Experiment 2: Implement simple queries of DDL and DML.

Theory:
DDL stands for Data Definition Language. It is used to specify a database's structure, which includes its tables, views, indexes, and constraints.
DDL statements only modify the database's schema, they have no direct effect on the data within the database.
For example, creating the table, altering the table or deleting the table schema.

DML stands for Data Manipulation Language. It is used to manipulate data in a database. It inserts, updates, and deletes the data from a database table.
Using DML commands, only the row data from the table is accessed. The schema of the table cannot be modified using these.
Examples of DML statements include INSERT, UPDATE, and DELETE

Learning Outcome:
we learnt the difference between DDL and DML commands and applied various DDL and DML operations like creating the table schema, inserting the data, altering the table data and table schema

Experiment 3: Implement basic queries to Create, Insert, Update, Delete and Select Statements for two different scenarios (For instance: College, Bank)

Theory:
Create: Used to create database objects such as tables, views, or indexes.
Syntax: CREATE TABLE table_name (column1 datatype, column2 datatype, ...);

Insert: Adds new records into a table.
Syntax: INSERT INTO table_name VALUES (value1, value2, ...);

Update: Modifies existing records in a table.
Syntax: UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition;

Delete: Removes records from a table.
Syntax: DELETE FROM table_name WHERE condition;

Select: Retrieves data from one or more tables based on specified criteria.
Syntax: SELECT column1, column2, ... FROM table_name WHERE condition;

Learning Outcome:
We learnt about using different operations like creating the table, inserting data into the table, updating the table data, deleting the data and displaying all or selective records from the database created for college and bank.

Experiment 4: Implement queries including various functions- mathematical, string, date etc.

Theory:
SQL mathematical functions, such as ROUND, CEIL, FLOOR, ABS, and POWER, enable numeric operations and rounding.

String functions like CONCAT, UPPER, LOWER, SUBSTRING, and LENGTH facilitate efficient manipulation and formatting of character data.

Date functions like SYSDATE, TO_DATE, MONTHS_BETWEEN, ADD_MONTHS, and LAST_DAY are crucial for handling date and time values within the database.

These functions enhance the versatility of SQL queries, allowing for computations, text manipulations, and date-related operations, contributing to the power and flexibility of SQL as a database language

Learning Outcome:
We learnt about using different mathematical functions, string functions as well as date functions and we learnt how to perform and apply these functions to our data.

Experiment 5: Implement queries including sorting, grouping and subqueries – LIKE, ANY, ALL, EXISTS, NOT EXISTS

Theory:
LIKE: Used to search for a specified pattern in a column.
Syntax: SELECT column_name FROM table_name WHERE column_name LIKE pattern;

ANY: Compares a value to a set of values returned by a subquery, and returns true if the comparison is true for any value in the set
Syntax: SELECT column_name FROM table_name WHERE column_name operator ANY (SELECT column_name FROM table_name WHERE condition);

ALL: Compares a value to a set of values returned by a subquery, and returns true if the comparison is true for all values in the set.
Syntax: SELECT column_name FROM table_name WHERE column_name operator ALL (SELECT column_name FROM table_name WHERE condition);

EXISTS: Tests for the existence of any rows in a subquery result set.
Syntax: SELECT column_name FROM table_name WHERE EXISTS (SELECT column_name FROM table_name WHERE condition);

NOT EXISTS: Tests for the non-existence of any rows in a subquery result set.
Syntax: SELECT column_name FROM table_name WHERE NOT EXISTS (SELECT column_name FROM table_name WHERE condition);

Learning Outcome:
We learnt about using different operations like sorting the table, grouping data, and applying subqueries on same or different tables.

Experiment 6: Implement queries using various set operations (UNION, INTERACTION, EXCEPT etc.)

Theory:
Set operations in SQL allow for combining or comparing the results of two or more queries.

UNION: Combines the results of two or more SELECT statements, removing duplicate rows.
Syntax: SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;

INTERSECT: Returns only the rows that appear in both result sets of two SELECT statements.
Syntax: SELECT column_name(s) FROM table1
INTERSECT
SELECT column_name(s) FROM table2;

EXCEPT: Returns only the distinct rows from the first SELECT statement that are not present in the result set of the second SELECT statement.
Syntax: SELECT column_name(s) FROM table1
EXCEPT
SELECT column_name(s) FROM table2;

Learning Outcome:
We learnt about different set operations like Union, Interaction and Except and how to apply them on different tables.

Experiment 7: Implement various join operations (INNER, OUTER).

Theory:
Join operations in SQL combine rows from two or more tables based on a related column between them

INNER: Returns rows from both tables where there is a match based on the specified join condition.
Syntax: SELECT table1.column1, table2.column2
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;

OUTER: Returns all rows from one or both tables, along with matching rows from the other table based on the specified join condition. It includes unmatched rows as well.
Syntax: SELECT table1.column1, table2.column2
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name = table2.column_name;

Learning Outcome:
We learnt about using different Join operations like inner join and outer join and learnt how to retrieve data that is linked in two or more different tables.

Experiment 8: Write a PL/SQL program using FOR loop to insert ten rows into a database table.

Theory:
PL/SQL (Procedural Language/Structured Query Language) is Oracle's extension to SQL, providing procedural constructs like loops, conditional statements, and exception handling for database programming.

It enables the creation of stored procedures, functions, triggers, and packages within the Oracle database, enhancing its functionality and allowing for complex data manipulation and business logic implementation.

In PL/SQL, a FOR loop is used to iterate over a range of values or a collection. It simplifies repetitive tasks by executing a block of statements for each iteration.

Syntax:

```
FOR loop_counter IN [REVERSE] lower_bound..upper_bound LOOP
  -- Statements
END LOOP;
```

- The loop_counter variable iterates from the lower_bound to the upper_bound value.
- The loop can be reversed by adding the REVERSE keyword before the lower_bound.

Learning Outcome:
We learnt how PL/SQL works and is implemented in SQL and also got to know about using FOR loop using PL/SQL statements.

Experiment 9: Given the table EMPLOYEE (Emp No, Name, Salary, Designation, DeptID), write a cursor to select the five highest-paid employees from the table.

Theory:
Cursors in SQL are database objects used to traverse through the result set of a query one row at a time. They provide a way to retrieve and process individual rows sequentially, making them useful for operations such as data manipulation and reporting.

Cursors consist of three main steps: declaration, opening, and fetching rows. They can be used to iterate over the result set of a SELECT query and perform operations on each row.

Syntax for Cursor Declaration:

```
DECLARE cursor_name CURSOR FOR
  SELECT column1, column2, ...
  FROM table_name
  WHERE condition;
```

Learning Outcome:
We learnt about the concept of cursors in PL/SQL and how to access information using cursors from tables.

Experiment 1: Write the steps to install and implement NOSQL databases MongoDB.

Learning Outcome:
We learnt how to download and install NOSQL database such as MongoDB and also practiced how to run MongoDB server.

Experiment 2: Study and implement basic commands of MongoDB.

Theory:
Database and Collection Creation: Use use command to switch to or create a database, and db.createCollection() to create a collection.

Document Insertion: Use insertOne() or insertMany() to insert documents into a collection.

Querying Documents: Use find() to retrieve documents from a collection based on specified criteria.

Updating Documents: Use updateOne() or updateMany() to modify existing documents in a collection.

Deleting Documents: Use deleteOne() or deleteMany() to remove documents from a collection.

Collection and Database Dropping: Use drop() to remove a collection and db.dropDatabase() to drop a database.

These commands are fundamental for interacting with MongoDB databases, enabling developers to perform CRUD operations and manage data effectively.

Learning Outcome:
We learnt how to use MongoDB and implement some of its basic queries like insert delete display and others for manipulation of data.

Experiment 3: Implement any one real-time project using MySQL/MongoDB such as Library Database Management System etc..

Learning Outcome:
We learnt how to use MySQL database for real world programs like Library management system, where we used different schemas with relations for storing data and understood the concept of triggers by using it for auto updating tables for quantity of books and fines.