**THEORY OF EACH DBMS PRACTICAL**

**Exp 1: -**

1) MySQL:

MySQL is an open-source, relational database management system (RDBMS) known for its speed, reliability, and ease of use. It’s widely used in web applications, particularly in LAMP (Linux, Apache, MySQL, PHP) stack environments, and offers strong community support.

Pros:

* Open-source and free to use
* Excellent for web applications
* High performance for read-heavy operations
* Large community and robust documentation

Cons:

* Limited scalability in comparison to other DBMS
* Lack of advanced features like full ACID compliance in certain configurations
* Less support for complex queries

2) Oracle:

Oracle Database is a comprehensive, enterprise-grade RDBMS known for its scalability, high performance, and extensive features. It's preferred for large applications with high transaction volumes and complex data management needs, especially in enterprise environments.

Pros:

* Highly scalable and secure
* Advanced features for data integrity and recovery
* Strong support for large enterprise systems
* Excellent for complex transactions

Cons:

* Expensive licensing
* Requires more resources (hardware and management)
* Steeper learning curve for beginners

3) PostgreSQL:

PostgreSQL is an advanced open-source relational database system known for its robustness, extensibility, and standards compliance. It supports both SQL (relational) and JSON (non-relational) data types, making it ideal for complex applications and hybrid workloads.

Pros:

* Open-source with a strong community
* Advanced features, including support for JSON, full-text search, and geospatial data
* High ACID compliance and data integrity
* Strong performance for complex queries and large datasets

Cons:

* Can be slower than MySQL for simple queries
* More complex to manage and tune
* Less widespread support in certain enterprise environments

3) SQL Server:

SQL Server is a relational database management system from Microsoft that integrates well with other Microsoft products. It offers a user-friendly interface, extensive reporting, and data analysis tools, making it a popular choice for businesses running on Microsoft infrastructure.

Pros:

* Strong integration with Microsoft technologies (e.g., Azure, .NET)
* Excellent business intelligence and reporting tools
* Good security features and scalability
* Comprehensive support and documentation

Cons:

* Licensing can be costly
* Runs primarily on Windows, although Linux support has improved
* Not as widely used in open-source environments

5) SQL\*Plus:

SQL\*Plus is an Oracle command-line interface tool used for managing and interacting with Oracle databases. It allows users to execute SQL queries, perform database administration tasks, and generate reports, providing a direct interface to Oracle Database.

Pros:

* Native tool for interacting with Oracle databases
* Lightweight and fast for executing SQL queries
* Script execution support for automating database tasks
* Integrates well with Oracle’s ecosystem

Cons:

* Command-line interface, which can be less user-friendly compared to graphical tools
* Limited functionality for complex database administration tasks compared to other Oracle tools
* Primarily designed for Oracle databases, not as versatile for other DBMS

**Exp 2: -**

**DDL (Data Definition Language)** consists of SQL commands used to define and modify the structure of database objects, such as tables, indexes, and views. Key DDL commands include CREATE, ALTER, and DROP, which allow users to establish, modify, or remove database schemas and structures. DDL is primarily concerned with the organization and architecture of the database, and any changes made through DDL commands are permanent, typically requiring database reorganization without affecting the actual data contained within the database.

**DML (Data Manipulation Language)** refers to a set of SQL commands used to manage and manipulate the data stored within database tables. DML commands such as SELECT, INSERT, UPDATE, and DELETE allow users to retrieve, insert, modify, or remove data. Unlike DDL, DML is focused on the content of the database rather than its structure. Changes made via DML affect the data itself and can be rolled back or committed, offering flexibility for data manipulation in transactional systems.

**Exp 3: -**

Primary Key in SQL Plus:  
In SQL, a primary key is a constraint used to uniquely identify each record in a table. It ensures that no two rows have the same value for the primary key column(s), maintaining data integrity. A primary key cannot contain NULL values and must be unique. Typically, primary keys are created using one or more columns, known as composite keys, to guarantee the uniqueness of each record. In SQL Plus, a primary key is defined when creating or altering a table using the PRIMARY KEY constraint. This helps maintain organized, accurate, and accessible data.

Foreign Key in SQL Plus:  
A foreign key in SQL is a column (or a set of columns) that creates a relationship between two tables. It references the primary key in another table, ensuring referential integrity. The foreign key constraint prevents actions that would lead to orphaned records, such as inserting a record with a value that does not exist in the referenced table. In SQL Plus, the foreign key is defined with the FOREIGN KEY constraint, specifying the column(s) that link to the primary key in the referenced table. This helps ensure data consistency and maintains logical relationships between tables.

**Exp 4: -**

* 1. Mathematical Functions:
     + SQRT() - Calculates the square root of the numeric expression argument. The numeric expression must evaluate to a nonnegative number.
     + SIN() / COS() / TAN() - Calculates the sine/cosine/tan of a numeric expression.
     + ABS() - Calculates the absolute value of a numeric expression.
     + CEIL() - Rounds a non-integer numeric expression to the next highest integer. If the numeric expression evaluates to an integer, the CEIL function returns that integer.
     + FLOOR() - Rounds a non-integer numeric expression to the next lowest integer. If the numeric expression evaluates to an integer, the FLOOR function returns that integer.
     + ROUND() - Rounds a numeric expression to N digits of precision.
  2. String Functions:
     + ASCII() - Converts a single character string to its corresponding ASCII code, between 0 and 255. If the character expression evaluates to multiple characters, the ASCII code corresponding to the first character in the expression is returned.
     + CHR() - Converts a numeric value between 0 and 255 to the character value corresponding to the ASCII code.
     + LENGTH() - Returns the length, in number of characters, of a specified string. The length is returned does not exclude any blank spaces.
     + CONCAT() - Concatenates two character strings.
     + LOWER() / UPPER() - Converts a character string to lowercase/uppercase.
     + SUBSTR() - Creates a new string starting from the given number of characters, of a fixed length from the original string.
  3. Date Functions:
     + CURRENT\_DATE() – It returns the current date in the session time zone, in a value in the Gregorian calendar of data type DATE.
     + CURRENT\_TIMESTAMP() – It returns the current date and time in the session time zone, in a value of data type TIMESTAMP WITH TIME ZONE.
     + MONTHS\_BETWEEN() – It returns number of months between dates *date1* and *date2*. The month and the last day of the month are defined by the parameter NLS\_CALENDAR. If *date1* is later than *date2*, then the result is positive. If *date1* is earlier than *date2*, then the result is negative.
     + SYSDATE() – It returns the current date and time of the system irrespective of the session time. The session time can be altered by sysdate() gives time from the system/PC.
     + LAST\_DAY() – It returns the date of the last day of the month according to the system date.
     + NEXT\_DAY – It returns the date of the first weekday named in the quotes that is later than the date passed.
  4. Aggregate Functions:
     + SUM() - Calculates the sum obtained by adding up all values satisfying the numeric expression argument.
     + AVG() - Calculates the average (mean) of a numeric set of values.
     + MIN() - Calculates the minimum value (lowest numeric value) of the rows satisfying the numeric expression argument.
     + MAX() - Calculates the maximum value (highest numeric value) of the rows satisfying the numeric expression argument.
     + COUNT(\*) - Determines the number of items with a non-null value.

**Exp 5: -**

The LIKE keyword/operator is used in a WHERE clause to search for a specified pattern in a column. It is mostly used with columns of datatype varchar to compare and find alike strings. It uses:

* The percent sign % represents zero, one, or multiple characters
* The underscore sign \_ represents one, single character

The ANY and ALL keywords are also very useful and allow you to perform a comparison between a single column value and a range of other values: -

The **ANY** operator is used in a WHERE or HAVING clause to compare a value with any of the values returned from a subquery. The **ANY** operator returns true if the comparison is true for any of the values in the subquery i.e. the condition will be true if the operation is true for ANY of the values in the range.

The **ALL** operator is used in a WHERE or HAVING clause to compare a value with all of the values returned from a subquery. The **ALL** operator returns true if the comparison is true for ALL of the values in the subquery, i.e. the condition will be true only if the operation is true for all values in the range.

The EXISTS and NOT EXISTS keywords return Boolean values.

The EXISTS command tests for the existence of any record in a subquery, and returns true if the subquery returns one or more records or false if no row is selected. The database engine does not have to run the subquery entirely. If a single record is matched, the EXISTS operator returns true, and the associated other query row is selected.

The NOT EXISTS condition consists of two logical operators: EXISTS, which was described above, and NOT, which is used to negate a Boolean input. Unlike EXISTS, NOT EXISTS returns TRUE if the result of the subquery does not contain any rows and return FALSE, in case a single record in a table matches the subquery stopping the queries execution. Hence, NOT EXISTS allows locating records that don’t match the subquery.

**Exp 6: -**

Set operations in SQL are techniques for combining or comparing the results of two or more SELECT statements. They act like mathematical set operations, letting us find the union, intersection, or difference between the rows returned by our queries. This makes them indispensable when analysing data from multiple sources or perspectives. All set operators have equal precedence. If a SQL statement contains multiple set operators, then Oracle Database evaluates them from the left to right unless parentheses explicitly specify another order.

Here's a quick overview of the core set operations:

* **UNION**: The UNION operator combines the results of two or more SELECT queries into a single result set, removing duplicate rows by default.
* **UNION ALL:** If we want to include NULL values in the result set and prevent their removal by the UNION operator, we can use the UNION ALL operator instead. This operator combines the results of multiple SELECT queries, including all rows from each result set, regardless of whether they are duplicates or contain NULL values.
* **INTERSECT:** The INTERSECT operator returns only the rows that appear in both result sets. Think of it as finding the people who belong to both groups.
* **EXCEPT:** The EXCEPT operator retrieves the rows present in the first result set but not in the second.

**Exp 7: -**

SQL JOIN clause is used to **query**and **access data** from multiple tables by establishing**logical relationships** between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

The **INNER JOIN** keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the **result-set** by combining all rows from both the tables where the **condition satisfies**i.e value of the common field will be the same.

**LEFT JOIN** returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is **no matching row**on the right side, the result-set will contain **null**. LEFT JOIN is also known as **LEFT OUTER JOIN**.

**RIGHT JOIN** returns all the rows of the table on the **right side of the join** and matching rows for the table on the left side of the join. It is very similar to **LEFT JOIN f**or the rows for which there is no matching row on the left side, the result-set will contain **null**. **RIGHT JOIN** is also known as **RIGHT OUTER JOIN**.

**FULL JOIN** creates the result-set by combining results of both **LEFT JOIN** and **RIGHT JOIN**. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

**Exp 8: -**

PL/SQL (Procedural Language/Structured Query Language) is a block-structured language developed by Oracle that allows developers to combine the power of SQL with procedural programming constructs. The PL/SQL language enables efficient data manipulation and control-flow logic. Some features of PL/SQL are: -

1. PL/SQL is basically a procedural language, which provides the functionality of decision-making, iteration, and many more features of procedural programming languages.
2. PL/SQL can execute a number of queries in one block using single command.
3. One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
4. PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
5. Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
6. PL/SQL Offers extensive error checking.

**SET SERVEROUTPUT ON**: It is used to display the buffer used by the dbms\_output.

**var1 INTEGER :** It is the declaration of variable, named ***var1*** which is of integer type. There are many other data types that can be used like float, int, real, smallint, long etc. It also supports variables used in SQL as well like NUMBER(prec, scale), varchar, varchar2 etc.

**PL/SQL procedure successfully completed.:** It is displayed when the code is compiled and executed successfully.

**Slash (/) after END;:** The slash (/) tells the SQL\*Plus to execute the block.

**Assignment operator (:=)** : It is used to assign a value to a variable.

**Exp 9: -**

In PL/SQL, a **cursor** is a pointer that allows row-by-row processing of query results. While SQL operates on sets of data, PL/SQL cursors enable procedural control by fetching one row at a time. This is useful in scenarios where individual row manipulation or conditional logic is required.

In this experiment, an **explicit cursor** is declared to retrieve the top five highest-paid employees from the employee table. The cursor query uses ORDER BY salary DESC along with FETCH FIRST 5 ROWS ONLY to limit the result to the highest earners. The cursor is then opened, and a LOOP is used to fetch each row into declared variables. The loop continues until all five rows are processed, and data is displayed using the DBMS\_OUTPUT.PUT\_LINE procedure.

This approach demonstrates the use of cursors for controlled, step-by-step row handling, making it ideal for reporting and business logic that involves ranked or limited data processing.

**Exp 10: -**

JDBC (Java Database Connectivity) is an API that enables Java applications to interact with databases. It provides a standard interface for connecting to a database, sending SQL queries, and retrieving results. SQL*Plus, on the other hand, is an Oracle command-line tool used for executing SQL and PL/SQL commands directly on an Oracle database. Connecting JDBC to SQL*Plus essentially means using JDBC in a Java program to interact with the Oracle database in the same way one would use SQL*Plus manually. The JDBC driver (usually Oracle's thin driver) acts as a bridge between the Java application and the Oracle database. This allows developers to automate database operations, such as inserting, updating, or querying data, from within a Java program. The main advantage of using JDBC is that it enables platform-independent and scalable applications that can perform database tasks dynamically and programmatically, unlike SQL*Plus which is mostly static and manual. In the Java program I gave the password as “kunsh” which is the password for my SQL Plus client and used it to debit a part of the employees salary from the table in SQL Plus by creating a PL/SQL procedure named debit\_salary.

**Beyond Curriculum exp 1: -**

MongoDB is a popular open-source NoSQL database designed for high performance, scalability, and flexibility. Unlike traditional relational databases, MongoDB stores data in a JSON-like format called BSON, allowing for dynamic schemas and hierarchical data structures. This makes it ideal for modern applications that require real-time analytics, rapid development, and handling of large volumes of unstructured or semi-structured data.

To install MongoDB: - google MongoDB download and click on the first link. On the webpage select the version you want to install and the platform you are on (Windows/Linux/Mac etc.). Click on download and once done run the downloaded installer. Accept the user Agreement and click next. Select ‘complete’ for complete installation. Change the account name, domain and password, if needed, and click next. Check the box at the bottom to install MongoDB Compass and once that’s done click next and then click install. Finish the installation and launch MongoDB Compass. Open command prompt and run command – mongosh, to verify if we have correctly downloaded MongoDB.

**Beyond Curriculum exp 2: -**

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, allowing for scalable and high-performance data storage. First we begin by connecting to our systems runtime(localhost:27017 in our case) and then create a new database with a table/collection names students. Then we may use MongoDB shell to execute basic MongoDB commands to interact with databases and collections, including show dbs to list databases, use college to switch to our databases, and show collections to view collections. The find() command retrieves documents, while insertOne() adds new records. updateOne() modifies documents, and deleteOne() removes them. countDocuments() provides the number of documents, and sort() arranges them in a specified order. MongoDB's flexibility and ease of use make it a popular choice for modern applications.