# 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;
```

---