EXPERIMENT- 01

Data Definition Language (DDL),Data Manipulation Language (DML) and Data Query Language(DQL) Commands.

1.1 AIM OF THE EXPERIMENT:

To practice and implement data definition language, data manipulation language and data query language commands.

1.3 BACKGROUND THEORY :

Structured Query Language (SQL) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). It is used to view or change data in databases. The sentences used in this language are called SQL Queries.SQL contains different data types those are CHAR(size), VARCHAR2(size), NUMBER, NUMBER(p,s), DATE, CLOB, LONG, RAW, LONG RAW, BLOB, ROWID, BFILE.

SQL commands are instructions, coded into SQL statements, which are used to communicate with the database to perform specific tasks, work, functions and queries with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users. SQL commands are grouped into four major categories depending on their functionality:

- Data Definition Language (DDL) - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.
- Data Manipulation Language (DML)- These SQL commands are used for storing, retrieving, modifying, and deleting data. These Data Manipulation Language commands are: INSERT, UPDATE, and DELETE.
- Data Query Language (DQL): . This command is used to retrieve the records / data from database. The Data Query Language command is SELECT.
- Transaction Control Language (TCL) - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.
- Data Control Language (DCL) - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

1.2.1. RULES FOR COMMANDS

i) Oracle reserved words cannot be used

ii) Underscore, numerals, letters are allowed but not blank space.

iii).Maximum length for the table name is 30characters.

iv) Two different tables should not have same name.

v) We should specify a unique column name.

vi) We should specify proper data type along with width.

vii)We can include "not null" condition when needed. By default it is "null.

1.2.2. DESCRIBE COMMAND:

It is used to describe a schema i.e. it provides information about the columns in a table.

Syntax: DESC tablename

Example: DESC Persons;

1.2.3. DDL COMMANDS

1.2.3.1. CREATE COMMAND

- It is used to create a table.
- It defines each column of the table uniquely. Each column has minimum of three attributes, a name, data type and size.

Syntax:

```
CREATE TABLE tablename
        (
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
        );
```

Examples:

1.
```
CREATE TABLE Persons
(
        PersonID        int,
LastName        varchar(255),
FirstName       varchar(255),
Address                 varchar(255),
City    varchar(255)
);
```

Output:

Table Created

To view Structure of created table

```
SQL> DESC Persons;
PersonID        int
LastName        varchar2(255)
FirstName       varchar2(255)
Address                 varchar2(255)
City            varchar2(255)
```

2.
```
CREATE TABLE Client_Master
(
ClientNo        varchar2(6),
Name            varchar2(20),
Address1        varchar(30),
Address         2       varchar(30),
City            varchar2(15),
PinCode         Number(8),
State           varchar2(15)
);
```

Output:

Table Created

```
SQL> DESC Client_Master;
ClientNo        varchar2(6)
Name            varchar2(20)
Address1        varchar(30)
Address         2       varchar(30)
City            varchar2(15)
PinCode         Number(8)
State           varchar2(15)
```

1.2.3.2. ALTER COMMAND

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

Syntax: i) To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

ii) To delete a column in a table, use the following syntax

ALTER TABLE table_name

DROP COLUMN column_name

iii) To change the data type of a column in a table, use the following syntax:

ALTER TABLE table_name

MODIFY COLUMN column_name datatype

Examples:

i)If we want to add a column named "DateOfBirth" in the "Persons" table ,the following SQL statement is used:

```
ALTER TABLE Persons
ADD DateOfBirth date
```

Output:

Table Altered

```
SQL>DESC Person;
PersonID        int
LastName        varchar(255)
FirstName       varchar(255)
Address                 varchar(255)
City             varchar(255)
DateOfBirth     date
```

ii) If we want to change the data type of the column named "DateOfBirth" in the "Persons" table, the following SQL statement is used:

```
ALTER TABLE Persons
ALTER COLUMN DateOfBirth year
```

Output:

Table Altered

```
SQL>DESC Persons;
PersonID        int
LastName        varchar(255)
FirstName       varchar(255)
Address         varchar(255)
City             varchar(255)
DateOfBirth      year
```

iii) If want to delete the column named "DateOfBirth" in the "Persons" table,the following SQL statement is used:

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth
```

Output:

```
Table Altered

SQL>DESC Persons;
PersonID      int
LastName      varchar(255)
FirstName     varchar(255)
Address             varchar(255)
City          varchar(255)
DateOfBirth   year
```

1.2.3.3. DROP COMMAND

The SQL **DROP TABLE** statement is used to remove a table definition and all data, indexes, triggers, constraints, and permission specifications for that table.

Syntax: DROP TABLE table_name;

Examples: DROP TABLE Client_Master

Output:

```
Table dropped
SQL>DESC Client_Master;
Table Client_Master doesn't exist.
```

1.2.3.4. TRUNCATE COMMAND:

The SQL **TRUNCATE TABLE** command is used to delete complete data from an existing table. The DROP TABLE command is also used to delete complete table but it would remove complete table structure form the database and it need to be re-created the table once again if some data has to be restored.

Syntax:TRUNCATE TABLE table_name;

Examples: Truncate table Persons

Output:

Table truncated

Query OK, 0 rows affected

SQL>DESC person;

PersonID	int
LastName	varchar(255)
FirstName	varchar(255)
Address	varchar(255)
City	varchar(255)
DateOfBirth	year

1.2.4. DQL COMMAND

DQL is abbreviation of Data Query Language. It used to retrieve the records / data from database. The DQL command is SELECT.

1.2.4.1 SELECTCOMMANDS

i) Selects all rows from the table

Syntax: Select * from tablename;

Example: Select * from PersonID;

iii) The retrieval of specific columns from a table

Syntax: Select column_name1,.....,column_name n from tablename;

Example: Select personID, FirstName from emp;

iv) **Elimination of duplicates from the select clause:**

Syntax: Select DISTINCT col1,col2 from table name;

Example: Select DISTINCT City from Persons;

v) **Select command with where clause:**

Syntax: Select column_name1,.....,column_name n from tablename where condition;

Example: Select PersonID, FirstName, City from Persons where PersonID>4;

vi) **Select command with order by clause:**

Syntax: Select column_name1,.....,column_name n from tablename where condition orderby colmnname;

Example: Select PersonID, FirstName from Persons orderby PersonID;

vi) **Select command to create a table:**

Syntax: create table tablename as select * from existing_tablename;

Example: create table Persons1 as select * from Persons;

SQL>DESC Persons1;

PersonID	int
LastName	varchar(255)
FirstName	varchar(255)
Address	varchar(255)
City	varchar(255)
DateOfBirth	year

1.2.5. DML COMMANDS

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are INSERT, UPDATE, and DELETE.

1.2.5.1 INSERT COMMAND

This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defines.

i) Inserting a single row into a table:

Syntax: insert into <table name> values (value list);

Example: insert into Persons values (1,"Flia","Bing","Torrento","Sandnes",).;

Output:

SQL>Select * from Persons;

PersonID	LastName	FirstName	Address	City
1	Flia	Bing	Torrento	Sandnes

ii) Inserting more than one record using a single insert commands:

Syntax: insert into <table name>values (&col1,&col2,....)

Example: insert into Persons values (&PersonID, &Lastname, &Firstnmae, & Address, &City);

iii) Skipping the fields while inserting:

Syntax: insert into <tablename (column names to which datas to be inserted)> values(list of values);

Other way is to give null while passing the values.

Example: insert into Persons (PersonID, FirstName,City) values(2,"Tove","Stavanger");

SQL>Select * from Persons;

PersonID	LastName	FirstName	Address	City
1	Flia	Bing	Torrento	Sandnes
2		Tove	Stavanger	

1.2.5.2 UPDATE COMMAND

Syntax: update table name set field=values where condition;

Example: update Persons set city="Mumbai" where PersonID=1;

SQL>Select * from Persons;

PersonID	LastName	FirstName	Address	City
1	Flia	Bing	Torrento	Mumbai
2		Tove	Stavanger	

1.2.5.3 DELETECOMMAND

Syntax: Delete from table where conditions;

Example: delete from Persons where PersonID=2;

PersonID	LastName	FirstName	Address	City
1	Flia	Bing	Torrento	Mumbai

1.3. ASSIGNMENT QUESTIONS:

Q1. **Database Schema for a employee scenario**

Consider the following tables namely "Departments" ,"Employeses" and "Saalry"

Departments (DEPTNO , DEPTNAME , DEPTLOC);

Employees (EMPNO, EMPNAME , JOB,DEPTNO,DOJ);Salary(EMPNO,MONTH/YEAR,SAL)

For the above schema, perform the following—

1. Create all above tables.
2. Rename Employee table as Employee_Master
3. Add column Date_of_Birth to Employee_Master

4. Delete Date_of_Birth for Employee_Master
5. Modify EMPNAME field to accept 100 characters.
6. Insert around 10 records in each of the tables
7. List all the employees with their EMPNO.
8. List all the employee numbers whose salary is above 20000 .
9. List all the employees who have joined in the year 2002.
10. List all the department numbers where department location is Odisha.
11. Truncate the Employee_Master table.

Q2. Database Schema for a Student Library scenario

Student(Stud_no : integer, Stud_name: string)
Membership(Mem_no: integer, Stud_no: integer)
Book(book_no: integer, book_name:string, author: string)
Iss_rec(iss_no:integer, iss_date: date, Mem_no: integer, book_no: integer)

For the above schema, perform the following—

1. Create the all above tables .
2. Insert around 10 records in each of the tables
3. Add column S_Address to Employee_Master
4. List all the student names with their membership numbers.
5. List all the issues for the current date with student and Book names.
6. List the details of students who borrowed book whose author is CJDATE.
7. Give a list of books taken by student with stud_no as 5.
8. Truncate the table Iss_rec.

1.4 RESULT:

Thus the data definition language, data manipulation language and data query language commands were performed and implemented successfully.

1.5 QUESTIONS AND ANSWERS:

1. What is SQL?

Structured Query Language (SQL) is a language designed specifically for communicating with databases. SQL is an ANSI (American National Standards Institute) standard.

2. What are the different types of SQL?

1. DDL – Data Definition Language

DDL is used to define the structure that holds the data. For example, table.

2. DML– Data Manipulation Language

DML is used for manipulation of the data itself. Typical operations are Insert, Delete, Update and retrieving the data from the table.

3. DCL–Data Control Language

DCL is used to control the visibility of data like granting database access and set privileges to create tables etc.

3. What are the Advantages of SQL

a. SQL is not a proprietary language used by specific database vendors. Almost every major DBMS supports SQL, so learning this one language will enable programmers to interact with any database like ORACLE, SQL ,MYSQL etc.
b. SQL is easy to learn. The statements are all made up of descriptive English words, and there aren't that many of them.
c. SQL is actually a very powerful language and by using its language elements you can perform very complex and sophisticated database operations.

4. **What is a Database system?**

The database and DBMS software together is called as Database system.

5. **What are the advantages of DBMS?**

1. Redundancy is controlled.
2. Unauthorised access is restricted.
3. Providing multiple user interfaces.
4. Enforcing integrity constraints.
5. Providing backup and recovery.

6. Describe the three levels of data abstraction?

There are three levels of abstraction:

1. **Physical level:** The lowest level of abstraction describes how data are stored.
2. **Logical level:** The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
3. **View level:** The highest level of abstraction describes only part of entire database.

6. What is a Record in a database?

A record is the collection of values / fields of a specific entity: i.e. an Employee, Salary etc.

EXPERIMENT- 02

Data Control Language (DCL) and Transaction Control Language (TCL) Commands.

2.1 AIM OF THE EXPERIMENT:

To practice and implement data control and transaction control language commands.

2.2 BACKGROUND THEORY:

2.2.1 DCL COMMAND:

Data Control Language (DCL) command is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE.

2.2.1.1 GRANT COMMAND

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax:

GRANT privilege_name ON object_name TO {user_name | PUBLIC | role_name}
[WITH GRANT OPTION];

- **privilege_name** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- **object_name** is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.
- **user_name** is the name of the user to whom an access right is being granted.
- **user_name** is the name of the user to whom an access right is being granted.
- **PUBLIC** is used to grant access rights to all users.
- **ROLES** are a set of privileges grouped together.
- **WITH GRANT OPTION** - allows a user to grant access rights to other users.

Example:

GRANT SELECT ON employee TO user1;

This command grants a SELECT permission on employee table to user1.

2.2.1.2 REVOKE COMMAND

The REVOKE command removes user access rights or privileges to the database objects.

Syntax:

REVOKE privilege_name ON object_name FROM {user_name | PUBLIC | role_name}

Example:

REVOKE SELECT ON employee FROM user1;

This command will REVOKE a SELECT privilege on employee table from user1

2.2.2. TCL COMMANDS:

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

Transaction control commands manage changes made by DML commands. These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, SAVEPOINT and SET TRANSACTION.

2.2.2.1 COMMIT COMMAND

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command.

Syntax:

```
COMMIT;
```

Example:

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Following is the example which would delete records from the table having age = 25 and then COMMIT the changes in the database.

```
SQL> DELETE FROM CUSTOMERS
   WHERE AGE = 25;
SQL> COMMIT;
SQL> SELECT * FROM CUSTTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
3	kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

2.2.2.2 ROLL BACK COMMAND

The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

Syntax:

```
ROLLBACK;
```

Example:

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

```
SQL> DELETE FROM CUSTOMERS
   WHERE AGE = 25;
SQL> ROLLBACK;
SQL> SELECT * FROM CUSTTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

2.2.2.3 SAVEPOINT COMMAND

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

i)Creation of SAVEPOINT among transactional statements.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

ii) Rolling back to a SAVEPOINT

Syntax:

ROLLBACK TO SAVEPOINT_NAME;

Example:

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, here is the series of operations:

```
SQL>SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM  CUSTOMERS WHERE ID=1;
1 row deleted.
SQL>SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM  CUSTOMERS WHERE ID=2;
1 row deleted.
SQL>SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM  CUSTOMERS WHERE ID=3;
1 row deleted.
```

Now that the three deletions have taken place, say you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP2. Because SP2 was created after the first deletion, the last two deletions are undone:

SQL> ROLLBACK TO SP2;
Rollback complete.

Notice that only the first deletion took place since you rolled back to SP2:

SQL> SELECT * FROM CUSTOMERS;

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

6 rows selected.

2.2.2.4 SET TRANSACTION COMMAND:

The SET TRANSACTION command can be used to initiate a database transaction. This command is used to specify characteristics for the transaction that follows.

Syntax:

SET TRANSACTION [READ WRITE | READ ONLY];

ASSIGNMENT QUESTIONS:

Considering the Employee table with the following schema:
Departments (DEPTNO , DEPTNAME , DEPTLOC);
Employees (EMPNO, EMPNAME , JOB,DEPTNO,DOJ);Q1: Develop a query to grant all privileges of employees table into departments table
Q2: Develop a query to grant some privileges of employees table into departments table
Q3: Develop a query to revoke all privileges of employees table from departments table.
Q4: Develop a query to revoke some privileges of employees table fro departments table
Q5: Write a query to implement the save point
Q6: Write a query to implement the rollback

RESULT:

Thus the Data Control Language and Transaction language commands were performed and implemented successfully.

QUESTIONS AND ANSWERS:

1. Define DCL?

The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated.

2. List the DCL commands used in data bases

The privilege commands are namely, Grant and Revoke

3. What type of privileges can be granted?

The various privileges that can be granted or revoked are,

- Select
- Insert
- Delete
- Update
- References
- Execute
- All

4. Write the syntax for grant command

Grant <database_priv [database_priv.....] >to <user_name>identified by <password> [,<password....];

Grant <object_priv>| All on <object>to <user | public>[With Grant Option];

5.What are TCL commands?

*Commit *Rollback *save point

EXPERIMENT- 03

PROGRAMS USING CONSTRAINT AND INBUILT FUNCTIONS

3.1 AIM OF THE EXPERIMENT:

To practice and implement the commands using constraint and inbuilt functions.

3.2 BACKGROUND THEORY:

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into following two types,

- **Column level constraints :** limits only column data
- **Table level constraints :** limits whole table data
- Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.
- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

3.2.1.1 NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value. One important point to note about NOT NULL constraint is that it cannot be defined at table level.

Example **using NOT NULL constraint**

CREATE table Student(s_id int NOT NULL, Name varchar(60), Age int);

The above query will declare that the **s_id** field of **Student** table will not take NULL value.

3.2.1.2 UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. UNIQUE constraint can be applied at column level or table level.

Example **using UNIQUE constraint when creating a Table (Table Level)**

CREATE table Student(s_id int NOT NULL UNIQUE, Name varchar(60), Age int);

The above query will declare that the **s_id** field of **Student** table will only have unique values and won't take NULL value.

Example **using UNIQUE constraint after Table is created (Column Level)**

ALTER table Student add UNIQUE(s_id);

The above query specifies that **s_id** field of **Student** table will only have unique value.

3.2.1.3 Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

Example **using PRIMARY KEY constraint at Table Level**

CREATE table Student (s_id int **PRIMARY KEY**, Name varchar(60) NOT NULL, Age int);

The above command will creates a PRIMARY KEY on the **s_id**.

Example **using PRIMARY KEY constraint at Column Level**

ALTER table Student add PRIMARY KEY (s_id);

The above command will creates a PRIMARY KEY on the **s_id**.

3.2.1.4 Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see it using two table.

Customer_Detail Table :

c_id	Customer_Name	Address
101 Adam	Noida	
102 Alex	Delhi	
103 Stuart	Rohtak	

Order_Detail Table :

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In **Customer_Detail** table, c_id is the primary key which is set as foreign key in **Order_Detail** table. The value that is entered in c_id which is set as foreign key in **Order_Detail** table must be present in **Customer_Detail** table where it is set as primary key. This prevents invalid data to be inserted into c_id column of **Order_Detail** table.

Example **using FOREIGN KEY constraint at Table Level**

CREATE table Order_Detail(order_id int PRIMARY KEY,
order_name varchar(60) NOT NULL,
c_id int **FOREIGN KEY** REFERENCES **Customer_Detail**(c_id));

In this query, c_id in table Order_Detail is made as foriegn key, which is a reference of c_id column of Customer_Detail.

Example **using FOREIGN KEY constraint at Column Level**

ALTER table Order_Detail add **FOREIGN KEY** (c_id) REFERENCES Customer_Detail(c_id);

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Example **using CHECK constraint at Table Level**

create table Student(s_id int NOT NULL **CHECK(s_id > 0)**,Name varchar(60) NOT NULL,Age int);

The above query will restrict the s_id value to be greater than zero.

3.2.2. INBUILT FUNCTIONS

Function is a group of code that accepts zero or more arguments and both return one or more results. Function can be classifies into **single row function and group functions**.

3.2.2.1 SINGLE ROW FUNCTIONS

A single row function or scalar function returns only one value for every row queries in table. Single row function can appear in a select command and can also be included in a where clause. The single row function can be broadly classified as,

- Date Function
- Character Function
- Miscellaneous Function
- Numeric Function
- Conversion Function

3.2.2.1.1 DATE FUNCTION

1. Add_month

This function returns a date after adding a specified date with specified number of months.

Syntax: Add_months(d,n); where d-date n-number of months

Example: Select add_months(sysdate,2) from dual; **2. last_day**

It displays the last date of that month. **Syntax:** last_day (d); where d-date

Example: Select last_day („1-jun-2009) from dual; **3. Months_between**

It gives the difference in number of months between d1 & d2. **Syntax:** month_between (d1,d2); where d1 & d2 -dates

Example: Select month_between („1-jun-2009 , 1-aug-2009) from dual;

.4. next_day

It returns a day followed the specified date. **Syntax**: next_day (d,day);

Example: Select next_day (sysdate, wednesday) from dual **5. round**

This function returns the date, which is rounded to the unit specified by the format model.

Syntax : round (d,[fmt]);

where d- date, [fmt] – optional. By default date will be rounded to the nearest day **Example:** Select round (to_date(„1-jun-2009 , dd-mm-yy), year) from dual; Select round („1-jun-2009 , year) from dual;

3.2.2.1.2 NUMERICAL FUNCTIONS

Command	Query	Output
Abs(n)	Select abs(-15) from dual;	15
Ceil(n)	Select ceil(55.67) from dual;	56
Exp(n)	Select exp(4) from dual; Select	54.59
Floor(n)	floor(100.2) from dual; Select	100 16
Power(m,n)	power(4,2) from dual; Select	1
Mod(m,n)	mod(10,3) from dual;	100.26
Round(m,n)	Select round(100.256,2) from dual;	100.23
Trunc(m,n)	Select trunc(100.256,2) from dual;	4
Sqrt(m,n)	Select sqrt(16) from dual;	

3.2.2.1.3 CHARACTER FUNCTIONS

Command	Query	Output
initcap(char);	select initcap("hello") from dual;	Hello
lower (char);	select lower („HELLO) from dual;	hello
upper (char);	select upper („hello) from dual;	HELLO
ltrim (char,[set]);	select ltrim („cseit , „cse) from dual; select	it
rtrim (char,[set]);	rtrim („cseit , „it) from dual;	cse
replace (char,search string, replace string);	select replace(„jack and jue ,„j ,„bl) from dual;	black and blue Form
substr (char,m,n);	select substr („information , 3, 4) from dual;	

3.2.2.1.4 CONVERSION FUNCTIONS

1. to_char()

Syntax: to_char(d,[format]);

This function converts date to a value of varchar type in a form specified by date format. If format is negelected then it converts date to varchar2 in the default date format.

Example: select to_char (sysdate, dd-mm-yy) from dual;

2. to_date()

Syntax: to_date(d,[format]);

This function converts character to date data format specified in the form character.

Example: select to_date(„aug 15 2009 , mm-dd-yy) from dual;

3.2.2.1.5 MISCELLANEOUS FUNCTIONS

1. uid – This function returns the integer value (id) corresponding to the user currently logged in.

Example: select uid from dual;

2. user – This function returns the logins user name. **Example:** select user from dual;

3. nvl – The null value function is mainly used in the case where we want to consider null values as zero.

yntax; nvl(exp1, exp2)

If exp1 is null, return exp2. If exp1 is not null, return exp1. **Example:** select custid, shipdate, nvl(total,0) from order;

4. vsize: It returns the number of bytes in expression. **Example:** select vsize(„tech) from dual;

3.2.2.2 GROUP FUNCTIONS

A group function returns a result based on group of rows

1. avg - It computes the average of a column
   Example: select avg (total) from student;
2. max - It returns the sum of all values in a column
   Example: select max (percentagel) from student;
3. min - It computes the smallest value in a column
   Example: select min (marksl) from student;
4. sum - It computes the largest value in a column
   Example: select sum(price) from product;

3.2.2.3 COUNT FUNCTION

In order to count the number of rows, count function is used.

1. count(*) – It counts all, inclusive of duplicates and nulls.

Example: select count(*) from student;

2. count(col_name)– It avoids null value.

Example: select count(total) from order;

3. count(distinct col_name) – It avoids the repeated and null values.

Example: select count(distinct ordid) from order;

3.2.2.4 GROUP BY CLAUSE

This allows us to use simultaneous column name and group functions.

Example: Select max(percentage), deptname from student group by deptname;

3.2.4 HAVING CLAUSE

This is used to specify conditions on rows retrieved by using group by clause. Example:
Select max(percentage), deptname from student group by deptname having
count(*)>=50;

3.2.5 SPECIAL OPERATORS:

In / not in – used to select a equi from a specific set of values

Any - used to compare with a specific set of values

Between / not between – used to find between the ranges

Like / not like – used to do the pattern matching

3.3 ASSIGNMENT QUESTIONS:

Considering Database Schema for a employee scenario

Departments (DEPTNO , DEPTNAME , DEPTLOC);

Employees (EMPNO, EMPNAME , JOB,DEPTNO,DOJ);Salary(EMPNO,MONTH/YEAR,SAL)

Q1. Create the table with all the required constraints.

Q2. Display all the details of the records whose employee name starts with ''A'.

Q3. Display all the details of the records whose employee name does not starts with A'.

Q4. Display the rows whose salary ranges from 15000 to 30000.

Q5. Calculate the total and average salary amount of the saary table.

Q6. Count the total records in the emp table.

Q7. Determine the max and min salary and rename the column as max_salary and min_salary.

Q9. Display the last day of that month in ¯05-Oct-09 .

Q10. Find how many job titles are available in employee table.

Q11. What is the difference between maximum and minimum salaries of employees in the organization?

Q12. Display the maximum salary of a group by month_year.

3.4 RESULT:

Thus the commands using constraint and inbuilt functions were implemented successfully.

3.5 QUESTIONS AND ANSWERS:

1. Define function?

Function is a group of code that accepts zero or more arguments and both return one or more results. Both are used to manipulate individual data items.

2. Write the two types of functions

i. Single row functions

ii. Group functions

3. What are single row functions?

A single row function or scalar function returns only one value for every row queries in table. Single row function can appear in a select command and can also be included in a where clause. The single row function can be broadly classified as,

- Date Function
- Character Function
- Miscellaneous Function
- Numeric Function
- Conversion Function

4. List some character funcitons

initcap(char);

lower (char);

upper (char);

ltrim (char,[set]); rtrim (char,[set]);

EXPERIMENT- 04

NESTED QUERIES AND JOIN QUERIES

4.1 AIM OF THE EXPERIMENT :

To perform nested Queries and joining Queries using DML command.

4.2 THEORY:

4.2.1 NESTED QUERIES:

Nesting of queries one within another is known as a nested queries.

Example: select ename, eno, address where salary >(select salary from employee where ename =jones);

4.2.2 Sub Queries:

The query within another is known as a sub query. A statement containing sub query is called parent statement. The rows returned by sub query are used by the parent statement.

4.2.2.1 Types of Sub Queries:

1. Sub queries that return several values

Sub queries can also return more than one value. Such results should be made use along with the operators in and any.

Example: select ename, eno, from employee where salary <any (select salary from employee where deptno =10);

2. Multiple queries

Here more than one sub query is used. These multiple sub queries are combined by means of „and & „or keywords

3. Correlated sub query

A sub query is evaluated once for the entire parent statement whereas a correlated Sub query is evaluated once per row processed by the parent statement.

Example: select * from emp x where x.salary >(select avg(salary) from emp where deptno =x.deptno);

4.2.3 Relating Data through Join Concept

The purpose of a join concept is to combine data spread across tables. A join is actually performed by the „where clause which combines specified rows of tables.

Syntax; select columns from table1, table2 where logical expression;

4.2.3.1 Types of Joins

1. Simple Join 2. Self Join 3. Outer Join 4. Inner Join

1. Simple Join

a) Equi-join: A join, which is based on equalities, is called equi-join.

Example: select * from item, cust where item.id=cust.id;

b) Non Equi-join: It specifies the relationship between table aliases.

Example: select * from item, cust where item.id<cust.id;

Table aliases are used to make multiple table queries shorted and more readable. We give an alias name to the table in the „from clause and use it instead of the name throughout the query.

2.Self join:

Joining of a table to itself is known as self-join. It joins one row in a table to another. It can compare each row of the table to itself and also with other rows of the same table.

Example: select * from emp x ,emp y where x.salary >=(select avg(salary) from x.emp where x. deptno =y.deptno);

3.Outer Join:

It extends the result of a simple join. An outer join returns all the rows returned by simple join as well as those rows from one table that do not match any row from the table. The symbol (+) represents outer join.

Example: select ename, job, dname from emp, dept where emp.deptno (+) =dept.deptno;

4.Inner join:

Inner join returns the matching rows from the tables that are being joined

Example: select e1.Username, e1.FirstName, e1.LastName, e2.DepartmentName from Employee e1 inner join Departments e2 on e1.DepartID=e2.id

4.3 ASSIGNMENT QUESTIONS:

Consider the following tables namely "DEPARTMENTS" and "EMPLOYEES"

Their schemas are as follows ,

Departments (DEPTNO , DEPTNAME , DEPTLOC);

Employees (EMPNO, EMPNAME , JOB,DEPTNO,DOJ);Salary(EMPNO,MONTH_YEAR,SAL)

Q1: Display all employee number and salary whose salary is greater than minimum salary of the company .

Q2: Issue a query to find all the employees who work in the same job as Arjun.

Q3: Issue a query to display information about employees who earn more than any employee in dept 1.

Q4..Employee Table

EMPNO	EMPNAME	JOB	DEPTNO	DOJ
e001	John S	Manager	01	1/12/2011
e002	Staven G	Clerk	03	1/3/2009
e003	Paula B	Peon	02	1/4/2012
e004	James S	Clerk	01	1/5/2012
e005	Rama C	Admin	02	10/12/2010
e009	Jagan D	Admin	02	12/12/2013

Salary Table

EMPNO	MONTH_YEAR	SAL
E001	12_2014	30000
e002	12_2014	5000
e003	12_2014	6000
e004	12_2014	7000
e005	12_2014	8000
e009	12_2014	9000

Department Table

DEPTNO	DEPTNAME	LOC
01	Sales	Loc1
02	Prod	Loc2
03	Admin	Loc3

Q1. Show EMPNO,EMPNAME,DEPTNAME from Employee and Department Table.

Q2. Add rows to Salary table.

EMPNO	MONTH_YEAR	SAL
e001	11_2014	30,500
e001	10_2014	31000
E001	09_2014	29900
e002	11_2014	6000
e002	10_2014	7000

Q3. **Show** EMPNO, EMPNAME, 12_2004 month's Salary, 11_2014 month's salary, 10_2014 month's salary for employee E001.

Q4. Show EMPNO, EMPNAME, 12_2014 month salary for all employee.

Q5. Add to row to DEPARTMENT table as fallows.

DEPTNO	DEPTNAME	LOC
04	HRD	Loc4

Q6. Show DEPTNO, DEPTNAME, TOTAL SALARY for all department for month 12_2014.

4.4 RESULT:

Thus the nested Queries and join Queries were performed successfully and executed.

4.5 QUESTIONS AND ANSWERS:

Q1. What is a join?

Ans: join' used to connect two or more tables logically with or without common field.

Q2. What is outer join? Explain Left outer join, Right outer join and Full outer join.

Ans:OUTER JOIN: In An outer join, rows are returned even when there are no matches through the JOIN criteria on the second table.

Q3. What is Self Join and why is it required?

Ans: Self Join is the act of joining one table with itself.Self Join is often very useful to convert a hierarchical structure into a flat structure.

Q4. **What is Equi-join**

Ans:I t is a sql join where we use the equal sign as the comparison operator i.e "=" between two tables.By default this join is inner-equi-join.