Module 4 – Introduction to DBMS

Introduction to SQL

Theory Questions:

1. What is SQL, and why is it essential in database management?

SQL (Structured Query Language) is a standard programming language designed for managing and manipulating relational databases

2. Explain the difference between DBMS and RDBMS.

DBMS (Database Management System) is a broad term for software managing data, while RDBMS (Relational Database Management System) is a specific type of DBMS that organizes data into structured, related tables (rows and columns) using a relational models and sql, offering better data integrity, fewer redundancies, and complex query capabilities compared to older DBMS types (hierarchical/network) that might store data in files or simpler structures

3. Describe the role of SQL in managing relational databases.

SQL (Structured Query Language) is the standard language for defining, manipulating, and controlling data within a relational database management system (RDBMS)

4. What are the key features of SQL?

SQL's key features include its role as a standardized language for database interaction, offering powerful data retrieval (SELECT), manipulation (INSERT, UPDATE, DELETE), and definition (CREATE, ALTER, DROP) through DDL and DML, along with robust transaction management, data integrity enforcement (constraints, keys), security controls, and advanced capabilities like joins, sub-queries, and stored procedures.

LAB EXERCISES:

Lab 1: Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.

create table student(student_id int primary key auto_increment, student_name varchar(50),age int,class int,address varchar(100));

Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.

```
INSERT INTO student (student_name, age, class, address) VALUES
   -> ('Alice Johnson', 14, 9, '123 Maple St, Springfield'),
   -> ('Bob Smith', 15, 10, '456 Oak Ave, Riverdale'),
   -> ('Charlie Davis', 13, 8, '789 Pine Rd, Lakeshore'),
   -> ('Diana Prince', 14, 9, '101 Cedar Ln, Hill Valley'),
   -> ('Ethan Hunt', 16, 11, '202 Birch Blvd, Mission City');
```

2. SQL Syntax

Theory Questions:

1. What are the basic components of SQL syntax?

The basic components of SQL syntax consist of **keywords**, **clauses**, and other elements like operators and identifiers that are combined to form **SQL statements** (queries)

2. Write the general structure of an SQL SELECT statement.

```
SELECT column1, column2, ...
FROM table_name;
```

3. Explain the role of clauses in SQL statements.

SQL clauses are specific components of an SQL statement that define the operations and constraints applied to the data, allowing users to filter, sort, group, and otherwise manipulate records

LAB EXERCISES:

Lab 1: Write SQL queries to retrieve specific columns (student_name and age) from the students table.

```
select student_name,age from student;
```

Lab 2: Write SQL queries to retrieve all students whose age is greater than 10

```
select * from student where age>10;
```

3. SQL Constraints

Theory Questions:

1. What are constraints in SQL? List and explain the different types of constraints.

Constraints in SQL are rules enforced on data columns in a table to limit the type of data that can go into a table, ensuring the **accuracy and reliability of the data** [1]. They are used to maintain the data integrity of a database.

NOT NULL

Ensures that a column cannot have a NULL value . This means every row must have a value for that column.

UNIQUE

Ensures that all values in a column are different and unique across all rows in the table

PRIMARY KEY

Uniquely identifies each row (record) in a database table . A primary key must contain UNIQUE values, and cannot contain NULL values

FOREIGN KEY

Uniquely identifies a row or a record in another database table . A foreign key links data between tables, maintaining referential integrity

CHECK

Ensures that all values in a column satisfy a specific condition

DEFAULT

Sets a default value for a column when no value is specified during an INSERT operation

2. How do PRIMARY KEY and FOREIGN KEY constraints differ?

PRIMARY KEY and FOREIGN KEY constraints differ in that a PRIMARY KEY uniquely identifies each record within its table, ensuring data integrity by enforcing that no two rows are identical and that the key column(s) contain no NULL values. In contrast, a FOREIGN KEY establishes a link between data in two tables by referring to the primary key of another table. Its purpose is to enforce referential integrity, ensuring that the values in the foreign key column (the "child" table) match an existing value in the primary key of the related table (the "parent" table), or are NULL, thereby linking related data across different tables

3. What is the role of NOT NULL and UNIQUE constraints?

NOT NULL and UNIQUE constraints enforce data integrity in databases by ensuring columns have values and are distinct, respectively, preventing bad data by disallowing NULL entries or duplicates, which is crucial for data accuracy, consistency, and reliable relationships between tables. NOT NULL makes a field mandatory, while UNIQUE ensures each entry in a column (or combination) is different, with PRIMARY KEY being a combination of both.

LAB EXERCISES:

Lab 1: Create a table teachers with the following columns: teacher_id (Primary Key),

teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

create table teachers(teacher_id int primary key auto_increment, teacher_name varchar(100) not null,subject varchar(100) not null,email varchar(50) unique key);

Lab 2: Implement a FOREIGN KEY constraint to relate the teacher_id from the

teachers table with the students table.

ALTER TABLE student ADD COLUMN teacher_id INT;

ALTER TABLE student

-> ADD CONSTRAINT fk_teacher

-> FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id);

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

Theory Questions:

1. Define the SQL Data Definition Language (DDL).

Create :it is used to create tables in sql

Alter :alter is used to make modifications in sql table like column rename , modify data type

DROP: The DROP command completely removes a table (or other objects like databases or indexes) from the database

Truncate : The TRUNCATE command removes all records from a table, but keeps the table structure (columns, data types, and constraints) intact for future use.

Call : The CALL command is used to execute a Stored Procedure. A stored procedure is a prepared SQL code snippet that you can save and reuse.

2. Explain the CREATE command and its syntax.

The CREATE command is a DDL statement used to build new database objects like tables or schemas. Its syntax requires the object type, a unique name, and defined parameters such as column names and data types. For instance, CREATE TABLE name (column type); establishes the structure for storing data. This command is crucial for defining the rules and organization of your database.

3. What is the purpose of specifying data types and constraints during table creation?

Specifying data types ensures the database understands what kind of information (integers, text, dates) to store, which optimizes memory and allows for accurate calculations. Constraints, such as UNIQUE or NOT NULL, act as automated rules that prevent invalid or duplicate data from entering the system. Together, they maintain data integrity by enforcing consistency and reliability across all records. By defining these rules upfront, you ensure the database remains organized and performs efficiently during complex queries

LAB EXERCISES:

Lab 1: Create a table courses with columns: course_id, course_name, and

course_credits. Set the course_id as the primary key.

create table courses (course_id int primary key, course_name varchar(255), course_credits int);

Lab 2: Use the CREATE command to create a database university_db.

create database university_db;

5. ALTER Command

Theory Questions:

1. What is the use of the ALTER command in SQL?

The ALTER command is used to modify the structure of an existing table without deleting it. It allows you to add, delete, or change columns and data types within your database schema. This statement is also essential for adding or removing constraints like primary or foreign keys. Essentially, it provides the flexibility to update your database design as requirements evolve over time.

2. How can you add, modify, and drop columns from a table using ALTER?

To add a column, use the alter table command followed by add and the column definition. To change a column's data type or properties, use the modify or alter column keyword. To remove a column entirely, use the drop column command followed by the specific column name. These operations allow you to restructure your data storage without losing existing records in the table.

alter table table_name add column_name datatype;

alter table table_name modify column_name new_datatype;

alter table table_name drop column column_name;

LAB EXERCISES:

Lab 1: Modify the courses table by adding a column course_duration using the ALTER command.

alter table courses add course_duration varchar(50);

Lab 2: Drop the course_credits column from the courses table.

alter table courses drop column course_credits;

6. DROP Command

Theory Questions:

1. What is the function of the DROP command in SQL?

The DROP command is a DDL statement used to permanently remove entire objects from a database. Unlike the DELETE command which removes rows, DROP deletes the structure, metadata, and all data. It can be applied to various database elements including tables, databases, views, and indexes. Once executed, this action cannot be undone in most systems, making it a powerful and final operation.

2. What are the implications of dropping a table from a database?

Dropping a table permanently deletes the entire structure and all stored records from the database. All associated metadata, such as indexes, triggers, and constraints, are also completely removed. Any existing foreign key relationships in other tables may be broken, leading to referential integrity errors. Because this action is irreversible without a backup, it should be executed with extreme caution.

LAB EXERCISES:

Lab 1: Drop the teachers table from the school_db database.

drop table teachers;

Lab 2: Drop the students table from the school_db database and verify that the table has been removed.

drop table students;

7. Data Manipulation Language (DML)

Theory Questions:

1. Define the INSERT, UPDATE, and DELETE commands in SQL.

insert into table_name (col1, col2) values (val1, val2);

update table_name set col1 = val1 where condition;

delete from table_name where condition;

2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

The WHERE clause is critical because it specifies exactly which records should be modified or removed. Without it, an UPDATE or DELETE command will apply to every single row in the entire table. This clause prevents accidental data loss by ensuring that only targeted information is affected. It acts as a safety filter, maintaining data accuracy by identifying rows based on unique criteria.

update table_name set column = value where condition;

Lab 1: Insert three records into the courses table using the INSERT command.

insert into courses (course_id, course_name, course_credits) values (1, 'math', 4), (2, 'physics', 3), (3, 'chemistry', 4);

Lab 2: Update the course duration of a specific course using the UPDATE command.

update courses set course_credits = '6 months' where course_id = 1;

Lab 3: Delete a course with a specific course_id from the courses table using the DELETE command.

delete from courses where course_id = 3;

8. Data Query Language (DQL)

Theory Questions:

1. What is the SELECT statement, and how is it used to query data?

The SELECT statement is the most common SQL command used to retrieve data from one or more tables. It allows you to specify which columns you want to see and which rows meet your search criteria. The data returned is stored in a result-set, which looks like a temporary table of your filtered information. This statement does not modify the underlying data; it only fetches and displays it for your review.

2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.

The WHERE clause filters data by including only the rows that meet a specific logical condition. The ORDER BY clause sorts the resulting data set in either ascending or descending alphabetical or numerical order. When used together, WHERE must appear before ORDER BY to filter the records before they are organized. These clauses are essential for narrowing down large datasets and presenting the most relevant information clearly.

LAB EXERCISES:

Lab 1: Retrieve all courses from the courses table using the SELECT statement.

```
select * from courses;
```

Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.

```
select * from courses order by course_duration desc;
```

Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

```
select * from courses order by course_duration desc limit 2;
```

9. Data Control Language (DCL)

Theory Questions:

1. What is the purpose of GRANT and REVOKE in SQL?

grant and revoke are part of data control language used to manage user permissions. grant gives specific privileges, like selecting or inserting data, to users or roles. revoke removes those previously granted privileges to maintain security and restrict access.

2. How do you manage privileges using these commands?

to manage privileges, you use the grant command to define what actions a user can perform and revoke to take those rights away. these commands control access at the database, table, or even column level.

LAB EXERCISES:

Lab 1: Create two new users user1 and user2 and grant user1 permission to SELECT

from the courses table.

create user 'user1'@'localhost' identified by 'password123'; create user 'user2'@'localhost' identified by 'password456'; grant select on courses to 'user1'@'localhost';

Lab 2: Revoke the INSERT permission from user1 and give it to user2.

revoke insert on courses from 'user1'@'localhost'; grant insert on courses to 'user2'@'localhost';

10. Transaction Control Language (TCL)

Theory Questions:

1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?

the purpose of commit is to permanently save all changes made during the current transaction to the database. rollback is used to undo changes that have not yet been saved, reverting the database to its previous consistent state.

2. Explain how transactions are managed in SQL databases.

transactions are managed using acid properties to ensure data integrity. they group multiple operations into a single unit of work that either succeeds completely or fails completely. the process is controlled using start transaction, commit to save changes, and rollback to discard them.

LAB EXERCISES:

Lab 1: Insert a few rows into the courses table and use COMMIT to save the changes.

insert into courses (course_id, course_name, course_credits) values (4, 'biology', 3), (5, 'history', 2); commit;

Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

start transaction; insert into courses (course_id, course_name, course_credits) values (6, 'art', 2), (7, 'music', 2); rollback;

Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back

specific changes.

savepoint sp1; update courses set course_credits = 5 where course_id = 1; rollback to sp1;

11. SQL Joins

Theory Questions:

1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

inner join returns records that have matching values in both tables. left join returns all records from the left table and the matched records from the right table. right join returns all records from the right table and the matched records from the left table. full outer join returns all records when there is a match in either left or right table.

2. How are joins used to combine data from multiple tables?

joins are used to combine rows from two or more tables based on a related column between them. this allows you to create a unified view of data that is stored across different relational structures. by specifying a join condition, usually an equality check on foreign and primary keys, the database aligns the corresponding records into a single result set.

LAB EXERCISES:

Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

create table departments (dept_id int, dept_name varchar(50)); create table employees (emp_id int, emp_name varchar(50), dept_id int); select employees.emp_name, departments.dept_name from employees inner join departments on employees.dept_id = departments.dept_id;

Lab 2: Use a LEFT JOIN to show all departments, even those without employees.

select departments.dept_name, employees.emp_name from departments left join employees on departments.dept_id = employees.dept_id;

12. SQL Group By

Theory Questions:

1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?

the group by clause arranges raw data into summary rows based on shared values in specific columns. when used with aggregate functions like count, sum, or avg, it performs calculations within each group rather than across the entire table. this allows you to collapse multiple rows into a single representative row for each distinct value.

2. Explain the difference between GROUP BY and ORDER BY.

group by is used to group rows that have the same values into summary rows, typically for use with aggregate functions to perform calculations on each group. order by is used to sort the result set in either ascending or descending order based on one or more columns without changing the data itself.

LAB EXERCISES:

Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.

```
select dept_id, count(*) from employees group by dept_id;
```

Lab 2: Use the AVG aggregate function to find the average salary of employees in each department.

```
select dept_id, avg(salary) from employees group by dept_id;
```

13. SQL Stored Procedure

Theory Questions:

1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?

a stored procedure is a prepared sql code block that you save so the code can be reused over and over again. unlike a standard sql query that is sent to the database and compiled every time it runs, a stored procedure is stored in the database server, compiled once, and can accept parameters to perform complex logic or multiple operations in a single call.

2. Explain the advantages of using stored procedures.

the advantages of using stored procedures include improved performance because the code is pre-compiled and stored on the server, which reduces execution time. they also enhance security by allowing users to execute the procedure without giving them direct access to the

underlying tables. additionally, stored procedures reduce network traffic because only the call to the procedure is sent over the network rather than multiple long queries. they also promote code reusability and easier maintenance since logic is centralized in one place.

LAB EXERCISES:

Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

create procedure get_employees_by_dept (in d_id int) begin select * from employees where dept_id = d_id; end;

Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.

create procedure get_course_details (in c_id int) begin select * from courses where course_id = c_id; end;

14. SQL View

Theory Questions:

1. What is a view in SQL, and how is it different from a table?

a view is a virtual table based on the result-set of an sql statement. unlike a table, which contains and stores its own data on the disk, a view does not store data itself but acts as a searchable object that dynamically pulls data from the underlying base tables whenever it is queried.

2. Explain the advantages of using views in SQL databases.

the advantages of using views include simplified data access by hiding complex joins and logic behind a single virtual table name. they enhance security by restricting user access to specific columns or rows instead of the entire base table. views also provide data independence, allowing the underlying table structure to change without affecting the external applications that rely on the view. finally, they ensure consistency by centralizing complex business calculations in one place.

LAB EXERCISES:

Lab 1: Create a view to show all employees along with their department names.

create view employee_dept_view as select employees.emp_name, departments.dept_name from employees inner join departments on employees.dept_id = departments.dept_id;

Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

create or replace view employee_dept_view as select employees.emp_name, departments.dept_name from employees inner join departments on employees.dept_id = departments.dept_id where employees.salary >= 50000;

15. SQL Triggers

Theory Questions:

1. What is a trigger in SQL? Describe its types and when they are used.

a trigger is a special type of stored procedure that automatically executes or fires when a specific event occurs in the database. triggers are used to maintain data integrity, audit changes, and enforce business rules that cannot be handled by constraints alone.

before triggers: these fire before the triggering event (insert, update, or delete) is executed on the table. they are often used to validate or modify data before it is saved.after triggers: these fire after the triggering event has been completed. they are commonly used for auditing purposes or to update data in other related tables.

2. Explain the difference between INSERT, UPDATE, and DELETE triggers.

insert triggers fire automatically when a new record is added to a table. they are commonly used to validate data or log the creation of new entries.

update triggers fire when existing data within a table is modified. these triggers are useful for tracking changes by comparing the old values with the new values.

delete triggers fire when a row is removed from a table. they are often used to archive data or prevent the deletion of critical records.

LAB EXERCISES:

Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

create trigger log_new_employee after insert on employees for each row begin insert into audit_log (action, emp_id, change_date) values ('insert', new.emp_id, now()); end;

Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

create trigger update_employee_timestamp before update on employees for each row begin set new.last_modified = now(); end;

16. Introduction to PL/SQL

Theory Questions:

1. What is PL/SQL, and how does it extend SQL's capabilities?

pl/sql stands for procedural language extensions to sql. it is oracle corporation's proprietary extension to sql that allows developers to combine the data manipulation power of sql with the processing power of procedural languages.

2. List and explain the benefits of using PL/SQL.

pl/sql is a procedural language that extends sql by adding programming features like variables, loops, and conditional logic. this allows you to handle complex data processing within the database itself rather than in the application code. by grouping multiple sql statements into a single block, it reduces network traffic and improves execution speed since the entire block is sent to the server at once.

LAB EXERCISES:

Lab 1: Write a PL/SQL block to print the total number of employees from the employees table.

declare v_count number; begin select count(*) into v_count from employees; dbms_output.put_line(v_count); end;

Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.

declare v_total number; begin select sum(order_amount) into v_total from orders; dbms_output.put_line(v_total); end;

17. PL/SQL Control Structures

Theory Questions:

1. What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

control structures in pl/sql manage the flow of code execution based on conditions. if-then statements execute a specific block of code only when a condition is true. loop structures allow you to repeat a sequence of statements multiple times efficiently. these tools enable the database to handle logic and decision-making during execution.

2. How do control structures in PL/SQL help in writing complex queries?

control structures allow you to implement logical decision-making directly inside the database. they use conditional branches to execute different code paths based on the data retrieved.

loops enable repetitive processing of data sets without writing redundant manual sql statements. this reduces complex nested logic into manageable blocks that improve overall code efficiency.

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

declare v_dept varchar2(50); begin select dept_name into v_dept from employees where emp_id = 101; if v_dept = 'sales' then dbms_output.put_line('this is the sales department'); end if; end;

Lab 2: Use a FOR LOOP to iterate through employee records and display their names.

begin for r_emp in (select emp_name from employees) loop
dbms_output.put_line(r_emp.emp_name); end loop;

18. SQL Cursors

Theory Questions:

1. What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

a cursor is a pointer to a memory area where the database stores results of a query. implicit cursors are created by the system for single dml statements like insert or update. explicit cursors are defined by the user to process multiple rows one by one in a loop. the main difference is that explicit cursors offer more control over data fetching and processing.

2. When would you use an explicit cursor over an implicit one?

you use an explicit cursor when you need to process a query that returns multiple rows. it provides better control by allowing you to fetch records one at a time within a loop. unlike implicit cursors, explicit cursors let you manage memory better for large result sets. they are essential when you need to perform complex logic on each row individually during a fetch.

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

declare cursor c_emp is select emp_name, salary from employees; v_name employees.emp_name%type; v_sal employees.salary%type; begin open c_emp; loop fetch c_emp into v_name, v_sal; exit when c_emp%notfound; dbms_output.put_line(v_name || ' ' || v_sal); end loop; close c_emp; end;

Lab 2: Create a cursor to retrieve all courses and display them one by one.

declare cursor c_courses is select course_name from courses; v_name courses.course_name%type; begin open c_courses; loop fetch c_courses into v_name; exit when c_courses%notfound; dbms_output.put_line(v_name); end loop; close c_courses; end;

19. Rollback and Commit Savepoint

Theory Questions:

1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?

savepoint creates a marker within a transaction to allow partial rollbacks of work. rollback to savepoint undoes only the changes made after that specific marker was set. commit deletes all savepoints and permanently saves every change made during the transaction. a full rollback ignores savepoints and reverses the entire transaction back to the beginning.

2. When is it useful to use savepoints in a database transaction?

savepoints are useful when you need to break a long transaction into smaller logical units. they allow you to undo specific errors or mistakes without losing all the work previously done. this is ideal for complex batch processing where you only want to retry a failed segment. they provide flexibility in multi-step operations to maintain data integrity during partial failures.

LAB EXERCISES:

Lab 1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

begin insert into employees (emp_id, emp_name) values (101, 'john'); savepoint sp1; insert into employees (emp_id, emp_name) values (102, 'jane'); rollback to sp1; commit; end;

Lab 2: Commit part of a transaction after using a savepoint and then rollback the remaining Changes

begin insert into departments (dept_id, dept_name) values (10, 'hr'); savepoint sp_dept; insert into employees (emp_id, emp_name, dept_id) values (500, 'alice', 10); rollback to sp_dept; commit; end;

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.

create database library_db;

create table books (book_id number primary key, title varchar2(100), author varchar2(100), publisher varchar2(100), year_of_publication number, price number);

insert into books values (1, 'the alchemist', 'paulo coelho', 'harpercollins', 1988, 20);

insert into books values (2, '1984', 'george orwell', 'secker', 1949, 15);

insert into books values (3, 'the hobbit', 'j.r.r. tolkien', 'allen & unwin', 1937, 25);

insert into books values (4, 'macbeth', 'william shakespeare', 'blount', 1623, 10);

insert into books values (5, 'it', 'stephen king', 'viking', 1986, 30);

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.

create table members (member_id number primary key, member_name varchar2(100), date_of_membership date, email varchar2(100));

insert into members values (1, 'john doe', '01-jan-2021', 'john@example.com');

insert into members values (2, 'jane smith', '15-mar-2022', 'jane@example.com');

insert into members values (3, 'sam brown', '10-dec-2019', 'sam@example.com');

insert into members values (4, 'lisa ray', '20-may-2023', 'lisa@example.com');

insert into members values (5, 'paul mark', '05-aug-2018', 'paul@example.com');

2. sql syntax

lab 3: retrieve all members who joined the library before 2022. use appropriate sql syntax with where and order by.

select * from members where date_of_membership < '01-jan-2022' order by date_of_membership;

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.

select title from books where author = 'george orwell' order by year_of_publication desc;

3. sql constraints

lab 3: add a check constraint to ensure that the price of books in the books table is greater than 0.

alter table books add constraint chk_price 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.

alter table members add constraint unq_email 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.

create table authors (author_id number primary key, first_name varchar2(50), last_name varchar2(50), country varchar2(50));

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.

create table publishers (publisher_id number primary key, publisher_name varchar2(100), contact_number varchar2(20) unique, address varchar2(200));

5. alter command

lab 3: add a new column genre to the books table. update the genre for all existing records.

alter table books add genre varchar2(30);

update books set genre = 'general';

lab 4: modify the members table to increase the length of the email column to 100 characters.

alter table members modify email varchar2(100);

6. drop command

lab 3: drop the publishers table from the database after verifying its structure.

describe publishers;

drop table publishers;

lab 4: create a backup of the members table and then drop the original members table.

create table members_backup as select * from members;

drop table members;

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.

insert into authors values (10, 'mark', 'twain', 'usa');

insert into authors values (11, 'charles', 'dickens', 'uk');

insert into authors values (12, 'leo', 'tolstoy', 'russia');

update authors set last_name = 'clemens' where author_id = 10;

lab 5: delete a book from the books table where the price is higher than $100.

delete from books where price > 100;

8. update command

lab 3: update the year_of_publication of a book with a specific book_id.

update books set year_of_publication = 2020 where book_id = 1;

lab 4: increase the price of all books published before 2015 by 10%.

update books set price = price * 1.10 where year_of_publication < 2015;

9. delete command

lab 3: remove all members who joined before 2020 from the members table.

delete from members where date_of_membership < '01-jan-2020';

lab 4: delete all books that have a null value in the author column.

delete from books where author is null;

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 asc fetch first 3 rows only;

11. data control language (dcl)

lab 3: grant select permission to a user named librarian on the books table.

grant select on books to librarian;

lab 4: grant insert and update permissions to the user admin on the members table.

grant insert, update on members to admin;

12. revoke command

lab 3: revoke the insert privilege from the user librarian on the books table.

revoke insert on books from librarian;

lab 4: revoke all permissions from user admin on the members table.

revoke all on members from admin;

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.

insert into books values (6, 'book a', 'author a', 'pub a', 2021, 10);

insert into books values (7, 'book b', 'author b', 'pub b', 2022, 12);

commit;

insert into books values (8, 'book c', 'author c', 'pub c', 2023, 14);

rollback;

lab 4: set a savepoint before making updates to the members table, perform some updates, and then roll back to the savepoint.

savepoint before_update;

update members set member_name = 'test';

rollback to savepoint before_update;

14. sql joins

lab 3: perform an inner join between books and authors tables to display the title of books and their respective authors' names.

select books.title, authors.first_name, authors.last_name from books inner join authors on books.author = authors.last_name;

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.

select * from books full outer join authors on books.author = authors.last_name;

15. sql group by

lab 3: group books by genre and display the total number of books in each genre.

select genre, count(*) 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 extract(year from date_of_membership), count(*) from members group by extract(year from date_of_membership);

16. sql stored procedure

lab 3: write a stored procedure to retrieve all books by a particular author.

create procedure get_author_books (p_author in varchar2) as begin for r in (select * from books where author = p_author) loop dbms_output.put_line(r.title); end loop; end;

lab 4: write a stored procedure that takes book_id as an argument and returns the price of the book.

create procedure get_book_price (p_id in number, p_price out number) as begin select price into p_price from books where book_id = p_id; end;

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_info as select title, author, price from books;

lab 4: create a view to display members who joined before 2020.

create view old_members as select * from members where date_of_membership < '01-jan-2020';

18. sql trigger

lab 3: create a trigger to automatically update the last_modified timestamp of the books table whenever a record is updated.

create trigger trg_books_update before update on books for each row begin :new.last_modified := sysdate; end;

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 trigger trg_books_delete after delete on books for each row begin insert into log_changes values (:old.book_id, 'deleted', sysdate); end;

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.

begin

insert into books values (20, 'plsql guide', 'it expert', 'tech pub', 2025, 45, 'edu');

dbms_output.put_line('book inserted successfully');

end;

lab 4: write a pl/sql block to display the total number of books in the books table.

declare

v_count number;

begin

select count(*) into v_count from books;

dbms_output.put_line(v_count);

end;

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.

declare

v_id number := 10;

v_price number := 99.99;

begin

dbms_output.put_line(v_id || ' ' || v_price);

end;

lab 4: write a pl/sql block using constants and perform arithmetic operations on book prices.

```
declare
c_tax constant number := 0.05;
v_price number := 100;
v_final number;
begin
v_final := v_price + (v_price * c_tax);
dbms_output.put_line(v_final);
end;
```

21. pl/sql control structures

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.

```
declare
v_price number := 120;
begin
if v_price > 100 then
dbms_output.put_line('expensive');
else
dbms_output.put_line('affordable');
end if;
end;
```

lab 4: use a for loop in pl/sql to display the details of all books one by one.

```
begin
for r in (select * from books) loop
dbms_output.put_line(r.title || ' by ' || r.author);
end loop;
```

end;

22. sql cursors

lab 3: write a pl/sql block using an explicit cursor to fetch and display all records from the members table.

```
declare
cursor c_mem is select * from members;
r_mem members%rowtype;
begin
open c_mem;
loop
fetch c_mem into r_mem;
exit when c_mem%notfound;
dbms_output.put_line(r_mem.member_name);
end loop;
close c_mem;
end;
```

lab 4: create a cursor to retrieve books by a particular author and display their titles.

```
declare
cursor c_books is select title from books where author = 'paulo coelho';
v_title books.title%type;
begin
open c_books;
loop
fetch c_books into v_title;
exit when c_books%notfound;
dbms_output.put_line(v_title);
```

```
end loop;

close c_books;

end;
```

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.

```
begin

insert into members values (30, 'new user', sysdate, 'new@web.com');

savepoint s1;

update members set member_name = 'changed' where member_id = 30;

rollback to s1;

commit;

end;
```

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.

```
begin

insert into books values (40, 'a', 'x', 'y', 2020, 10, 'z');

insert into books values (41, 'b', 'x', 'y', 2021, 11, 'z');

commit;

savepoint s2;

delete from books where author = 'x';

rollback to s2;

end;
```

would you like me to explain the logic behind any of these triggers or cursors?