_RATE CONSTANT DECIMAL(5,2) := 8.5; -- Tax rate of 8.5%

-- Declare variables for book prices and final calculations

original\_price DECIMAL(10,2)\_

1. Write a PL/SQL block using IF-THEN-ELSE to check if a book's price is above $100 and print a message accordingly.

Ans. DECLARE

book\_price DECIMAL(10,2) := 120.50; -- Declare and assign the book price

BEGIN

-- Check if the price is above $100

IF book\_price > 100 THEN

DBMS\_OUTPUT.PUT\_LINE('The price of the book is above $100.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The price of the book is $100 or below.');

END IF;

END;

/

1. Use a FOR LOOP in PL/SQL to display the details of all books one by one.

Ans. DECLARE

-- Declare a record type variable to hold book details

CURSOR book\_cursor IS

SELECT book\_id, title, author, price

FROM books;

-- Declare a variable to hold the book record

book\_record book\_cursor%ROWTYPE;

BEGIN

-- Open the cursor and loop through each book record

FOR book\_record IN book\_cursor LOOP

-- Display the book details using DBMS\_OUTPUT

DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || book\_record.book\_id);

DBMS\_OUTPUT.PUT\_LINE('Title: ' || book\_record.title);

DBMS\_OUTPUT.PUT\_LINE('Author: ' || book\_record.author);

DBMS\_OUTPUT.PUT\_LINE('Price: $' || book\_record.price);

DBMS\_OUTPUT.PUT\_LINE('---------------------------');

END LOOP;

END;

/

1. Write a PL/SQL block using an explicit cursor to fetch and display all records from the members table.

Ans. DECLARE

-- Declare the explicit cursor to fetch member details

CURSOR member\_cursor IS

SELECT member\_id, member\_name, date\_of\_membership, email

FROM members;

-- Declare a variable to hold the member record

member\_record member\_cursor%ROWTYPE;

BEGIN

-- Open the cursor and fetch the records

OPEN member\_cursor;

-- Loop throu

1. : Create a cursor to retrieve books by a particular author and display their titles.

Ans. DECLARE

-- Declare a variable for the author's name

author\_name VARCHAR2(100) := 'J.K. Rowling'; -- Example: change to the author you're interested in

-- Declare a cursor to retrieve books by the particular author

CURSOR book\_cursor IS

SELECT title

FROM books

WHERE author = author\_name;

-- Declare a variable to hold the book title

book\_record book\_cursor%ROWTYPE;

BEGIN

-- Open the cursor to fetch books by the specified author

OPEN book\_cursor;

-- Loop through each record fetched by the cursor

LOOP

FETCH book\_cursor INTO book\_record; -- Fetch a book record into book\_record

EXIT WHEN book\_cursor%NOTFOUND; -- Exit\_

1. : Perform a transaction that includes inserting a new member, setting a SAVEPOINT, and rolling back to the savepoint after making updates.

And. DECLARE

-- Variables for new member details

new\_member\_id INT := 123;

new\_member\_name VARCHAR2(100) := 'John Doe';

new\_member\_email VARCHAR2(100) := 'johndoe@example.com';

new\_member\_date DATE := SYSDATE;

BEGIN

-- Insert a new member into the members table

INSERT INTO members (member\_id, member\_name, email, date\_of\_membership)

VALUES (new\_member\_id, new\_member\_name, new\_member\_email, new\_member\_date);

-- Set a SAVEPOINT after the insert

SAVEPOINT before\_update;

1. : Use COMMIT after successfully inserting multiple books into the books table, then use ROLLBACK to undo a set of changes made after a savepoint.

Ans. Variables for new book details

new\_book\_id INT := 1;

new\_title VARCHAR2(100);

new\_author VARCHAR2(100) := 'Author Name';

new\_price DECIMAL(10,2) := 29.99;

new\_year INT := 2023;

BEGIN

-- Inserting multiple books into the books table

FOR i IN 1..5 LOOP

new\_title := 'Book ' || i; -- Create a unique title for each book

INSERT INTO books (book\_id, title, author, price, year\_of\_publication)

VALUES (new\_book\_id + i, new\_title, new\_author, new\_price, new\_year);

END LOOP;

-- COMMIT after successfully inserting the books

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Books successfully inserted and committed.');

-- Set a SAVEPOINT before making further changes

SAVEPOINT before\_update;

-- Performing some updates after the savepoint (e.g., increasing the price of all books)

UPDATE books

SET price = price + 5

WHERE year\_of\_publication = 2023;

-- Rollback to the savepoint to undo the price changes

ROLLBACK TO SAVEPOINT before\_update;

DBMS\_OUTPUT.PUT\_LINE('Price updates rolled back to the savepoint.');

-- Optionally, commit the transaction if needed (committing the insert only)

COMMIT;

EXCEPTION

WHEN OTHERS THEN

-- In case of an error, rollback the entire transaction

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;