Module-4: Introduction to DBMS

Lab exercise(Extra)

* 1.INTRODUCTION TO SQL:-

🡪Lab 3: Create a database called library\_db and a table books with columns: book\_id, title, author, publisher, year\_of\_publication, and price. Insert five records into the table.

Query:-

create database library\_db;//create database

use library\_db;//use it

create table books(book\_id int primary key,title varchar(20),author varchar(20),publisher varchar(20),year\_of\_publication year,price int);//for books table

INSERT INTO books VALUES

(1, 'C Programming', 'Dennis Ritchie', 'Prentice Hall', 1988, 450),

(2, 'Data Structures', 'Seymour Lipschutz', 'McGraw-Hill', 1990, 520),

(3, 'DBMS Concepts', 'Korth', 'McGraw-Hill', 2006, 600),

(4, 'Let Us C', 'Yashwant Kanetkar', 'BPB', 2010, 300),

(5, 'Operating System', 'Galvin', 'Wiley', 2014, 750);

-------------------------------------------------------------------------------------

🡪Lab 4: Create a table members in library\_db with columns: member\_id, member\_name, date\_of\_membership, and email. Insert five records into this table.

Query:-

create table members(member\_id int primary key,member\_name varchar(20),date\_of\_membership date,email varchar(35) unique not null);

INSERT INTO members VALUES

(1, 'Yogesh Patel', '2022-01-15', 'patelyogesh26042005@gmail.com'),

(2, 'Ayan Mansuri', '2021-11-10', 'ayanmansuri@gmail.com'),

(3, 'Jay Mokariya', '2023-05-25', 'jaymokariya@gmail.com'),

(4, 'Kuntal Nayee', '2020-07-30', 'kuntal@gmail.com'),

(5, 'Yash Parmar', '2024-03-18', 'yash@gmail.com');

------------------------------------------------------------------------------------

* 2.SQL SYNTAX-

🡪Lab 3: Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.

Query:-

select \* from members where date\_of\_membership < '2022-01-01';

🡪Lab 4: Write SQL queries to display the titles of books published by a specific author. Sort the results by year\_of\_publication in descending order.

Query:-

Select title ,author,year\_of\_publication from books order by

year\_of\_publication desc;

------------------------------------------------------------------------------------

* Que-3. SQL CONSTRAINTS:-

🡪Lab 3: Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.

Query:

alter table books add constraint check(price>0);

-------------------------------------------------------------------------------------

🡪 Lab 4: Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.

Query:-

alter table members add constraint unique (email);

-------------------------------------------------------------------------------------

* 4. Main SQL Commands and Sub-commands(DDL):-

🡪Lab 3: Create a table authors with the following columns: author\_id, first\_name, last\_name, and country. Set author\_id as the primary key.

Query:

create table authors(author\_id int primary key,first\_name varchar(20),last\_name varchar(20),country varchar(20))

-------------------------------------------------------------------------------------

🡪Lab 4: 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.

Query:-

create table publisher(publisher\_id int primary key ,publisher\_name varchar(20),contact\_number bigint unique,address varchar(50))

-------------------------------------------------------------------------------------

* 5. ALTER COMMAND:-

🡪Lab 3: Add a new column genre to the books table. Update the genre for all existing records.

Query:

alter table books add genre varchar(20);

UPDATE books

SET genre = 'Programming' where book\_id=1;

UPDATE books

SET genre = 'Programming' where book\_id=2;

UPDATE books

SET genre = 'Programming' where book\_id=3;

UPDATE books

SET genre = 'Programming' where book\_id=4;

UPDATE books

SET genre = 'Programming' where book\_id=5;

🡪Lab 4: Modify the members table to increase the length of the email column to 100 characters.

Query:-

alter table members modify email varchar(100);

desc members;

-------------------------------------------------------------------------------------

* 6. DROP COMMAND:-

🡪Lab 3: Drop the publishers table from the database after verifying its structure.

Query:-

desc publisher;

drop table publisher;

🡪Lab 4: Create a backup of the members table and then drop the original members table.

Query:

CREATE TABLE members\_backup AS

SELECT \* FROM members;

drop table members;

select \* from members\_backup;

-------------------------------------------------------------------------------------

* 7. Data Manipulation Language (DML):-

🡪Lab 4: Insert three new authors into the authors table, then update the last name of one of the authors.

desc authors;

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

VALUES

(201, 'John', 'Smith', 'USA'),

(202, 'Emily', 'Clark', 'UK'),

(203, 'David', 'Brown', 'Canada');

update authors set last\_name='patel' where author\_id=203;

select \* from authors;

🡪 Lab 5: Delete a book from the books table where the price is higher than $100.

SET SQL\_SAFE\_UPDATES = 0;

delete from books where price >100;

set SQL\_SAFE\_UPDATES=1;

select \* from books;

-------------------------------------------------------------------------------------

* 8.Update Command:-

🡪Lab 3: Update the year\_of\_publication of a book with a specific book\_id.

-- first add data in books

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

VALUES

(1, 'Harry Potter', 'J.K. Rowling', 250.00, 'Fantasy', 1997, 'Bloomsbury'),

(2, 'Great Gatsby', 'Fitzgerald', 90.00, 'Classic', 1925, 'Scribners'),

(3, 'Mockingbird', 'Harper Lee', 120.00, 'Fiction', 1960, 'Lippincott'),

(4, 'Brief History', 'S. Hawking', 300.00, 'Science', 1988, 'Bantam Books'),

(5, 'The Hobbit', 'Tolkien', 180.00, 'Fantasy', 1937, 'Allen & Unwin');

update books set year\_of\_publication=1996 where book\_id=1;

update books set year\_of\_publication=1926 where book\_id=2;

update books set year\_of\_publication=1966 where book\_id=3;

update books set year\_of\_publication=1986 where book\_id=4;

update books set year\_of\_publication=1936 where book\_id=5;

select \* from books;

🡪Lab 4: Increase the price of all books published before 2015 by 10%.

set sql\_safe\_updates=0;

update books set price=price \*1.10 where year\_of\_publication < 2015;

set sql\_safe\_updates=1;

select \* from books;

-------------------------------------------------------------------------------------

* 9. Delete Command:-

🡪Lab 3: Remove all members who joined before 2020 from the members table.

set sql\_safe\_updates=0;

delete from members\_backup where date\_of\_membership < '2020-01-01';

set sql\_safe\_updates=1;

select \* from members\_backup;

------------------------------------------------------------------------------------

🡪Lab 4: Delete all books that have a NULL value in the author column.

update books set author=null where book\_id=1;

set sql\_safe\_updates=0;

delete from books where author is null;

set sql\_safe\_updates=1;

select \* from books;

-------------------------------------------------------------------------------------

* 10.Data Query Language(DQL):-

🡪Lab 4: Write a query to retrieve all books with price between $50 and $100.

select \* from books where price between 50 and 100;

🡪Lab 5: Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.

select \* from books order by author limit 3;

-------------------------------------------------------------------------------------

* 11. Data Control Language(DCL):-

🡪Lab 3: Grant SELECT permission to a user named librarian on the books table.

use library\_db;

create user 'librarian'@'localhost' identified by 'Mysql';

grant select on library\_db.books to 'librarian'@'localhost';

🡪Lab 4: Grant INSERT and UPDATE permissions to the user admin on the members table

create user 'admin1'@'localhost' identified by 'Mysql';

grant insert,update on members\_backup to 'admin1'@'localhost';

-------------------------------------------------------------------------------------

* 12.Revoke Command :-

🡪Lab 3: Revoke the INSERT privilege from the user librarian on the books table.

revoke insert on books from 'librarian'@'localhost';

🡪Lab 4: Revoke all permissions from user admin on the members table.

revoke select,insert,update,delete on library\_db.members\_backup from 'admin1'@'localhost';

-------------------------------------------------------------------------------------

* 13.Transaction Control language(TCL):-

🡪Lab 3: Use COMMIT after inserting multiple records into the books table, then make another insertion and perform a ROLLBACK.

truncate table books;

set autocommit=0;

start transaction;

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

VALUES

(1, 'Harry Potter', 'YOGESH PATEL', 250.00, 'Fantasy', 2010, 'Bloomsbury'),

(2, 'Great Gatsby', 'Yash Parmar', 90.00, 'Classic', 2005, 'Scribners'),

(3, 'Mockingbird', 'Jay Mokariya', 120.00, 'Fiction', 2007, 'Lippincott'),

(4, 'Brief History', 'Ayan Mansuri', 300.00, 'Science', 2011, 'Bantam Books'),

(5, 'The Hobbit', 'Om Patel', 180.00, 'Fantasy', 2015, 'Allen & Unwin');

commit;

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

VALUES

(6, 'Harry Potter', 'YOGESH PATEL', 250.00, 'Fantasy', 2010, 'Bloomsbury'),

(7, 'Great Gatsby', 'Yash Parmar', 90.00, 'Classic', 2005, 'Scribners'),

(8, 'Mockingbird', 'Jay Mokariya', 120.00, 'Fiction', 2007, 'Lippincott'),

(9, 'Brief History', 'Ayan Mansuri', 300.00, 'Science', 2011, 'Bantam Books'),

(10, 'The Hobbit', 'Om Patel', 180.00, 'Fantasy', 2015, 'Allen & Unwin');

rollback;

select \* from books;

set autocommit=1;

🡪Lab 4: Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

start transaction;

savepoint sp1;

select \* from books;

update books set price=price\*1.10 where book\_id=1;

update books set price=price\*1.10 where book\_id=2;

update books set price=price\*1.10 where book\_id=3;

update books set price=price\*1.10 where book\_id=4;

update books set price=price\*1.10 where book\_id=5;

rollback to sp1;

select \* from books;

-------------------------------------------------------------------------------------

* 14. SQL Join:-

🡪Lab 3: Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.

select first\_name,last\_name,country,author from books inner join authors ;

🡪Lab 4: Use a FULL OUTER JOIN to retrieve all records from the books and authors tables, including those with no matching entries in the other table.

--full join not support;

-----------------------------------------------------------------------------------

* 15. SQL Group By :-

🡪Lab 3: Group books by genre and display the total number of books in each genre.

select genre,count(book\_id) from books group by genre;

🡪Lab 4: Group members by the year they joined and find the number of members who joined each year.

select year(date\_of\_membership) as ye,count(member\_id) from members\_backup group by year(date\_of\_membership) order by ye;

------------------------------------------------------------------------------------

* 16. SQL Stored Procedure:-

🡪Lab 3: Write a stored procedure to retrieve all books by a particular author.

select \* from books;

delimiter //

create procedure Book\_aut(auth\_name varchar(20))

begin

select \* from books where author=auth\_name;

end//

delimiter ;

call Book\_aut('Yogesh Patel');

🡪Lab 4: Write a stored procedure that takes book\_id as an argument and returns the price of the book.

delimiter //

create procedure get\_price\_by\_book\_id(p\_bid int,out b\_price int )

begin

select price into b\_price from books where book\_id=p\_bid;

end//

delimiter ;

call get\_price\_by\_book\_id(2,@price);

select @price

-------------------------------------------------------------------------------------

* 17. SQL View:-

🡪Lab 3: Create a view to show only the title, author, and price of books from the books table.

create view book\_details

as

select title,author,price from books;

select \* from book\_details;

🡪Lab 4: Create a view to display members who joined before 2020.

create view member\_det

as

select \* from members\_backup where date\_of\_membership < '2020-01-01';

insert into members\_backup values(6,'Yogesh Patel','2019-04-26','patelyog@gmail.com');

select \* from member\_det;

-------------------------------------------------------------------------------------

* 18. SQL Trigger :-

🡪Lab 3: Create a trigger to automatically update the last\_modified timestamp of the books table whenever a record is updated.

alter table books add column last\_modified datetime default current\_timestamp;

select \* from books;

delimiter //

create trigger update\_last\_modified

before update on books

for each row

begin

SET new.last\_modified = current\_timestamp;

end //

delimiter ;

update books set price =250 where book\_id=1;

select \* from books;

🡪Lab 4: Create a trigger that inserts a log entry into a log\_changes table whenever a DELETE operation is performed on the books table.

create table log\_changes

as

select \* from books;

truncate log\_changes;

delimiter //

create trigger log\_entry

after delete on books

for each row

begin

insert into log\_changes

values(old.book\_id,old.title,old.author,old.publisher,old.year\_of\_publication,old.price,old.genre,old.last\_modified);

end//

delimiter ;

select \* from log\_changes;

delete from books where book\_id=5;

select \* from log\_changes;

-------------------------------------------------------------------------------------

* 19. Introduction To PL/SQL :-

🡪Lab 3: Write a PL/SQL block to insert a new book into the books table and display a confirmation message.

delimiter //

create procedure insert\_book()

begin

declare v\_book\_id int default 102;

declare v\_title varchar(20) default 'python programming';

declare v\_author varchar(20) default 'yogesh Patel';

declare v\_price int default 500;

insert into books(book\_id,title,author,price) values(v\_book\_id,v\_title,v\_author,v\_price);

select concat('Book id= ',v\_book\_id,' Book title= ',v\_title,' author = ',v\_author,' Price = ',v\_price,'inserted successfully') As message;

end//

delimiter ;

call insert\_book();

🡪Lab 4: Write a PL/SQL block to display the total number of books in the books table.

delimiter //

create procedure total\_book()

begin

declare total\_cou\_book int;

select count(book\_id) into total\_cou\_book from books ;

select concat('total books= ',total\_cou\_book) as output;

end//

delimiter ;

call total\_book();

* 20. PL/SQL Syntax :-

🡪Lab 3: Write a PL/SQL block to declare variables for book\_id and price, assign values, and display the results.

delimiter //

create procedure did\_res()

begin

declare p\_book\_id int default 110;

declare p\_price int default 200;

select concat('book\_id= ',p\_book\_id,' price= ',p\_price) as output;

end//

delimiter ;

call did\_res();

🡪Lab 4: Write a PL/SQL block using constants and perform arithmetic operations on book prices.

DELIMITER //

CREATE PROCEDURE book\_price\_arithmetic()

BEGIN

-- Simulating constants

DECLARE BASE\_PRICE DECIMAL(8,2) DEFAULT 250.00;

DECLARE TAX\_RATE DECIMAL(5,2) DEFAULT 0.10; -- 10%

DECLARE DISCOUNT DECIMAL(5,2) DEFAULT 0.05; -- 5%

-- Result variables

DECLARE final\_price DECIMAL(8,2);

DECLARE discounted\_price DECIMAL(8,2);

-- Calculate final price after tax

SET final\_price = BASE\_PRICE + (BASE\_PRICE \* TAX\_RATE);

-- Calculate discounted price

SET discounted\_price = final\_price - (final\_price \* DISCOUNT);

-- Display the results

SELECT

CONCAT('Base Price: ₹', BASE\_PRICE) AS base,

CONCAT('Final Price after Tax: ₹', final\_price) AS after\_tax,

CONCAT('Price after Discount: ₹', discounted\_price) AS after\_discount;

END//

DELIMITER ;

call book\_price\_arithmetic();

-------------------------------------------------------------------------------------

* 21. PL/SQL Control Structure :-

🡪Lab 3: Write a PL/SQL block using IF-THEN-ELSE to check if a book's price is above $100 and print a message accordingly.

delimiter //

create procedure p\_if(in p\_book\_id int)

begin

declare p\_price int default 100;

select price into p\_price from books where book\_id=p\_book\_id;

if p\_price>100

then select concat('book price is greter than 100');

else

select concat('book price is less than 100');

end if;

end//

delimiter ;

call p\_if(2);

🡪Lab 4: Use a FOR LOOP in PL/SQL to display the details of all books one by one.

delimiter //

create procedure lop()

begin

declare v\_title varchar(20);

declare v\_author varchar(20);

declare v\_price int;

declare done int default false;

declare det\_cur cursor for

select title,author,price from books;

declare continue handler for not found set done=true;

open det\_cur;

read\_loop:loop

fetch det\_cur into v\_title,v\_author,v\_price ;

if done then

leave read\_loop;

end if;

select concat('Title= ',v\_title) as info1,

concat('author= ',v\_author)as info2,

concat('price= ',v\_price) as info3;

end loop;

close det\_cur;

end//

delimiter ;

call lop();

-------------------------------------------------------------------------------------

* 22. SQL Cursor :-

🡪Lab 3: Write a PL/SQL block using an explicit cursor to fetch and display all records from the members table.

delimiter //

create procedure sp\_cur1()

begin

declare v\_mem\_name varchar(20);

declare v\_d\_o\_m date;

declare v\_email varchar(100);

declare done int default false;

declare cur1 cursor for

select member\_name,date\_of\_membership,email from members\_backup;

declare continue handler for not found set done=true;

open cur1;

read\_loop:loop

fetch cur1 into v\_mem\_name,v\_d\_o\_m,v\_email ;

if done then

leave read\_loop;

end if;

select concat('Member name= ',v\_mem\_name) as info1,

concat('date of join= ',v\_d\_o\_m) as info2,

concat('email= ',v\_email);

end loop;

close cur1;

end//

delimiter ;

call sp\_cur1();

🡪Lab 4: Create a cursor to retrieve books by a particular author and display their titles.

delimiter //

create procedure sp\_cur2(in aut\_name varchar(20))

begin

declare done int default false;

declare v\_title varchar(20);

declare cur2 cursor for

select title from books where author=aut\_name;

declare continue handler for not found set done=true;

open cur2;

read\_loop:loop

fetch cur2 into v\_title;

if done then

leave read\_loop;

end if;

select concat('Title = ',v\_title) as tit;

end loop;

close cur2;

end//

delimiter ;

call sp\_cur2('Yogesh Patel');

------------------------------------------------------------------------------------

* 23. Rollback And Commit Savepoint :-

🡪Lab 3: Perform a transaction that includes inserting a new member, setting a SAVEPOINT, and rolling back to the savepoint after making updates.

start transaction;

savepoint sp1;

insert into members\_backup values(110,'daksh patidar','2025-07-22','Daksh@gmail.com');

savepoint sp2;

set sql\_safe\_updates=0;

update members\_backup set member\_id=7 where member\_name='daksh patidar';

rollback to sp2;

select \* from members\_backup;

🡪Lab 4: Use COMMIT after successfully inserting multiple books into the books table, then use ROLLBACK to undo a set of changes made after a savepoint.

start transaction;

insert into books(book\_id,title,author,price) values(201,'science','yash parmar',200),(202,' social science','Jay Mokariya',100);

commit;

start transaction;

savepoint sp4;

insert into books(book\_id,title,author,price) values(203,'Maths','yogesh patel',200),(204,' history','Ayan MAnsuri',100);

rollback to sp4;

commit;

select \* from books;

-------------------------------------------------------------------------------------