# EXPERIMENT:2

## QUERIES (ALONG WITH SUB QUERIES) USING ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSECT

#### SOLUTION:

**To Create employee table:**

**Sql> create table employee( Fname varchar2(20), Lname varchar2(20),**

**Ssn number(4) primary key, B\_date date,**

**Address varchar2(30), Gender char(1), Salary number(7,2),**

**Super\_ssn references employee(ssn), Dno number(4)**

**);**

Table created.

#### SQL> INSERT INTO EMPLOYEE VALUES('SMITH',NULL,1111,'03-NOV- 2016','BJD','M',2000,NULL,10);

1 row created.

#### SQL> INSERT INTO EMPLOYEE VALUES('ALLEN',NULL,2222,'03-NOV- 2016','SBC','M',3000,1111,20);

1 row created.

#### SQL> INSERT INTO EMPLOYEE VALUES('MARTIN',NULL,3333,'03-NOV- 2016','HYD','M',4000,1111,30);

1 row created.

Like this we can insert the values into the table. To view data in the table following query is used

#### SQL> SELECT \*FROM EMPLOYEE;

FNAME LNAME SSN BDATE ADDRESS G SALARY SUPER\_SSN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SMITH | 1111 | 01-JAN-06 | BZA | M | 2000 | 10 |
| ALLEN | 2222 | 12-DEC-04 | SBC | M | 3000 | 1111 |
| MARTIN | 3333 | 15-DEC-07 | HYD | M | 4000 | 1111 |
| JONES | 4444 | 28-SEP-05 | TNU | M | 1500 | 2222 |
| BLAKE | 5555 | 04-SEP-04 | VZA | M | 2500 | 2222 |
| TURNER | 6666 | 21-OCT-99 | GNT | M | 6000 | 3333 |
| 6 rows selected. |  |  |  |  |  |  |

**To Create dependent table**:

**SQL> CREATE TABLE DEPENDENT (**

**ESSN NUMBER (4) REFERENCES EMPLOYEE (SSN), DEPENDENT\_NAME VARCHAR2 (20),**

**GENDER CHAR (1), B\_DATE DATE,**

**RELATIONSHIP VARCHAR2 (20),**

**PRIMARY KEY (ESSN, DEPENDENT\_NAME)**

**);**

Table created

Inserting values in the dependent table as follows

#### SQL> INSERT INTO DEPENDENT VALUES (1111,'SMITH',’G’);

1 row is created.

#### SQL> INSERT INTO DEPENDENT VALUES (2222,'POOJA',’F’);

1 row is created.

#### SQL> INSERT INTO DEPENDENT VALUES (3333,'MARTIN',’M’);

1 row is created.

#### SQL> INSERT INTO DEPENDENT VALUES (3333,'RAJA',’M’);

1 row is created.

To view data in the dependent table as follows.

#### SQL>SELECT \* FROM DEPENDENT;

**ESSN DEPENDENT\_NAME G B\_DATE RELATIONSHIP**

|  |  |  |
| --- | --- | --- |
| 1111 | SMITH | M |
| 2222 | POOJA | F |
| 3333 | MARTIN | M |
| 3333 | RAJA | M |

1. **ALL:** Retrieve the names of employees whose salary is greater than the salary of all the employees in department 10

#### SQL> SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY> ALL ( SELECT SALARY FROM EMPLOYEE WHERE DNO=10);

FNAME LNAME ALLEN

MARTIN TURNER

1. **ANY:** Retrieve the names of employees whose salary is greater than the salary of any one of the employees in department 10

#### SQL> SELECT FNAME, LNAME FROM EMPLOYEE WHERE SALARY> ANY( SELECT SALARYFROM EMPLOYEE WHERE DNO=10);

FNAME LNAME TURNER MARTIN

ALLEN BLAKE SMITH

1. **IN:** Retrieve the name of each employee who has a dependent with the firstname and same gender as theemployee

SQL> SELECT e.FNAME, e.LNAME FROM EMPLOYEE e WHERE e.SSN IN ( SELECT ESSN FROM DEPENDENT WHERE e.GENDER=GENDER AND e.FNAME = DEPENDENT\_NAME);

FNAME LNAME SMITH MARTIN

1. **EXISTS:** Retrieve the name of each employee who has a dependent with the firstname and same gender as the employee

##### SQL> SELECT e.FNAME, e.LNAME FROM EMPLOYEE e WHERE EXISTS (SELECT \*FROM DEPENDENT WHERE e.SSN=ESSN AND e.GENDER=GENDER AND e.FNAME = DEPENDENT\_NAME);

FNAME LNAME SMITH MARTIN

1. **NOT EXISTS:** Retrieve the names of employees who have no dependents

#### SQL> SELECT FNAME, LNAME FROM EMPLOYEE WHERE NOT EXISTS (SELECT \* FROM DEPENDENT WHERE SSN=ESSN);

FNAME LNAME ALLEN

### 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 be insert 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 Example:

##### SQL> create table person1 (id int, name varchar2 (10) not null, age int);

Table created.

* + - **UNIQUE** - Ensures that all values in a column are different Example:

##### SQL> create table person(id int unique, name varchar2(10),age int);

Table created.

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

Example:

##### SQL> create table emp1(id number(10) primary key, name varchar2(10),sal int);

Table created.

* + - **FOREIGN KEY** - Uniquely identifies a row/record in another table
      * A FOREIGN KEY is a key used to link two tables together.
      * A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
      * The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Example:

##### SQL> create table emp2 (eid int, city varchar2(10),foreign key(eid) references emp1(id));

Table created

* + - **CHECK** - Ensures that all values in a column satisfies a specific condition Example:

##### SQL> CREATE TABLE person1( ID int ,Age int, City varchar(10), CONSTRAINT chk CHECK(Age>=18 AND City='vja');

Table created.

* + - **DEFAULT** - Sets a default value for a column when no value is specified.
      * The DEFAULT constraint is used to provide a default value for a column.
      * The default value will be added to all new records IF no other value is specified. SQL DEFAULT on CREATE TABLE

##### SQL> create table emp(id number(10),name varchar2(10),city varchar2(10) default 'vja');

Table created.