CSL 333 DATABASE MANAGEMENT SYSTEMS LAB

SYLLABUS

1. Design a database schema for an application with ER diagram from a problem description **.
2. Creation, modification, configuration, and deletion of databases using UI and SQL Commands **.
3. Creation of database schema - DDL (create tables, set constraints, enforce relationships, create indices, delete and modify tables). Export ER diagram from the database and verify relationships** (with the ER diagram designed in step 1).
4. Database initialization - Data insert, Data import to a database (bulk import using UI and SQL Commands)**.
5. Practice SQL commands for DML (insertion, updating, altering, deletion of data, and viewing/querying records based on condition in databases)**.
6. Implementation of built-in functions in RDBMS**.
7. Implementation of various aggregate functions in SQL**.
8. Implementation of Order By, Group By & Having clause **.
9. Implementation of set operators nested queries, and join queries **.
10. Implementation of queries using temp tables.
11. Practice of SQL TCL commands like Rollback, Commit, Savepoint **.
12. Practice of SQL DCL commands for granting and revoking user privileges **.
13. Practice of SQL commands for creation of views and assertions ** .
14. Implementation of various control structures like IF-THEN, IF-THEN-ELSE, IF-THEN-ELSIF, CASE, WHILE using PL/SQL **.
15. Creation of Procedures, Triggers and Functions**.
16. Creation of Packages **.
17. Creation of Cursors **.
18. Creation of PL/SQL blocks for exception handling **.
19. Database backup and restore using commands.
20. Query analysis using Query Plan/Show Plan.

21. Familiarization of NoSQL Databases and CRUD operations**.

22. Design a database application using any front end tool for any problem selected. The application constructed should have five or more tables**.

** mandatory

LAB CYCLE

1. Design a database schema for an application with ER diagram from a problem description
2. Creation, modification, configuration, and deletion of databases
3. Creation of database schema - DDL (create tables, set constraints, enforce relationships, create indices, delete and modify tables). Practice SQL commands for DML (insertion, updating, altering, deletion of data, and viewing/querying records based on condition in databases).
4. Database initialization - Data insert, Data import to a database (bulk import using UI and SQL Commands)
5. Implementation of Order By, Group By & Having clause.
6. Implementation of various aggregate functions in SQL.
7. Implementation of set operators nested queries, and join queries .
8. Implementation of built-in functions in RDBMS.
9. Implementation of nested queries
10. Practice of SQL TCL commands like Rollback, Commit, Savepoint .
11. Practice of SQL DCL commands for granting and revoking user privileges .
12. Practice of SQL commands for creation of views and assertions .
13. Implementation of various control structures like IF-THEN, IF-THEN-ELSE, IF-THEN-ELSIF, CASE, WHILE using PL/SQL .
14. Creation of PL/SQL blocks for exception handling .
15. Creation of Procedures, Triggers and Functions.
16. Creation of Cursors .
17. Creation of Packages .
18. Familiarization of NoSQL Databases and CRUD operations.
19. Design a database application using any front end tool for any problem selected. The application constructed should have five or more tables

PROGRAM NO 1: Design a database schema for COMPANY with ER diagram

COMPANY

EMPLOYEE

ENO	ENAME	DESIGNATION	SEX	AGE	SALARY	DATEOFJOIN	PNO

PROJECT

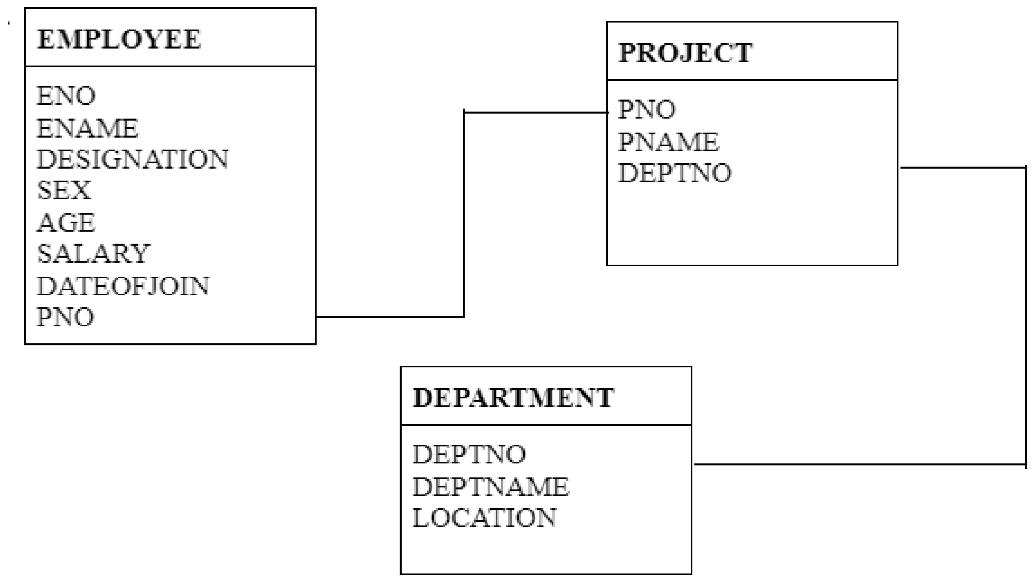
PNO	PNAME	DEPTNO

DEPARTMENT

DEPTNO	DEPTNAME	LOCATION

The design of the database is called a schema. This tells about the structural view of the database. It gives an overall description of the database. A database schema defines how the data is organized using the schema diagram. A schema diagram is a diagram which contains entities and the attributes that will define that schema. A schema diagram only shows us the database design. It does not show the actual data of the database. Schema can be a single table or it can have more than one table which is related.

Here are three tables: Employee, Department and Project. So, we can represent the schema of these three tables using the schema diagram as follows. In this schema diagram, Employee and Department and Project tables are related.

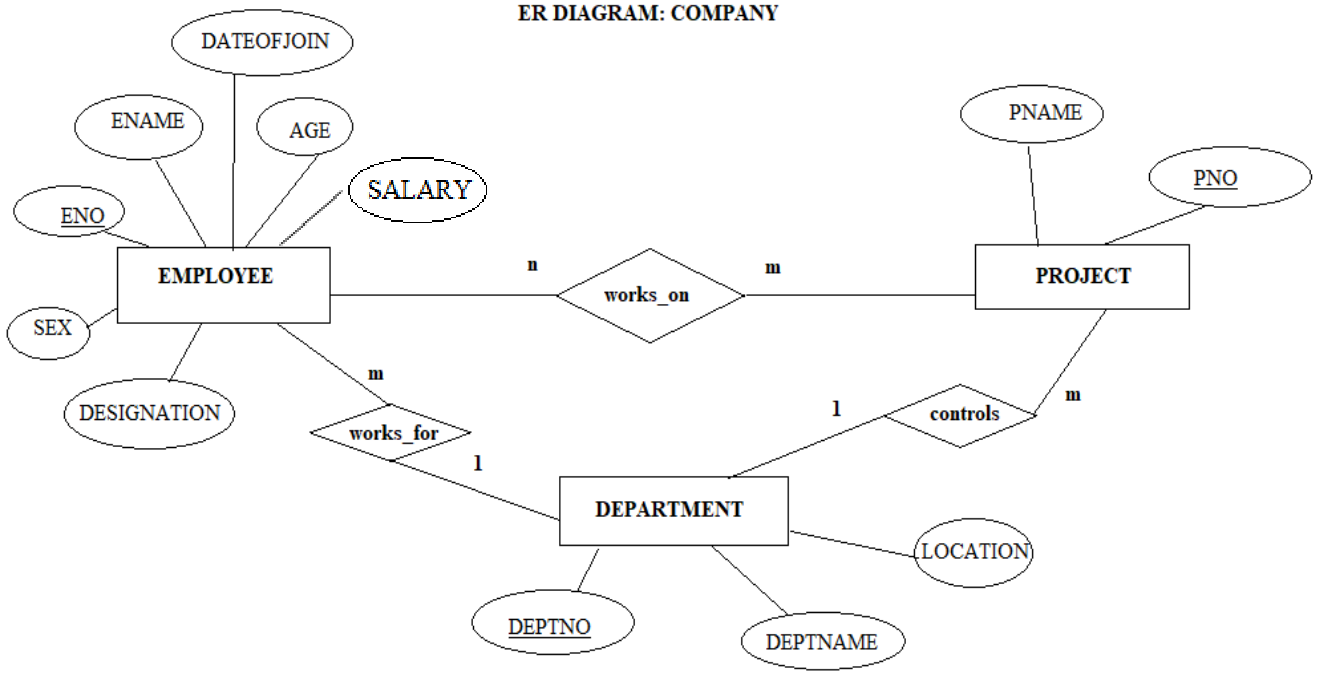


COMPANY SCHEMA

ENTITY RELATIONSHIP (ER) DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of the E-R model are: entity set and relationship set. An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in a database, so by showing relationships among tables and their attributes, ER diagram shows the complete logical structure of a database.

ER DIAGRAM: COMPANY



STRUCTURED QUERY LANGUAGE (SQL)

SQL is a Structured Query language. It is a non-procedural language to retrieve data from a database.

Different SQL COMMANDS are,

DDL (Data Definition Language) : DDL is used to define the database structure or table.

Commands such as Create, Alter, Drop are used.

DML (Data Manipulation Language): DML is used for managing data within table objects.

Commands such as Select, insert, update, delete are used.

DCL(Data Control Language) : DCL is used to give privileges to access limited data. Commands such as Grant , Revoke are used.

TCS(Transaction Control Statement) : TCS are used to apply the changes permanently saved into the database. Commands such as Commit, Rollback, Savepoint are used.

DDL (Data Definition Language) Statement and its constraint

TO CREATE TABLES:

CREATE TABLE table_name (column_name column_type);

TO DROP TABLES:

DROP TABLE table_name ;

ALTER TABLE:

ALTER TABLE table_name ADD column_name datatype;

ALTER TABLE table_name DROP COLUMN column_name;

SQL CONSTRAINTS:

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted. Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table. The following constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value.

UNIQUE - Ensures that all values in a column are different.

PRIMARY KEY - A combination of NOT NULL and UNIQUE. Uniquely identifies each row in a table.

CHECK - Ensures that all values in a column satisfies a specific condition.

```
CREATE TABLE Persons (

   ID int NOT NULL UNIQUE,

   LastName varchar(25) NOT NULL,

   FirstName varchar(25) NOT NULL,

   Age int CHECK (Age>=18),

PRIMARY KEY (ID) );
```

FOREIGN KEY - Uniquely identifies a row/record in another table

```
CREATE TABLE Orders (

   OrderID int NOT NULL,

   OrderNumber int NOT NULL,

   PersonID int,

   PRIMARY KEY (OrderID),

   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

);
```

PROGRAM NO 2: CREATION AND DELETION OF DATABASES

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

The CREATE DATABASE statement is used to create a new SQL database.

<u>Syntax</u>

CREATE DATABASE *databasename*;

The DROP DATABASE statement is used to drop an existing SQL database.

<u>Syntax</u>

DROP DATABASE *databasename*;

Creation and deletion of database Company

```
CREATE DATABASE company;

DROP DATABASE company;
```

PROGRAM NO 3: TABLE CREATION

Aim: This program is to create tables for the database company. Tables to be created are given below:

EMPLOYEE

Fieldname	Type	Constraint
ENO	Text	Primary key, Begin with ‘E’
ENAME	Text	
DESIGNATION	Text	
SEX	Text	Values: ‘M’ or ‘F’
AGE	Number	Greater than 0
SALARY	Real	
DATEOFJOIN	Date	
PNO	Text	Foreign Key references Project Table

PROJECT

Fieldname	Type	Constraint
PNO	Text	Primary key, Begin with ‘P’
PNAME	Text	
DEPTNO	Number	Foreign Key references Department Table

DEPARTMENT

Fieldname	Type	Constraint
DEPTNO	Number	Primary key
DEPTNAME	Text	
LOCATION	Text	Values: ‘CSE’, ‘IT’,’EEE’,ECE’,’ME’

Question:

1. Create the above tables

Result: Tables created successfully

Instructions:

1. Right Side of Record:Date, Program Number, Heading, Aim, Question, Result
2. Left side of Record: Diagram(if any), print out of queries and output

PROGRAM NO 4: TABLE ALTERATION & DATA INSERTION INTO TABLES

Aim: To alter the structure of the above created table and inserting values into the tables.

Questions:

1. Alter the table EMPLOYEE by adding the field CITY
2. Alter table EMPLOYEE by modifying the type of SALARY as float number of form (10,3)
3. Alter table EMPLOYEE by adding the constraint SAL for checking whether salary is greater than 0 or not
4. Alter table EMPLOYEE by dropping the constraint SAL
5. Insert 5 values each into the tables EMPLOYEE, PROJECT, DEPARTMENT
6. Display the values inserted in each table
7. Show referential integrity violation

Result: Table altered successfully and data are inserted

DML (Data Manipulation Language) and Clauses

TO INSERT DATA INTO THE TABLE:

INSERT INTO table_name (field1, field2,...fieldN) VALUES (value1, value2,...valueN);

SELECT QUERY :

SELECT field1, field2,...fieldN FROM table_name1, table_name2... [WHERE Clause]

UPDATE QUERY:

UPDATE table_name SET field1 = new-value1, field2 = new-value2 [WHERE Clause]

DELETE QUERY:

DELETE FROM table_name [WHERE Clause]

AND:

SELECT column1, column2, ... FROM table_name WHERE condition1 AND condition2 AND condition3 ...;

OR:

SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2 OR condition3 ...;

NOT :

SELECT column1, column2, ... FROM table_name WHERE NOT condition;

BETWEEN :

SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;

ALIAS:

SELECT column_name AS alias_name FROM table_name;

SELECT column_name FROM table_name AS alias_name ;

ORDER BY:

SELECT field1, field2,...fieldN table_name1, table_name2... ORDER BY field1, [field2...] [ASC] [DESC]]

LIKE :

SELECT field1, field2,...fieldN table_name1, table_name2... WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'

NULL VALUE:

SELECT column_names FROM table_name WHERE column_name IS NULL;

PROGRAM NO 5: IMPLEMENTING DML COMMANDS

Aim: To execute SQL commands that implements data manipulation in the tables

Questions:

1. List the name of all employees
2. List the name of employees who are working as ‘Analyst’ and ‘Salesman’
3. List the details of employees who has joined the company before 30th September 2016
4. List the details of employees who have ENO as E2,E4,E6,E8
5. List the details of all employees
6. List the list of employees not working under projects P2,P3,P4
7. List of employees who have joined the company between 30th june 2016 and 1st July 2017
8. List the different designation of employees
9. List the employees who are not assigned to any project
10. List the name of employees whose name either start or end with ‘S’
11. List the employees whose name have ‘i’ as the second character
12. List the employees whose name begins and end with ‘A’ and has atleast 3 characters
13. List the employees whose salary is in the range of 5000 and 8000
14. Remove the details of employees whose age > 60
15. Provide 20% increment in salary for all employees whose salary is in the range of 5000 and 8000
16. Display the details of employees whose name contain ‘e’ as the second character

Result: The queries for above questions executed successfully.

AGGREGATE FUNCTIONS:

SELECT **COUNT** (column_name) FROM table_name WHERE condition;

SELECT **AVG** (column_name) FROM table_name WHERE condition;

SELECT **SUM** (column_name) FROM table_name WHERE condition;

SELECT **MAX** (column_name) FROM table_name WHERE condition;

SELECT **MIN** (column_name) FROM table_name WHERE condition;

GROUP BY, HAVING CLAUSE :

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns. The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

SYNTAX:

SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s)

HAVING condition ;

PROGRAM NO 6: AGGREGATE FUNCTIONS

Aim: To execute commands to perform aggregate functions

Questions:

1. List the maximum salary in EMPLOYEE
2. List the minimum salary in EMPLOYEE
3. List the total salary of all the employees in the EMPLOYEE table
4. List the average salary in EMPLOYEE
5. List the total number of employees in EMPLOYEE
6. List the maximum salary paid to a ‘Salesman’
7. List the total salary paid to a ‘Manager’
8. List the age of senior most employee
9. List the age of younger most employee
10. Display the details of employee in ascending order of their name
11. Display the details of employees in descending order of salary
12. Find the number of employees in each project
13. Find total, max, min, and average salary of employee joined for the project P1 and having average salary >1000

Result: The commands for above questions executed successfully.

JOIN & SET OPERATIONS

JOIN OPERATION

SELECT column_name(s) FROM table1

INNER JOIN table2 **ON** table1.column_name = table2.column_name;

SELECT column_name(s) FROM table1

LEFT JOIN table2 **ON** table1.column_name = table2.column_name;

SELECT column_name(s) FROM table1

RIGHT JOIN table2 **ON** table1.column_name = table2.column_name;

SELECT column_name(s) FROM table1

FULL OUTER JOIN table2 **ON** table1.column_name = table2.column_name;

SET OPERATOR:

UNION:

SELECT column_name FROM table_name1

UNION

SELECT column_name FROM table_name2

UNION ALL: This operation is similar to Union. But it also shows the duplicate rows.

INTERSECT :

SELECT column_name FROM table_name1

INTERSECT

SELECT column_name FROM table_name2

MINUS:

SELECT column_name FROM table_name1

```
MINUS

SELECT column_name FROM table_name2
```

PROGRAM NO 7: JOIN & SET OPERATIONS

Aim: To execute commands to implement join & set operations

Questions:

1. Display the result of performing left join between EMPLOYEE & PROJECT
2. Display the result of performing right join between EMPLOYEE & PROJECT
3. Display the result of performing full join between EMPLOYEE & PROJECT
4. Display the result of performing natural join between EMPLOYEE & PROJECT
5. Display salary of employees with ENO as E1 and E3 (use UNION operator)
6. Display common salary (if any) of employees working in project P1 and P2 (use INTERSECT operator)
7. Display different designations present in project P3 but not in P1 (use MINUS operator)

Result: The commands for above questions executed successfully.

BUILT – IN FUNCTIONS

1) Numeric Functions: Numeric functions are used to perform operations on numbers. They accept numeric values as input and return numeric values as output. Few of the Numeric functions are:

Function Name	Return Value
ABS (x)	Absolute value of the number '*x*'
CEIL (x)	Integer value that is Greater than or equal to the number '*x*'
FLOOR (x)	Integer value that is Less than or equal to the number '*x*'
TRUNC (x, y)	Truncates value of number '*x*' up to '*y*' decimal places
ROUND (x, y)	Rounded off value of the number '*x*' up to the number '*y*' decimal places

2) Character or Text Functions: Character or text functions are used to manipulate text strings. They accept strings or characters as input and can return both character and number values as output.

Few of the character or text functions are as given below:

Function Name	Return Value
LOWER (string_value)	All the letters in 'string_value' are converted to lowercase.
UPPER (string_value)	All the letters in 'string_value' are converted to uppercase.
INITCAP (string_value)	All the letters in 'string_value' are converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of 'trim_text' are removed from the left of 'string_value'.
RTRIM (string_value, trim_text)	All occurrences of 'trim_text' are removed from the right of 'string_value' .
TRIM (trim_text FROM string_value)	All occurrences of 'trim_text' from the left and right of 'string_value' , 'trim_text' can also be only one character long.
SUBSTR (string_value, m, n)	Returns 'n' number of characters from 'string_value' starting from the 'm' position
INSTR(string, substring, startposition,occurrence)	Returns a positive integer that is the position of a substring within a string

LENGTH (string_value)	Number of characters in 'string_value' is returned.
CONCAT(string1, string2, ..., string_n)	The function adds two or more strings together.
LPAD (string_value, n, pad_value)	Returns 'string_value' left-padded with 'pad_value' . The length of the whole string will be of 'n' characters.
RPAD (string_value, n, pad_value)	Returns 'string_value' right-padded with 'pad_value' .

3) Date Functions: These are functions that take values that are of datatype DATE as input and return values of datatypes DATE, except for the MONTHS_BETWEEN function, which returns a number as output.

Few date functions are as given below.

Function Name	**Return Value**
ADD_MONTHS (date, n)	Returns a date value after adding *'n'* months to the date *'x'*.
MONTHS_BETWEEN (x1, x2)	Returns the number of months between dates x1 and x2.
ROUND (x, date_format)	Returns the date *'x'* rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the *'date_format'*.
TRUNC (x, date_format)	Returns the date *'x'* lesser than or equal to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'.
NEXT_DAY (x, week_day)	Returns the next date of the *'week_day'* on or after the date *'x'* occurs.
LAST_DAY (x)	It is used to determine the number of days remaining in a month from the date *'x'*specified.
SYSDATE	Returns the systems current date and time.
NEW_TIME (x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

4) Conversion Functions: These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

Few of the conversion functions available in oracle are:

Function Name	**Return Value**
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x and 'y'* must be of the same datatype.

PROGRAM NO 8: BUILT IN FUNCTIONS

Aim: To execute commands to perform built in functions

Questions:

1. Display the name of employees in upper case
2. Display the name of employees in lower case
3. Display the name of employees in proper case
4. Display the length of each employee name
5. Display the name of employees and concatenate it with employee no
6. Extract 3 characters starting from 2nd character from string ‘Oracle’
7. Find the first occurrence of character ‘a’ in the string ‘COMPUTER MAINTENANCE’
8. Replace every occurrence of alphabet ‘a’ with ‘b’ in the string ‘MATHEMATICS’
9. Display the first 3 characters of all the city in EMPLOYEE table
10. Display your age in days
11. Display your age in months
12. Display the current date and time
13. Display the date 3 months before the current date
14. Display the current date in the format ‘DATE-MONTH-DAY’
15. Mr/Ms ------ who is ---------- years old is residing in ---------- Display the details of employees in the given format.(Fill blanks with ENAME, AGE,CITY respectively)

Result: Commands for above questions executed successfully.

NESTED QUERIES

IN : To compare multiple values.

SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);

SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT column_name(s) FROM table_name)

ANY, ALL, EXIST:

The ANY and ALL operators are used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The ALL operator returns true if all of the subquery values meet the condition. The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns true if the subquery returns one or more records.

SELECT column_name(s) FROM table_name WHERE column_name operator **ANY**

(SELECT column_name FROM table_name WHERE condition);

SELECT column_name(s) FROM table_name WHERE column_name operator **ALL**

(SELECT column_name FROM table_name WHERE condition);

SELECT column_name(s) FROM table_name WHERE **EXISTS**

(SELECT column_name FROM table_name WHERE condition);

PROGRAM NO 9: NESTED QUERIES

Aim: To execute commands to understand working of nested queries

Questions:

1. Find the number of employees working in PROJECT where Praveen & Aruna are not working
2. Find the number of employees whose salary is greater than average salary
3. Find the project details of employees with maximum salary
4. Find the department in which youngest employee work
5. Find the details of employee who has joined in the month of January
6. Display the month and count of employee joined in each month
7. Update the details of employee who joined in the month of March and salary greater than average salary
8. Update the salary of senior most employee by 10%
9. Find the details of employee working in a project in which Arun is not working
10. Find the details of employee working in the CSE department and display the details in alphabetical order of name

Result: Commands for above questions executed successfully.

TCL COMMANDS IN SQL

Transaction Control Language(TCL) commands are used to manage transactions in the database

1.COMMIT

COMMIT command is used to permanently save any transaction into the database.

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed, the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Following is commit command's syntax,

COMMIT;

2.SAVEPOINT

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Following is savepoint command's syntax,

SAVEPOINT savepoint_name;

3.ROLLBACK:

This command restores the database to the last committed state. It is also used with the SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have used the UPDATE command to make some changes into the database, and realise that those changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command.

Following is rollback command's syntax,

ROLLBACK TO savepoint_name;

PROGRAM NO 10: TCL COMMANDS IN SQL

Aim: Implementation of TCL commands in SQL

Questions:

1. Create a table **CLASS** with attributes id (integer type) and name (text type)
2. Insert 2 values to **CLASS** and save the entered values using commit operation.
3. Create two save points named A and B on insertion or updation of table **CLASS**
4. Display the contents of table **CLASS** before and after apply of rollback of save point B.
5. Display the contents of table **CLASS** after the rollback of save point A.

Result: Commands for above questions executed successfully.

PROGRAM NO. 11: DCL COMMANDS IN SQL

Data Control Language(DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,

- System: This includes permissions for creating session, table, etc and all types of other system privileges.
- Object: This includes permissions for any command or query to perform any operation on the database tables.

In DCL we have two commands,

- GRANT: Used to provide any user access privileges or other privileges for the database.
- REVOKE: Used to take back permissions from any user.

GRANT:

1. Allow a User to create session

When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/privileges are granted to the user.

Following command can be used to grant the session creating privileges.

GRANT CREATE SESSION TO username;

2. Allow a User to create table

To allow a user to create tables in the database, we can use the below command,

GRANT CREATE TABLE TO username;

3. Grant all privilege to a User

sysdba is a set of privileges which has all the permissions in it. So if we want to provide all the privileges to any user, we can simply grant them the sysdba permission.

GRANT sysdba TO username

4. Grant permission to create any table

Sometimes the user is restricted from creating come tables with names which are reserved for system tables. But we can grant privileges to a user to create any table using the below command,

GRANT CREATE ANY TABLE TO username

5. Grant permission to drop any table

As the title suggests, if you want to allow user to drop any table from the database, then grant this privilege to the user,

GRANT DROP ANY TABLE TO username

REVOKE:

And, if you want to take back the privileges from any user, use the REVOKE command.

REVOKE CREATE TABLE FROM username

<u>CREATION OF VIEW & ASSERTION</u>

<u>SQL VIEW:</u>

A VIEW in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

1. <u>View Creation:</u> Syntax for creating a View,

```
CREATE or REPLACE VIEW view_name
   AS
   SELECT column_name(s)
   FROM table_name
   WHERE condition
```

A view is created using data fetched from some other table(s). It's more like a temporary table created with data.

2. <u>Displaying a VIEW:</u> The syntax for displaying the data in a view is similar to fetching data from a table using a SELECT statement.

```
SELECT * FROM view_name;
```

3. <u>Force VIEW Creation</u>

FORCE keyword is used while creating a view, forcefully. This keyword is used to create a View even if the table does not exist. After creating a force View, if we create the base table and enter values in it, the view will be automatically updated. Syntax for forced View is,

```
CREATE or REPLACE FORCE VIEW view_name AS
```

```
SELECT column_name(s)

FROM table_name

WHERE condition;
```

4. Update a VIEW: UPDATE command for the view is the same as for tables. Syntax to Update a View is,

```
UPDATE view-name SET VALUE

WHERE condition;
```

If we update a view it also updates base table data automatically.

ASSERTION:

Assertions means the conditions that the database must always satisfy. Domain constraints and referential-integrity constraints are specific forms of assertions

• CHECK – verify the assertion on one-table, one-attribute

• ASSERTION – verify one or more tables, one or more attributes

When a constraint involves 2 (or) more tables, the table constraint mechanism is sometimes hard and results may not come as expected. To cover such a situation SQL supports the creation of assertions that are constraints not associated with only one table. And an assertion statement should ensure a certain condition will always exist in the database. DBMS always checks the assertion whenever modifications are done in the corresponding table.

Syntax for assertion creation:

```
CREATE ASSERTION  assertion_name

CHECK ( condition  );
```

Syntax for assertion removal:

```
 DROP ASSERTION  assertion_name ;
```

PROGRAM NO. 12: CREATION OF VIEW & ASSERTION

Aim: To implement creation of view and assertion in tables

Questions:

1. Create a view of employees whose age is in the range of 20 and 30 and display the view contents
2. Create a view to display the ENO, ENAME, DEPTNAME, PNO of employees and display the view contents.
3. Create an assertion ECOUNT to check that the total number of entries in EMPLOYEE must not exceed 100.
4. Remove the above created ECOUNT assertion

Result: Commands for above questions executed successfully.

PL / SQL

PL/SQL is a block-structured language. Each block consists of three sub-parts −

1. Declaration : This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.
2. Executable Commands : This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program.
3. Exception Handling : This section starts with the keyword EXCEPTION. This optional section contains exception(s) that handle errors in the program.

```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
EXCEPTION
   <exception handling>
END;
```

Syntax for Control Structures:

1. IF STATEMENTS:

```
IF  CONDITION  THEN
…
END IF;


IF  CONDITION  THEN
…
ELSIF
…
```

```
END IF;
```

2. LOOPS:

```
WHILE CONDITION LOOP
LOOP STATEMENTS;
END LOOP;

FOR VARIABLE IN INITIAL_VALUE .. FINAL_VALUE LOOP
  LOOP STATEMENTS;
END LOOP;

FOR VARIABLE  in  REVERSE  INITIAL_VALUE .. FINAL_VALUE  LOOP
   LOOP STATEMENTS;
  END LOOP;
```

3. CASE STATEMENTS

```
CASE [ expression ]  /TRUE
WHEN condition_1 THEN result_1
   WHEN condition_2 THEN result_2
   ...
   WHEN condition_n THEN result_n
 ELSE result
END
```

PROGRAM NO.13: CONTROL STRUCTURES IN PL/SQL

Aim: To implement different control structures like IF ELSE, WHILE,FOR, CASE etc in PL/SQL

Questions:

1. Write a PL/SQL program to find the area of the circle
2. Write a PL/SQL program to check whether a given number is palindrome or not
3. Write a PL/SQL program to print the sum first n odd numbers and even numbers separately.
4. Write a PL/SQL program to give mark to equivalent grade conversion

Marks	Grade
100>=marks>=90	OUTSTANDING
90>marks>=80	A
80>marks>=70	B+
70>marks>=60	B
60>marks>=45	C+
marks<45	FAIL

5. Write a PL/SQL program to find the total number of employees and total salary in EMPLOYEE table
6. Write a PL/SQL program to satisfy the following
   a. If salary<5000 then give 50% increment
   b. If salary between 5000 and 10000 then give 10% increment
   c. If salary>10000 then display a message “No Increment”
7. Write a PL/SQL program to display ENAME, AGE, SALARY, PNAME,DEPTNAME of a particular employee.

Result: Commands for above questions executed successfully.

<u>EXCEPTION HANDLING IN PL/SQL</u>

An error occurs during the program execution is called Exception in PL/SQL.

PL/SQL facilitates programmers to catch such conditions using exception blocks in the program and an appropriate action is taken against the error condition.

There are two type of exceptions:

- System-defined Exceptions
- User-defined Exceptions

<u>Syntax For Exceptions</u>

```
DECLARE
  <declarations section>
BEGIN
  <executable command(s)>
EXCEPTION
  <exception handling goes here >
  WHEN exception1 THEN
    exception1-handling-statements
  WHEN exception2  THEN
    exception2-handling-statements
  …..........
  WHEN others THEN
    exception3-handling-statements
END;
```

<u>Syntax for raising an exception:</u>

```
DECLARE
```

```
  exception_name EXCEPTION;
BEGIN
  IF condition THEN
    RAISE exception_name;
  END IF;
EXCEPTION
  WHEN exception_name THEN
  statement;
END;
```

PROGRAM NO.14:_ EXCEPTION HANDLING IN PL/SQL

Aim: To implement exception handling in PL/SQL

Questions:

1. Write a PL/SQL program to check whether an employee with a given EID is present in the EMPLOYEE table or not. If present display EID, ENAME & DESIGNATION of that employee otherwise display “Employee Not Present” message
2. Write a PL/SQL program to raise an exception whenever age of an employee is greater than 50
3. Write a PL/SQL program to raise a ZERO_DIVIDE exception whenever we try to perform division operation with 0 as denominator otherwise display the result

Result: Commands for above questions executed successfully.

CURSOR

A cursor holds the rows (one or more) returned by a SQL statement.SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The following table provides the description of the most used attributes −

1 %FOUND : Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.

2. %NOTFOUND : The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

3 %ISOPEN : Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.

4 %ROWCOUNT: Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

```
CURSOR c_customers IS
   SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

```
CLOSE c_customers;
```

PROGRAM NO.15: CURSORS

Aim: To implement cursors in PL/SQL

Questions:

1. Write a PL/SQL program to select only those rows where salary>5000 and update the city to ‘London’
2. Write a PL/SQL program to find average salary of each month
3. Write a PL/SQL program to display all the details of employees in EMPLOYEE table

Result: Commands for above questions executed successfully.

PROCEDURE, FUNCTIONS & TRIGGERS

PROCEDURE

A subprogram is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.

Procedures are subprograms which do not return a value directly; mainly used to perform an action.

SYNTAX:

```
CREATE [OR REPLACE] PROCEDURE procedure_name

[(parameter_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

  < procedure_body >

END procedure_name;
```

Syntax for drop procedure

```
DROP PROCEDURE procedure_name;
```

FUNCTIONS

A function is the same as a procedure except that it returns a value.

SYNTAX:

```
CREATE [OR REPLACE] FUNCTION function_name

[(parameter_name [IN | OUT | IN OUT] type [, ...])]

RETURN return_datatype

{IS | AS}

BEGIN

   < function_body >

END [function_name];
```

Syntax for removing your created function:

If you want to remove your created function from the database, you should use the following syntax.

```
DROP FUNCTION function_name;
```

TRIGGER

Triggers are stored programs, which are automatically executed or fired when some events occur.

Syntax:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.

- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer to new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

Syntax to show list of triggers associated with a table :

```
SELECT TRIGGER_NAME
FROM ALL_TRIGGERS
WHERE TABLE_NAME='tablename';
```

Syntax for enabling & disabling trigger:

When this trigger is created, the database enables it automatically. You can subsequently disable the trigger with this statement:

```
ALTER TRIGGER triggername DISABLE;
```

When the trigger is disabled, the database does not fire the trigger. After disabling the trigger, you can subsequently enable it with this statement:

```
ALTER TRIGGER triggername  ENABLE;
```

Syntax to drop trigger:

```
DROP TRIGGER triggername;
```

PROGRAM NO.16: PROCEDURE, FUNCTIONS & TRIGGER

Aim: To implement procedure , functions and triggers in PL/SQL

Questions:

1. Create a procedure named INSERTDEPT to insert new department details into DEPARTMENT table
2. Create a procedure MAXSAL to find maximum salary of employee working in a particular project
3. Create a function FINDMAX to find largest of two numbers
4. Create a recursive function FINDFACT to find factorial of a number
5. Create a function AVGSAL to find the average salary of all employees in the EMPLOYEE table
6. Create a database trigger which is triggered before deletion of each row from PROJECT table
7. Create a database trigger which is triggered before insert/update/delete for each row of the DEPARTMENT table. It should check that no alteration is made in the DEPARTMENT table on Wednesday, Thursday and Friday.
8. Create a database trigger to not do updation on EMPLOYEE if new salary is lesser than the current salary
9. Create a database trigger to restrict deletion from EMPLOYEE if age is greater than 60
10. Create a database trigger to insert to EMPLOYEE only if city is Bombay
11. Show all the triggers applied to the EMPLOYEE , PROJECT & DEPARTMENT table.
12. Remove all the above created triggers .

Result: Commands for above questions executed successfully.

PACKAGES

A package is a schema object that groups logically related PL/SQL types, variables, constants, subprograms, cursors, and exceptions. A package is compiled and stored in the database, where many applications can share its contents.

A package will have two mandatory parts −

- Package specification
- Package body or definition

Package Specification

The specification is the interface to the package. It just declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.

All objects placed in the specification are called public objects. Any subprogram not in the package specification but coded in the package body is called a private object.

Syntax:

```
CREATE [OR REPLACE ] PACKAGE package_name AS

<variable/procedure declaration>

END package_name;
```

Package Body

The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package.

Syntax:

```
CREATE [OR REPLACE] PACKAGE BODY package_name AS

<private element definition>

<subprogram definition> IS

BEGIN

<package initialization code>

END package_name;
```

Using the Package Elements

The package elements (variables, procedures or functions) are accessed with the following syntax −

```
Package_name.element_name;
```

PROGRAM NO.17: PACKAGES IN PL/SQL

Aim: To implement packages in PL/SQL

Questions:

1. Create a package ESAL which contains a procedure FINDSAL to find the salary of a particular employee.
2. Create a package DEPTUPDT which contains procedures
    a. DINSERT to add new records to DEPARTMENT table
    b. DDEL to delete a particular record from DEPARTMENT table
    c. DDISPLAY to display the contents of DEPARTMENT table

Result: Commands for above questions executed successfully.

PROGRAM NO 18: NOSQL & CRUD OPERATIONS

The conventional database is a SQL database system that uses a tabular relational model to represent data and their relationship. The NoSQL database is the newer one database that provides a mechanism for storage and retrieval of data other than the tabular relations model used in relational databases.

NoSQL databases are categorized as Non-relational or distributed database systems. NoSQL databases have dynamic schemas. NoSQL databases display data as collection of key-value pairs, documents, graph databases or wide-column stores.NoSQL databases are best suited for hierarchical data storage. MongoDB, BigTable, Redis, RavenDB, Cassandra, Hbase, Neo4j, CouchDB etc. are examples of nosql databases.

CRUD Operations: The acronym CRUD stands for create, read, update and delete. These are the four basic functions of persistent storage.

MongoDB is a Nosql database. MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod

MongoDB CRUD Operations

CRUD operations denote *create*, *read*, *update*, and *delete* documents.

a. Create a Collection

First Select Or Create a database.

```
> use mydb
```

```
switched to db mydb
```

Using db.createCollection("yourCollectionName") method you can explicitly create Collection.

```
> db.createCollection("newCollection1")
```

```
{ "ok" : 1 }
```

Using show collections command see all collections in the database.

```
> show collections
```

```
newCollection1

system.indexes
```

The db.createCollection() method has the following parameters:

Parameter	**Type**	**Description**
name	string	The name of the collection to create.
options	document	Optional. Configuration options for creating a capped collection or for preallocating space in a new collection.

The following example shows the syntax of createCollection() method with few important options

```
>db.createCollection("newCollection4", {capped :true, autoIndexId : true, size : 6142800, max: 10000})

{ "ok" : 1 }
```

Both the db.collection.insert() and the db.collection.createIndex() operations create their respective collection if they do not already exist.

```
> db.newCollection2.insert({name : "XXX"})
> db.newCollection3.createIndex({accountNo : 1})
```

Now, Show All the collections using show collections command

```
> show collections
newCollection1
newCollection2
newCollection3
newCollection4
System.indexes
```

b. Read a Collection

If you want to see the inserted document, use the find() command.

```
> db.newCollection2.find()

{ "_id" : ObjectId("58f26876cabafaeb509e9c1f"), "name" : "XXX" }
```

c. Update a Collection

```
>db.collection.update(query, update, options)
```

The method returns a WriteResult document that contains the status of the operation.

d. <u>Drop/Delete a Collection</u>

MongoDB's db.collection.drop() is used to drop a collection from the database.
First, check the available collections into your database mydb.

```
> use mydb
switched to db mydb
> show collections
newCollection1
newCollection2
newCollection3
system.indexes
```

Now drop the collection with the name newCollection1.

```
> db.newCollection1.drop()
True
```

Note: If the collection dropped successfully then the method will return true otherwise it will return false.
Again check the list of collections into the database.

```
> show collections
newCollection2
newCollection3
system.indexes
```

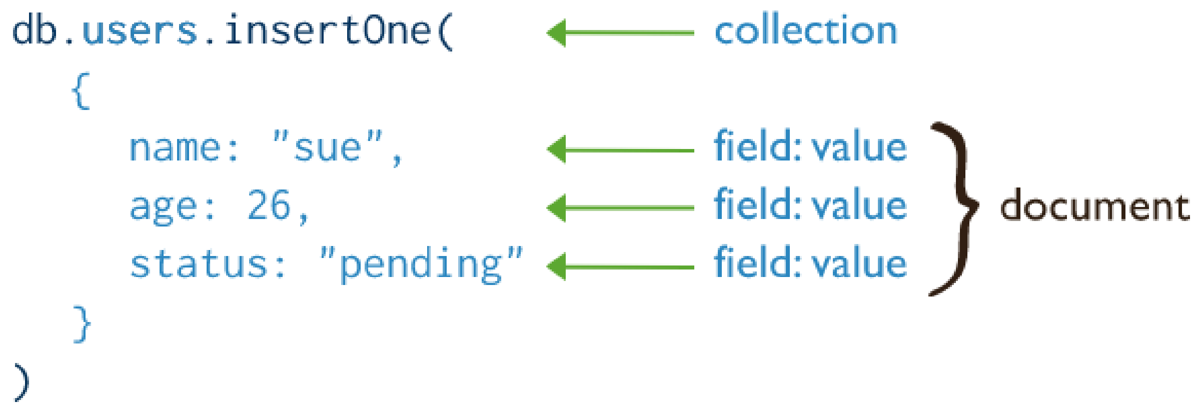
<u>Create Operations in Documents</u>

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany()

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.



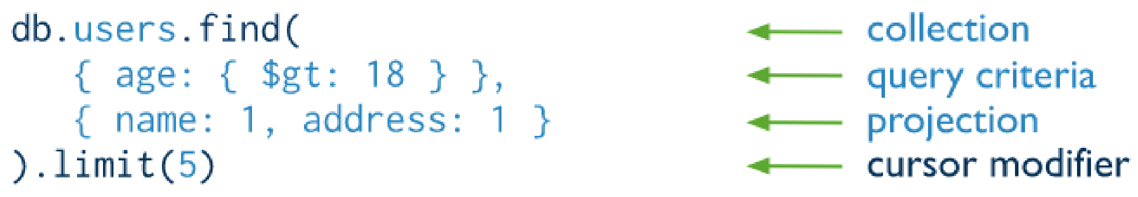
```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

Read Operations in Documents

Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- db.collection.find()

You can specify query filters or criteria that identify the documents to return.



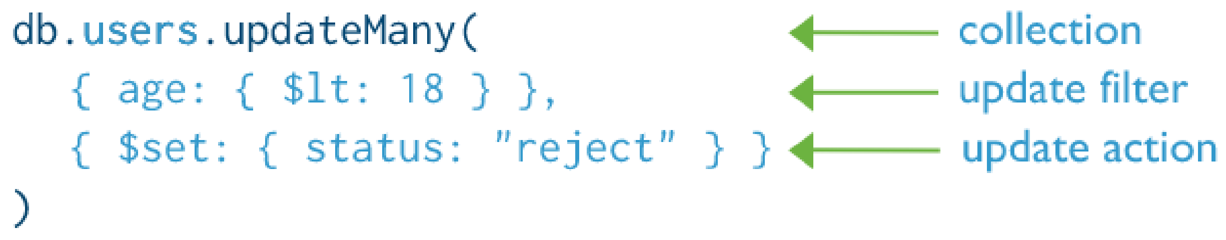
```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

Update Operations in Documents

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

- db.collection.updateOne()
- db.collection.updateMany()
- db.collection.replaceOne()

In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.



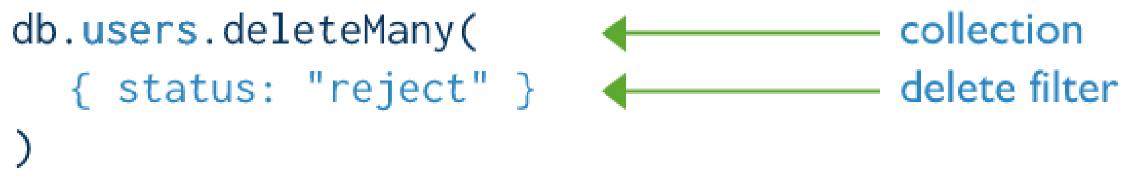
```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

Delete Operations in Documents

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- db.collection.deleteOne()
- db.collection.deleteMany()

In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.



```
db.users.deleteMany(
   { status: "reject" }
)
```

PROGRAM NO 19: APPLICATION DEVELOPMENT USING JAVA & SQL

Note: Done by students in groups

Topics:

1. Railway Reservation System
2. Library Management
3. Hotel Reservation system
4. Blood Bank System
5. Vaccination portal
6. Game Launcher etc..