**UNIT IV**

**STRUCTURED QUERY LANGUAGE (SQL)**

***Introduction***

A Database management system requires a query language to enable the user to access data. A structured query language (SQL) is a language used by the most of the relational database management systems.

The SQL language was developed by IBM in the mid 1970’s. In the 1979, the oracle corporation introduced the first commercial software SQL is implemented.

***Features of SQL***

The SQL is an English like language. It uses the words like CREATE,SELECT,UPDATE,DELETE etc., as part of its command set.

SQL is a nonprocedural language. We cannot execute more than one instruction at a time and also this will not provide by the programming concept.

***DATA TYPES***

The following table shows the different data types in DBMS.

|  |  |
| --- | --- |
| DATA TYPE | MAY CONTAIN |
| *Numeric:-*  NUMBER(P)  NUMBER(P,S)  *Character:-*  CHAR(N)  VARCHAR2(N)  LONG  *Date:-*  DATE | Allow to store digits of numbers  P means total number of digits  S means number of digits on the right side of the decimal point.  Fixed length character data, n characters long, Maximum size is 2000 bytes.  Variable length character data, Maximum size is 4000 bytes.  Allows to store 1GB to 4GB of characters.  Allows to store data in the form of date. The default date format is DD-MON-YY. |

***TABLE:*** *-* Table is a collection of related data composed of rows and columns.

(or)

Basic unit of storage composed rows and columns.

***QUERY:* -** Query is an expression, which is used to perform required operations on data.

***BASIC SQL OPERATIONS***

The basic SQL operations can be divided into three types.

**1. DATA DEFINITION LANGUAGE (DDL) COMMANDS.**

**2. DATA MANIPULATION LANGUAGE (DML) COMMANDS.**

**3. DATA CONTROL LANGUAGE (DCL) COMMANDS.**

1. DATA DEFINITION LANGUAGE (DDL) COMMANDS:-

The Data Definition Language (DDL) commands are used for to define the database. The DDL commands are CREATE,DESCRIBE,ALTER,RENAME,DROP, TRUNCATE. To create the table, to display the table structures, to change the table structure, to change the name of the table, to remove a table, to remove the rows from the table respectively.

2. DATA MANIPULATION LANGUAGE (DML) COMMANDS:-

The Data Manipulation Language commands are SELECT,INSERT,DELETE,UPDATE. To list the rows from the table, to add the one or more records to the table, to delete the rows, to change the values in a table.

3. DATA CONTROL LANGUAGE (DCL) COMMANDS:-

The Data Control Language commands are used for to control the database. The DCL commands are COMMIT, ROLLBACK, and SAVEPOINT.

***DDL COMMANDS***

***CREATE TABLE command****:-*

The general syntax of the CREATE TABLE command is

Syntax:-

CREATE TABLE tablename

(columnname1 datatype1, columnname2 datatype2,…………..);

Tablename--> The name of the table we want to create.

Columnname--> The name of the column.

Datatype-->Datatype of the column.

Example:-

AIM:-Write a query to create a table called STUDENT with the columns as given below.

COLUMN NAME TYPE SIZE

Studentnumber Numeric 4

Studentname Character 20

Studentaddress Alphanumeric 20

QUERY:***-***SQL>CREATE TABLE STUDENT

(SNO NUMBER(4),

SNAME VARCHAR2(20),

SADDRESS VARCHAR2(20));

OUTPUT: -Table created.

***DESCRIBE COMMAND***:-

This command is used for to display the structure of the table. The general syntax of the describe command is

Syntax:-

DESC[RIBE] tablename;

Example:-

AIM:-Write a query to display the structure of the table STUDENT.

QUERY: - SQL>DESCRIBE STUDENT;

OUTPUT:-

NAME NULL? TYPE

--------------------------------------------------------------------------------

SNO NUMBER(4)

SNAME VARCHAR2(20)

SADDRESS VARCHAR2(20)

***ALTER TABLE COMMAND***:-

Sometimes we have to change the table structure by adding the additional column to the existing table we can use the ALTER TABLE command with ADD option. The general syntax of the ALTER TABLE command with ADD option is

Syntax:-

ALTER TABLE tablename

ADD(Columnname1 datatype1, Columnname2 datatype2,…..);

Example:-

AIM:-Write a query to add the new column TOTAL of numeric of size 4 to the existing table STUDENT.

QUERY:-

SQL>ALTER TABLE STUDENT

ADD(TOTAL NUMBER(4));

OUTPUT:-

Table altered.

AIM:-Write a query to display the structure of the table EMPLOY.

QUERY: - SQL>DESC STUDENT;

OUTPUT:-

NAME NULL? TYPE

--------------------------------------------------------------------------------

SNO NUMBER(4)

SNAME VARCHAR2(20)

SADDRESS VARCHAR2(20)

TOTAL NUMBER(4)

To change the column length of the existing table. We can use the ALTER TABLE command with MODIFY option. The general syntax of the ALTER TABLE command with MODIFY option is

Syntax:-

ALTER TABLE tablename

MODIFY(Columnname datatype);

Example:-

AIM:-Write a query to change the SADDRESS column length of the existing table STUDENT. To increase the width of the SADDRESS column by 20.

QUERY:-

SQL>ALTER TABLE STUDENT

MODIFY(SADDRESS VARCHAR2(40));

OUTPUT:-

Table altered.

AIM:-Write a query to display the structure of the table EMPLOY.

QUERY:-

SQL>DESC STUDENT;

OUTPUT:-

NAME NULL? TYPE

--------------------------------------------------------------------------------

SNO NUMBER(4)

SNAME VARCHAR2(20)

SADDRESS VARCHAR2(40)

TOTAL NUMBER(4)

***RENAME COMMAND***:-

The RENAME command is used for to change the name of the table. The general syntax of the RENAME command is

Syntax:-

RENAME oldname TO newname;

Example:-

AIM:-Write a query to change the name of the table STUDENT to STU.

QUERY:-

SQL>RENAME STUDENT TO STU;

OPUTPUT:-

Table renamed.

***DROP TABLE COMMAND***:-

The DROP TABLE command is used for to remove a table from the database permanently. The general syntax of the DROP TABLE command is

Syntax:-

DROP TABLE tablename;

Example:-

AIM: - Write a query to remove a table STUDENT.

QUERY:-

SQL>DROP TABLE STUDENT;

OUTPUT:-

Table dropped.

***TRUNCATE TABLE COMMAND***:-

The TRUNCATE TABLE command is used for to remove all rows from a table. But this command, no rollback information is retained, that is releases the storage information. The general syntax of the TRUNCATE TABLE command is

Syntax:-

TRUNCATE TABLE tablename;

Example:-

AIM: - Write a query to remove all rows from the table STUDENT.

QUERY:-

SQL>TRUNCATE TABLE STUDENT;

OUTPUT:-

Table truncated.

***DML COMMANDS***

***INSERT COMMAND***:-

This command is used for to add the new record into the table. The general syntax of the INSERT command is

Syntax:-

INSERT INTO tablename

VALUES(Columnvalue1,columnvalue2,……….);

Example1:-

AIM1:-Write a query to add the new record into the table STUDENT.

QUERY:-

SQL>INSERT INTO STUDENT

VALUES(1,’srinivas’,’tenali’,900);

OUTPUT:-

1 row created.

Note: - The character fields are enclosed with in parenthesis and numeric fields are not.

Example2:-

AIM2:-Write a query to add another record into the table STUDENT.

QUERY:-

SQL>INSERT INTO STUDENT(SNAME,TOTAL,SADDRESS,SNO)

VALUES(‘krishna’,700,’guntur’,2);

OUTPUT:-

1 row created.

Note: - In the above example we can also change the order.

Example3:-

AIM2:-Write a query to add another record into the table STUDENT.

QUERY:-

SQL>INSERT INTO STUDENT(SNO,SNAME)

VALUES(3,’vasu’);

OUTPUT:-

1 row created.

Note: - The rest of the columns missed in the above statement will set to the value NULL (empty (or) no information).

Example4:-

AIM2:-Write a query to add another record into the table STUDENT.

QUERY:-

SQL>INSERT INTO STUDENT

VALUES(&SNO,’SNAME’,’&SADDRESS’,TOTAL);

OUTPUT:-

Enter value for SNO:4

Enter value for SNAME:raju

Enter value for SADDRESS:tenali

Enter value for TOTAL:700

OLD 2:VALUES(&SNO,’SNAME’,’&SADDRESS’,TOTAL);

NEW 2:VALUES(4,’raju’,’tenali’,700);

1 row created.

SQL>/

Note: - In the above example we can enter into the more than one record.

***SELECT COMMAND***:-

SELCET is a powerful statement in SQL. This allows to accessing the data from the database tables. The general syntax of the SELECT command is

Syntax:-

SELECT \* FROM tablename;

Example:-

AIM:-write a query to display the records from the table STUDENT.

QUERY:-

SQL>SELECT \* FROM STUDENT;

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

2 krishna guntur 700

3 vasu

4 raju Tenali 700

The another syntax of the SELECT command is

Syntax:-

SELECT columnname1[,columnname2,…..] FROM tablename;

Example:-

AIM:-write a query to display the student name, student address from the table STUDENT.

QUERY:-

SQL>SELECT SNAME,SADDRESS FROM STUDENT;

OUTPUT:-

SNAME SADDRESS

------------------------------------------

srinivas tenali

krishna guntur

vasu

raju Tenali

SELECT COMMAND USING THE LITERAL CHARACTER STRING

QUERY:-

SQL>SELECT SNAME||’belongs to’||SADDRESS FROM STUDENT;

OUTPUT:-

srinivas belongs to tenali

krishna belongs to guntur

vasu belongs to

raju belongs to Tenali

SELECT COMMAND USING THE ARITHMETIC OPERATORS

QUERY:-

SQL>SELECT TOTAL+100 FROM STUDENT;

OUTPUT:-

TOTAL

----------------

1000

800

800

ELIMINATE THE DUPLICATE ROWS BY USING THE KEYWORD DISTINCT (or) LISTING UNIQUE VALUES

QUERY:-

SQL>SELECT DISTINCT TOTAL FROM STUDENT;

OUTPUT:-

TOTAL

----------------

900

700

SELECT COMMAND USING THE WHERE CLAUSE (OR) SELECTING THE ROWS WITH CONDITIONAL RESTRICTION

Syntax:-

SELECT \*[Columnname1,columnname2,…..] FROM tablename

[WHERE condition];

AIM:-write a query to display the student name, total from the table STUDENT who are getting total 700.

QUERY:-

SQL>SELECT SNAME,TOTAL FROM STUDENT

WHERE TOTAL=700;

OUTPUT:-

SNAME TOTAL

-----------------------------------------

krishna 700

raju 700

TO LIST ALL THE DATABASE TABLES AVAILABLE IN THE CURRENT USER

QUERY:-

SQL>SELECT \* FROM TAB;

OUTPUT:-

TNAME TABTYPE

-----------------------------

STUDENT TABLE

EMP TABLE

DEPT TABLE

***DELETE COMMAND***:-

This command is used to delete the one or more rows from the specified table. The general syntax of the DELETE command is

Syntax:-

DELETE [FROM] tablename

[WHERE condition]:

Example:-

AIM: - Write a query to delete all the rows from the table STUDENT.

QUERY:-

SQL>DELETE STUDENT; (or) SQL>DELETE FROM STUDENT;

OUTPUT:-

4 rows deleted.

DELETE command using the WHERE clause

AIM: - Write a query to delete the rows from the table STUDENT who are belongs to tenali.

QUERY:-

SQL>DELETE STUDENT

WHERE SADDRESS=’tenali’;

OUTPUT:-

2 rows deleted.

***UPDATE COMMAND***:-

The UPDATE command is used for to change the values in a table. The general syntax of the UPDATE command is

Syntax:-

UPDATE tablename

SET columnname=value[expression]

[WHERE condition];

Example1:-

AIM: - Write a query to change the ADDRESS column by Tenali in all rows in the table STUDENT.

QUERY:-

SQL>UPDATE STUDENT

SET SADDRESS=’tenali’;

OUTPUT:-

4 rows updated.

Example2:-

AIM: - Write a query to change the ADDRESS column by guntur in the table STUDENT only the student name is krishna.

QUERY:-

SQL>UPDATE STUDENT

SET SADDRESS=’guntur’

WHERE SNAME=’krishna’;

OUTPUT:-

1 rows updated.

Example3:-

AIM: - Write a query to change the TOTAL column by adding 100 in the table STUDENT only the student name is srinivas.

QUERY:-

SQL>UPDATE STUDENT

SET TOTAL=TOTAL+100

WHERE SNAME=’srinivas’;

OUTPUT:-

1 rows updated.

***DCL COMMANDS***

***COMMIT command (or) SAVING TABLE CHANGES***:-

The command is used for to conform all the transactions permanently. The general syntax of the COMMIT command is

Syntax:-

COMMIT;

***ROLLBACK command (or) RESTORING TABLE CONTENTS***:-

This command cancel or revert all transactions since the last commit. The general syntax of the ROLLBACK command is

Syntax:-

ROLLBACK [TO SAVEPOINT];

***SAVEPOING command***:-

SAVEPOINT is a bookmark to control all transactions. The general syntax of the SAVEPOINT command is

Syntax:-

SAVEPOINT savepointname;

***ROLLBACK TO SAVEPOINT***:-

It allows to use transactions since the specified savepoint location.

Example:-

Create a table called CUSTOMER containing CUSTCODE,CUSTNAME,CUSTADDRESS as a fields and also enter into the three records.

CUSTOMER

CUSTCODE CUSTNAME CUSTADDRESS

------------------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

Now, we can use the COMMIT command

SQL>COMMIT;

O/P

Commit complete.

Now, to add the new record into the table CUSTOMER.

SQL>INSERT INTO CUSTOMER

VALUES(103,’D’,’TNL’);

O/P

1 row created.

Now, to add the another new record into the table CUSTOMER.

SQL>INSERT INTO CUSTOMER

VALUES(104,’E’,’GNT’);

O/P

1 row created.

Now, to display the records from the table CUSTOMER.

SQL>SELECT \* FROM CUSTOMER;

O/P

CUSTCODE CUSTNAME CUSTADDRESS

-----------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

103 D TNL

104 E GNT

Now, we can use the ROLLBACK command

SQL>ROLLBACK;

O/P

Rollback complete.

Now, to display the records from the table CUSTOMER.

SQL>SELECT \* FROM CUSTOMER;

CUSTCODE CUSTNAME CUSTADDRESS

------------------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

Now, to add the new record into the table CUSTOMER.

SQL>INSERT INTO CUSTOMER

VALUES(103,’D’,’TNL’);

O/P

1 row created.

Now, we can create a savepoint s1.

SQL>SAVEPOINT S1;

O/P

S1 savepoint created.

Now, to add another new record into the table CUSTOMER.

SQL>INSERT INTO CUSTOMER

VALUES(104,’E’,’GNT’);

O/P

1 row created.

Now, we can create a another savepoint s2.

SQL>SAVEPOINT S2;

O/P

S2 savepoint created.

Now, to add the another new record into the table CUSTOMER.

SQL>INSERT INTO CUSTOMER

VALUES(105,’F’,’VJA’);

O/P

1 row created.

Now, we can create a another savepoint s3.

SQL>SAVEPOINT S3;

O/P

S3 savepoint created.

Now, to display the records from the table CUSTOMER.

SQL>SELECT \* FROM CUSTOMER;

O/P

CUSTCODE CUSTNAME CUSTADDRESS

-----------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

Commit 🡪

103 D TNL

S1 savepoint 🡪

104 E GNT

S2 savepoint 🡪

105 F VJA

S3 savepoint 🡪

Now, we can use the complete syntax of the rollback command.

SQL>ROLLBACK TO S2;

O/P

Rollback complete.

Now, to display the records from the table CUSTOMER.

SQL>SELECT \* FROM CUSTOMER;

O/P

CUSTCODE CUSTNAME CUSTADDRESS

-----------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

103 D TNL

104 E GNT

Now, we can use the complete syntax of the rollback command.

SQL>ROLLBACK TO S1;

O/P

Rollback complete.

Now, to display the records from the table CUSTOMER.

SQL>SELECT \* FROM CUSTOMER;

O/P

CUSTCODE CUSTNAME CUSTADDRESS

-----------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

103 D TNL

BUFFER: - Buffer is a temporary memory area.

ED:-Load the current buffer text into editor. It allows to do modifications in a buffer text.

SLASH(/):- Re execute the last SQL statement.

SEMICOLON(;):-It display the last SQL statement.

***LAB SECTION 1***

***AIM1:-***

Write a query to create a table called EMPLOY with the columns as given below.

COLUMN NAME TYPE SIZE

Employ number Numeric 4

Employ name Character 10

Employ job Character 10

Hire date Date

Employ salary Numeric 7, 2

***QUERY:-***

SQL>CREATE TABLE EMPLOY

(EMPNO NUMBER(4),

ENAME VARCHAR2(10),

JOB VARCHAR2(10),

HIREDATE DATE,

SALARY NUMBER(7,2));

***OUTPUT:-***

Table created.

***AIM2:-***

Write a query to create a table called DEPART with the columns as given below.

COLUMN NAME TYPE SIZE

Department number Numeric 4

Department name Character 10

***QUERY:-***

SQL>CREATE TABLE DEPART

(DEPTNO NUMBER(4),

DNAME VARCHAR2(10));

***OUTPUT:-***

Table created.

***AIM3:-***

Write a query to display the structure of the table EMPLOY.

***QUERY:-***

SQL>DESC EMPLOY;

***OUTPUT:-***

NAME NULL? TYPE

--------------------------------------------------------------------------------

EMPNO NUMBER(4)

ENAME VARCHAR2(10)

JOB VARCHAR2(10)

HIREDATE DATE

SALARY NUMBER(7,2)

***AIM4:-***

Write a query to display the structure of the table DEPART.

***QUERY:-***

SQL>DESC DEPART;

***OUTPUT:-***

COLUMNNAME NULL? TYPE

--------------------------------------------------------------------------------

DEPTNO NUMBER(4)

DNAME VARCHAR2(10)

***AIM5:-***

Write a query to add the new column LOC of character type of size 10 to the table DEPART.

***QUERY:-***

SQL>ALTER TABLE DEPART

ADD(LOC VARCHAR2(10));

***OUTPUT:-***

Table altered.

***AIM6:-***

Write a query to increase the size of the column ENAME to 20 characters in the table EMPLOY.

***QUERY:-***

SQL>ALTER TABLE EMPLOY

MODIFY(ENAME VARCHAR2(20));

***OUTPUT:-***

Table altered.

***AIM7:-***

Write a query to change the name of the table EMPLOY to EMPL.

***QUERY:-***

SQL>RENAME EMPLOY TO EMPL;

***OUTPUT:-***

Table renamed.

***AIM8:-***

Write a query to remove a table DEPART.

***QUERY:-***

SQL>DROP TABLE DEPART;

***OUTPUT:-***

Table removed.

***AIM9:-***

Write a query to add the new record into the table EMPLOY.

***QUERY:-***

SQL>INSERT INTO EMPLOY

VALUES(1,’srinivas’,’manager’,’01-jan-01’,20000);

***OUTPUT:-***

1 row created.

(or)

SQL>INSERT INTO EMPLOY

VALUES(&EMPNO,’&ENAME’,’&JOB’,’HIREDATE’,&SALARY);

O/P

Enter value for EMPNO:2

Enter value for ENAME:Krishna

Enter value for JOB:clerk

Enter value for HIREDATE:02-FEB-02

Enter value for SALARY:10000

OLD 2: VALUES(&EMPNO,’&ENAME’,’&JOB’,’HIREDATE’,&SALARY);

NEW 2: VALUES(2,’krishna’,’clerk’,’02-feb-02’,10000);

1 row created.

SQL>\

***AIM10:-***

Write a query to add the new record into the table DEPART.

***QUERY:-***

SQL>INSERT INTO DEPART

VAUES(10,’accounting’,’newyork’);

***OUTPUT:-***

1 row created.

***LAB SECTION 2***

***AIM1:-***

Write a query to list all the records from the table EMPLOY.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

2 krishna clerk 02-feb-02 10000

***AIM2:-***

Write a query to display the rows from the table DEPART.

***QUERY:-***

SQL>SELECT \* FROM DEPART;

***OUTPUT:-***

DEPTNO DNAME LOC

----------------------------------------------

10 accounting newyork

***AIM3:-***

Write a query to display the employ name, employ job from the table EMPLOY.

***QUERY:-***

SQL>SELECT ENAME,JOB FROM EMPLOY;

***OUTPUT:-***

ENAME JOB

------------------------------

srinivas manager

krishna clerk

***AIM4:-***

Write a query to display all the distinct job types from the table EMPLOY.

***QUERY:-***

SQL>SELECT DISTINCT JOB FROM EMPLOY;

***OUTPUT:-***

JOB

-----------------

manager

clerk

***AIM5:-***

Write a query to display all the employee names, employee salary from the table EMPLOY whose job is manager.

***QUERY:-***

SQL>SELECT ENAME,SALARY FROM EMPLOY

WHERE JOB=’manager’;

***OUTPUT:-***

ENAME SALARY

---------------------------------

srinivas 20000

***AIM6:-***

Write a query to list all the database tables available in the current user.

***QUERY:-***

SQL>SELECT \* FROM TAB;

***OUTPUT:-***

TNAME TABTYPE

------------------------------------

EMPLOYE TABLE

DEPART TABLE

STUDENT TABLE

***AIM7:-***

Write a query to conform all the transactions permanently in the EMPLOY and DEPART tables.

***QUERY:-***

SQL>COMMIT;

***OUTPUT:-***

Commit complete.

***AIM8:-***

Write a query to delete all the records from the table DEPART.

***QUERY:-***

SQL>DELETE DEPART;

***OUTPUT:-***

1 row deleted.

***AIM9:-***

Write a query to delete the records from the table EMPLOY whose job is clerk.

***QUERY:-***

SQL>DELETE EMPLOY

WHERE JOB=’clerk’;

***OUTPUT:-***

1 row deleted.

***AIM10:-***

Write a query to undo the delete from the table EMPLOY and DEPART.

***QUERY:-***

SQL>ROLLBACK;

***OUTPUT:-***

Rollback complete.

***AIM11:-***

Write a query to modify the salary column in the table EMPLOY for each employ salary is 20000 rupees only.

***QUERY:-***

SQL>UPDATE EMPLOY

SET SALARY=20000;

***OUTPUT:-***

2 rows updated.

***AIM12:-***

Write a query to modify the salary column in the table EMPLOY to add the 2000 rupees for each employee.

***QUERY:-***

SQL>UPDATE EMPLOY

SET SALARY=SALARY+2000;

***OUTPUT:-***

2 rows updated.

***AIM13:-***

Write a query to modify the job column by manager in the table EMPLOY whose job is clerk.

***QUERY:-***

SQL>UPDATE EMPLOY

SET JOB=’manager’

WHERE JOB=’clerk’;

***OUTPUT:-***

1 row updated.

***ARITHMETIC OPERATORS***

The following table shows the arithmetic operators in DBMS.

|  |  |
| --- | --- |
| OPERATOR | MEANING |
| +  -  \*  / | addition  subtraction  multiplication  division |

***The Rules of precedence(priority):-***

1. Perform operations within the parenthesis.

2. Perform power operations.

3. Perform multiplication and division.

4. Perform addition and subtraction.

***RELATIONAL OPERATORS***

The following table shows the Relational operators in DBMS.

|  |  |
| --- | --- |
| OPERATOR | MEANING |
| <  >  <=  >=  =  != (or) <> | Less than  Greater than  Less than or equal to  Greater than or equal to  Equal to  Not equal to |

STUDENT

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

2 krishna guntur 700

3 vasu

4 raju Tenali 700

Example:-

AIM:-write a query to display the records from the table STUDENT who are getting total greater than 700.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE TOTAL > 700;

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

While listing the records from the table STUDENT it will check each record weather the record is satisfied the given condition or not. The satisfied record will be displayed otherwise not.

***LOGICAL OPERATORS***

The logical operators are AND, OR and NOT.

AND: - Join the two or more conditions and return result only when all conditions are true.

OR: - Join the two or more conditions and return result when any condition is true.

NOT: - negation.

Examples:

AIM1:-write a query to display the records from the table STUDENT who are belongs to Tenali with the name is srinivas.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE SADDRESS=’tenali’ AND SNAME=’srinivas’;

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

AIM2:-write a query to display the records from the table STUDENT who are belongs to either Tenali or guntur.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE SADDRESS=’tenal’ OR SADDRESS=’guntur’;

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

2 krishna guntur 700

4 raju Tenali 700

AIM3:-write a query to display the records from the table STUDENT who are belongs to either Tenali or Guntur and with the name is srinivas.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE SNAME=’srinivas’ AND

(SADDRESS=’tenali’ OR SADDRESS=’guntur’);

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

***SPECIAL OPERATORS***

The special operators include

1) The IN special operator.

2) The BETWEEN special operator (or) RANGES.

3) The LIKE special operator (or) WILD CARDS

4) The IS NULL special operator.

5) The EXISTS special operator.

1) ***The IN special operator***:-

The general syntax of the IN operator is

Columnname IN (column values list);

This operator returns true if the columnname is one of the values in column values list.

Examples:-

AIM1:-write a query to display the records from the table STUDENT who are belongs to either Tenali,guntrur,Vijayawada,hyderabad.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE SADDRESS IN(’tenali’,’guntur’,’vijayawada’,’hyderabad’);

OUTPUT:-

SNO SNAME SADDRESS TOTAL

------------------------------------------------------------------------

1 srinivas tenali 900

2 krishna guntur 700

4 raju Tenali 700

AIM2:-write a query to display the records from the table STUDENT who are not belongs to either Tenali,guntrur,Vijayawada,hyderabad.

QUERY:-

SQL>SELECT \* FROM STUDENT

WHERE SADDRESS NOT IN(’tenali’,’guntur’,’vijayawada’,’hyderabad’);

OUTPUT:-

No rows selected.

***2)*** ***The BETWEEN special operator (or) RANGES***:-

The general syntax of the BETWEEN operator is

Columnname BETWEEN columnvalue1 AND columnavalue2;

This operator returns true the Columnname is existed in the range of columnvalue1 and columnvalue2.

EMPLOYEE

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

1 VASU MANAGER 02-FEB-02 15000

2 SRINU MANAGER 01-JAN-01 20000

3 RAJU CLERK 03-MAR-03 12000

4 RAMU CLERK 04-APR-04 10000

Example:

AIM:-Write a query if you want to retrieve the information about the employees whose salary is existed in the range of 15000 to 20000.

QUERY:-

SQL>SELECT \* FROM EMPLOYEE

WHERE SALARY BETWEEN 15000 AND 20000;

OUT PUT:-

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

1 VASU MANAGER 02-FEB-02 15000

2 SRINU MANAGER 01-JAN-01 20000

***3) The LIKE operator (or) WILD CARDS***:-

The LIKE operator can be consisting of the following two special characters.

Percentage(%):- Substituted by the any expression (more than one character)

Underscore( \_ ):-Substituted by the single character.

Example:

AIM1:-Write a query if you want to retrieve the information about the employees whose names are starting with ‘RA’.

QUERY:-

SQL>SELECT \* FROM EMPLOYEE

WHERE ENAME LIKE ‘RA’;

OUT PUT:-

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

3 RAJU CLERK 03-MAR-03 12000

4 RAMU CLERK 04-APR-04 10000

Sometimes we have to retrieve the information about certain details may be begin with some characters and end with some characters.

AIM2:-Write a query if you want to retrieve the information about the employees whose names are starting with ‘RA’ and ending with ‘U’.

QUERY:-

SQL>SELECT \* FROM EMPLOYEE

WHERE ENAME LIKE ‘RA%U’;

OUT PUT:-

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

3 RAJU CLERK 03-MAR-03 12000

4 RAMU CLERK 04-APR-04 10000

AIM3:-Write a query if you want to retrieve the information about the employee names whose second character must be ‘R’.

QUERY:-

SQL>SELECT ENAME FROM EMPLOYEE

WHERE ENAME LIKE ‘\_R%’;

OUT PUT:-

ENAME

---------------

SRINU

***4) The IS NULL special operator:-***

The standard SQL allows the use of IS NULL is to check for a NULL attribute(column) values.

Example:

AIM1:-Write a query if you want to retrieve the information about the employees that do not have a hiredate is assigned(HIREDATE IS NULL).

QUERY:-

SQL>SELECT \* FROM EMPLOYEE

WHERE HIREDATE IS NULL;

OUT PUT:-

No rows selected.

***5) The EXISTS special operator:-***

This operator returns true a subquery returns at least one row otherwise it is false.

EMPS

EMPNO ENAME JOB SAL DEPTNO

------------------------------------------------------------------------------------------------------

1 SRINIVAS MANAGER 5000 10

2 JONES MANAGER 5000 20

3 VASU MANAGER 4000 30

4 RAJU CLERK 2000 10

5 RAMU CLERK 1000 20

6 SURESH CLERK 1000 30

7 FORD ANALYST 3000 10

8 BLAKE CLERK 2000 20

DEPT

DEPTNO DANME LOC

------------------------------------------------------------------

10 ACCOUNTING NEWYORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

Example:

QUERY:-

SQL>SELECT \* FROM EMPS

WHERE EXISTS(SELECT \* FROM DEPT);

OUTPUT:-

EMPNO ENAME JOB SAL DEPTNO

------------------------------------------------------------------------------------------------------

1 SRINIVAS MANAGER 5000 10

2 JONES MANAGER 5000 20

3 VASU MANAGER 4000 30

4 RAJU CLERK 2000 10

5 RAMU CLERK 1000 20

6 SURESH CLERK 1000 30

7 FORD ANALYST 3000 10

8 BLAKE CLERK 2000 20

Here, the subquery SELECT \* FROM DEPT returns at least one row. So, the main query SELECT \* FROM EMPS list all the rows.

***ORDER BY CLAUSE***

The order by clause is used for rows have displayed in sorting order. If used, the ORDER BY must be the last clause in the SELECT statement.

The general syntax of the ORDER BY clause is

ORDER BY columnname [ASC/DESC];

Here, columnname refers to the field.

ASC stands for ascending.

DESC stands for descending.

The default sorting order is ascending.

Example:

***AIM1:-***

Write a query to retrieve the information about the employees with respect to his salary in ascending order.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

ORDER BY SALARY ASC;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

4 RAMU CLERK 04-APR-04 10000

3 RAJU CLERK 03-MAR-03 12000

1 VASU MANAGER 02-FEB-02 15000

2 SRINU MANAGER 01-JAN-01 20000

***AIM2:-***

Write a query to retrieve the information about the employees with respect to his salary in descending order.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

ORDER BY SALARY DESC;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

---------------------------------------------------------------------------------

2 SRINU MANAGER 01-JAN-01 20000

1 VASU MANAGER 02-FEB-02 15000

3 RAJU CLERK 03-MAR-03 12000

4 RAMU CLERK 04-APR-04 10000

***LAB SECTION 3***

EMPLOY

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

2 krishna clerk 02-feb-02 10000

DEPT

DEPTNO DNAME LOC

----------------------------------------

10 accounting newyork

***AIM1:-***

Write a query to list all the records from the table EMPLOY whose salary is greater than the 15000.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

WHERE SALARY > 15000;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

***AIM2:-***

Write a query to list the employ names whose job is clerk and his salary 10000 from the table EMPLOY.

***QUERY:-***

SQL>SELECT ENAME FROM EMPLOY

WHERE JOB=’clerk’ AND

SALARY=10000;

***OUTPUT:-***

ENAME

----------------

krishna

***AIM3:-***

Write a query to list all the records from the table EMPLOY whose job is either clerk or manager.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

WHERE JOB=’manager’ OR

JOB=’clerk’;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

2 krishna clerk 02-feb-02 10000

***AIM4:-***

Write a query to list all the records from the table EMPLOY their employee number is 1, 2 and 4.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

WHERE EMPNO IN(1,2,4);

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

2 krishna clerk 02-feb-02 10000

***AIM5:-***

Write a query to find the employee names from the table EMPLOY whose salary is between 10000 to 20000.

***QUERY:-***

SQL>SELECT ENAME FROM EMPLOY

WHERE SALARY BETWEEN 10000 AND 20000;

***OUTPUT:-***

ENAME

------------

srinivas

krishna

***AIM6:-***

Write a query to find the employee names from the table EMPLOY which have “ni” or “sh” in them.

***QUERY:-***

SQL>SELECT ENAME FROM EMPLOY

WHERE ENAME LIKE ‘%ni%’ OR

ENAME LIKE ‘%sh%’;

***OUTPUT:-***

ENAME

------------

srinivas

krishna

***AIM7:-***

Write a query to list all the records from the table EMPLOY whose third character must be “i”.

***QUERY:-***

SQL>SELECT \* FROM EMPLOY

WHERE ENAME LIKE ‘\_ \_i%’;

***OUTPUT:-***

EMPNO ENAME JOB HIREDATE SALARY

--------------------------------------------------------------------------

1 srinivas manager 01-jan-01 20000

2 krishna clerk 02-feb-02 10000

***AIM8:-***

Write a query to list the employee name and their job title from the table EMPLOY who are not get a salary.

***QUERY:-***

SQL>SELECT ENAME,JOB FROM EMPLOY

WHERE SALARY IS NULL;

***OUTPUT:-***

No rows selected.

***AIM9:-***

Write a query to display department name is ascending order.

***QUERY:-***

SQL>SELECT DNAME FROM DEPT

ORDER BY DNAME ASC;

***OUTPUT:-***

DNAME

---------------

Accounting

***AIM10:-***

Write a query to display department name is descending order.

***QUERY:-***

SQL>SELECT DNAME FROM DEPT

ORDER BY DNAME DESC;

***OUTPUT:-***

DNAME

---------------

Accounting

***AIM11:-***

Write a query to create a table EMPDUP which consist of employ name, employ job, and employ salary of EMPLOY table.

***QUERY:-***

SQL>CREATE TABLE EMPDUP

AS SELECT ENAME,JOB,SALARY FROM EMPLOY;

***OUTPUT:-***

Table created.

***AIM12:-***

Write a query to determine ENAME, JOB, Return the title for both the columns as THE\_OUTPUT.

***QUERY:-***

SQL>SELECT (ENAME||’ ‘||JOB) THE\_OUTPUT FROM EMPLOYE;

***OUTPUT:-***

THE\_OUTPUT

-----------------------

srinivas manager

krishna clerk

***GROUP FUNCTIONS (or) AGGRIGATE FUNCTIONS***

TEST

EMPNO ENAME SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------

1 SRINIVAS 5000 10 10

2 JONES 5000 20

3 VASU 4000 10 30

4 RAJU 2000 10 10

5 RAMU 1000 10 20

6 SURESH 1000 10 30

7 FORD 3000 10

8 BLAKE 2000 10 20

COUNT():- Count is used for to count the number of rows in a table.

(or)

The number of rows containing the non-null values.

Example1:

***AIM1:-***

Write a query to count the number of rows in the relational table TEST.

***QUERY:-***

SQL>SELECT COUNT(\*) FROM TEST;

***OUTPUT:-***

COUNT(\*)

---------------

8

***AIM2:-***

Write a query retrieving the information about the commission in terms of how many persons drawing the commission in the table TEST.

***QUERY:-***

SQL>SELECT COUNT(COMM) FROM TEST;

***OUTPUT:-***

COUNT(COMM)

---------------------

6

MAX():- MAX is used for to find the highest value in the given column.

(or)

The maximum attribute(column) value encountered in a given column.

Example:

***AIM:-***

Write a query to find the highest salary of an employee in the table TEST.

***QUERY:-***

SQL>SELECT MAX(SAL) FROM TEST;

***OUTPUT:-***

MAX(SAL)

---------------

5000

MIN():- This function is used for to find the lowest value in the given column.

(or)

The minimum attribute(column) value encountered in a given column

Example:

***AIM:-***

Write a query to find the lowest salary of an employee in the table TEST.

***QUERY:-***

SQL>SELECT MIN(SAL) FROM TEST;

***OUTPUT:-***

MIN(SAL)

---------------

1000

SUM():- This function is used for to find the sum of all values in the given column.

Example:

***AIM:-***

Write a query to find the total salary of all the employees in the table TEST.

***QUERY:-***

SQL>SELECT SUM(SAL) FROM TEST;

***OUTPUT:-***

SUM(SAL)

---------------

23000

AVG():- This function is used for to find the average value of the given column.

(or)

The attribute(column) average(mean) value for a specified column.

Example:

***AIM:-***

Write a query to find the average salary of an employee in the table TEST.

***QUERY:-***

SQL>SELECT AVG(SAL) FROM TEST;

***OUTPUT:-***

AVG(SAL)

---------------

2875.7143

***GROUP BY CLAUSE***

The GROUP BY clause is used to arrange the data in a table to form of certain groups. The general syntax of the GROUP BY clause is

GROUP BY Columnname[,columnname2,……];

Example:

***AIM:-***

Write a query to calculate the average salary of the employees in each department in the table TEST.

Enter the several separate AVG(SAL) queries in each department.

***QUERY1:-***

SQL>SELECT AVG(SAL) FROM TEST

WHERE DEPTNO=10;

***OUTPUT:-***

AVG(SAL)

---------------

3333.33

***QUERY2:-***

SQL>SELECT AVG(SAL) FROM TEST

WHERE DEPTNO=20;

***OUTPUT:-***

AVG(SAL)

---------------

2666.66

***QUERY3:-***

SQL>SELECT AVG(SAL) FROM TEST

WHERE DEPTNO=30;

***OUTPUT:-***

AVG(SAL)

---------------

2500

But we can get the same result with a single query by using the GROUP BY DEPTNO Clause.

***QUERY1:-***

SQL>SELECT DEPTNO,AVG(SAL) FROM TEST

GROUP BY DEPTNO;

***OUTPUT:-***

DEPTNO AVG(SAL)

-------------------------------

10 3333.33

20 2666.66

30 2500

GROUP BY DEPTNO clause divides all the employees into groups based on their department number. The group function AVG(SAL) is then applied the rows in each group.

***HAVING CLAUSE***

The HAVING clause select the specific groups. The HAVING clause compares some property of group with a constant value, if a group satisfied the condition in the HAVING clause, it is included query result.

Example:

***AIM:-***

Write a query if you want to retrieve the department numbers contained more than two employees in the table TEST.

***QUERY:-***

SQL>SELECT DEPTNO,COUNT(DEPTNO) FROM TEST

GROUP BY DEPTNO

HAVING COUNT(DEPTNO) > 2;

***OUTPUT:-***

DEPTNO COUNT(DEPTNO)

-----------------------------------

10 3

20 3

***NVL() FUNCTION***

This function is used for evaluating the NULL values.

Example:

***AIM1:-***

Write a query to display the employ number and total salary of an each employ it is included commission in the table TEST.

***QUERY:-***

SQL>SELECT EMPNO,SAL+COMM FROM TEST;

***OUTPUT:-***

EMPNO SAL+COMM

-----------------------------------------------------

1 5010

2

3 4010

4 2010

5 1010

6 1010

7

8 2010

In the above output 2nd and 7th columns filled with the NULL values to avoid this one we can write the query as follows.

***QUERY:-***

SQL>SELECT EMPNO, SAL+NVL(COMM,0) FROM TEST;

***OUTPUT:-***

EMPNO SAL+NVL(COMM,0)

---------------------------------------------------------

1 5010

2 5000

3 4010

4 2010

5 1010

6 1010

7 3000

8 2010

***AIM2:-***

Write a query to calculate the total salary of all the employees for each department including commission in the table TEST.

***QUERY:-***

SQL>SELECT DEPTNO,SUM(SAL+NVL(COMM,0)) FROM TEST

GROUP BY DEPTNO;

***OUTPUT:-***

DEPTNO SUM(SAL+NVL(COMM,0) )

----------------------------------------------------

10 10020

20 8020

30 5030

***LAB SECTION 4***

EMP

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7369 smith clerk 7902 17-dec-80 800 20

7499 allen salesman 7698 20-feb-81 1600 300 30

7521 ward salesman 7698 22-feb-81 1250 500 30

7566 jones manager 7839 02-apr-81 2975 20

7654 martin salesman 7698 28-sep-81 1250 1400 30

7698 blake manager 7839 01-may-81 2850 30

7782 clark manager 7839 04-jun-81 2450 10

7788 scott analyst 7566 09-dec-82 3000 20

7839 king president 17-nov-81 5000 10

7844 turner salesman 7698 08-sep-81 1500 30

7876 adams clerk 7788 12-jan-83 1100 20

7900 james clerk 7698 03-dec-81 950 30

7902 ford analyst 7566 03-dec-81 3000 20

7934 miller clerk 7782 23-jan-82 1300 10

DEPT

DEPTNO DNAME LOC

--------------------------------------------------------------------------

10 ACCOUNTING NEWYORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

***AIM1:-***

Write a query to list all the records from the table EMP.

***QUERY:-***

SQL>SELECT \* FROM EMP;

***OUTPUT:-***

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7369 smith clerk 7902 17-dec-80 800 20

7499 allen salesman 7698 20-feb-81 1600 300 30

7521 ward salesman 7698 22-feb-81 1250 500 30

7566 jones manager 7839 02-apr-81 2975 20

7654 martin salesman 7698 28-sep-81 1250 1400 30

7698 blake manager 7839 01-may-81 2850 30

7782 clark manager 7839 04-jun-81 2450 10

7788 scott analyst 7566 09-dec-82 3000 20

7839 king president 17-nov-81 5000 10

7844 turner salesman 7698 08-sep-81 1500 30

7876 adams clerk 7788 12-jan-83 1100 20

7900 james clerk 7698 03-dec-81 950 30

7902 ford analyst 7566 03-dec-81 3000 20

7934 miller clerk 7782 23-jan-82 1300 10

***AIM2:-***

Write a query to list all the records from the table DEPT.

***QUERY:-***

SQL>SELECT \* FROM DEPT;

***OUTPUT:-***

DEPTNO DANME LOC

------------------------------------------------------------------

10 ACCOUNTING NEWYORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

***AIM3:-***

Write a query to find the highest salary and lowest salary of the employees in department number 10.

***QUERY:-***

SQL>SELECT MAX(SAL),MIN(SAL) FROM EMP

WHERE DEPTNO=10;

***OUTPUT:-***

MAX(SAL) MIN(SAL)

-------------------------------------

5000 1300

***AIM4:-***

Write a query to find the total salary of all the employees in department number 20.

***QUERY:-***

SQL>SELECT SUM(SAL) FROM EMP

WHERE DEPTNO=20;

***OUTPUT:-***

SUM(SAL)

-------------------------------------

10875

***AIM5:-***

Write a query to find the average salary of the employees in department number 30.

***QUERY:-***

SQL>SELECT AVG(SAL) FROM EMP

WHERE DEPTNO=30;

***OUTPUT:-***

AVG(SAL)

-----------------------------

8608.3333

***AIM6:-***

Write a query to count the number of employees in each department.

***QUERY:-***

SQL>SELECT DEPTNO,COUNT(\*) FROM EMP

GROUP BY DEPTNO;

***OUTPUT:-***

DEPTNO COUNT(\*)

---------------------------------------

10 3

20 5

30 6

***AIM7:-***

Write a query to find the highest salary, lowest salary, total salary, average salary of the employees in each department.

***QUERY:-***

SQL>SELECT MAX(SAL),MIN(SAL),SUM(SAL),AVG(SAL) FROM EMP

GROUP BY DEPTNO;

***OUTPUT:-***

DEPTNO MAX(SAL) MIN(SAL) SUM(SAL) AVG(SAL)

-----------------------------------------------------------------------------------

10 5000 1300 8750 2916.6666

20 3000 800 10875 2175

30 2850 950 9400 1566.6666

***AIM8:-***

Write a query to find the average salary of the employees in each department having more than three employees.

***QUERY:-***

SQL>SELECT DEPTNO,AVG(SAL) FROM EMP

GROUP BY DEPTNO

HAVING COUNT(\*) > 3;

***OUTPUT:-***

DEPTNO AVG(SAL)

-------------------------------------

20 2175

30 1566.6666

***AIM9:-***

Write a query to display the total salary and average salary of the employees for each job type within a department.

***QUERY:-***

SQL>SELECT DEPTNO,JOB,SUM(SAL) TOTAL\_SALARY,AVG(SAL) AVERAGE\_SALARY FROM EMP

GROUP BY JOB,DEPTNO;

***OUTPUT:-***

DEPTNO JOB TOTAL\_SALARY AVERAGE\_SALARY

--------------------------------------------------------------------------------------------

10 clerk 1300 1300

10 manager 2450 2450

10 president 5000 5000

20 anlyst 6000 3000

20 clerk 1900 950

20 manager 2975 2995

30 clerk 950 950

30 manager 2850 2850

30 salesman 5600 1400

***AIM10:-***

Write a query to print the employee name and their annual salary including commission.

***QUERY:-***

SQL>SELECT ENAME,(SAL+NVL(COMM,0))\*12 ANNUAL\_SAL FROM EMP;

***OUTPUT:-***

ENAME AUNUAL\_SAL

------------------------------------------

smith 9600

allen 22800

ward 21000

jones 35700

martin 31800

blake 34200

clark 29400

scott 36000

king 60000

turner 18000

adams 13200

james 11400

ford 36000

miller 15600

***AIM11:-***

Write a query to count the number of people with the same job.

***QUERY:-***

SQL>SELECT JOB,COUNT(JOB) FROM EMP

GROUP BY JOB;

***OUTPUT:-***

JOB COUNT(JOB)

------------------------------------

Manager 3

Clerk 4

Salesperson 4

Analyst 2

president 1

***AIM12:-***

Write a query to find difference between the highest salary and lowest salary of the employees, label the column as DIFFERENCE.

***QUERY:-***

SQL>SELECT (MAX(SAL) - MIN(SAL)) DIFFERENCE FROM EMP;

***OUTPUT:-***

DIFFERENCE

----------------------

4200

***JOINING METHODS (or) PROCESSING MULTIPLE TABLES***

***JOIN (or) CROSS JOIN (or) PRODUCT***:-

A join is used when a SQL query requires data from the more than one table.

Generally the tables are joined by using the following method.

All the rows in the first table with the first row in second table. All the rows in the first table with the second row in second table and so on.

PERSON VISITED

ID NAME ID PLACE

---------------------------------- ----------------------------------------------------

1 A 1 TNL

2 B 2 VJA

3 C 1 GNT

4 D 3 VJA

2 GNT

3 TNL

1 VJA

***AIM:-***

Write a query to join the two tables PERSON, VISITED.

***QUERY:-***

SQL>SELECT \* FROM PERSON,VISITED;

***OUTPUT:-***

ID NAME ID PLACE

-----------------------------------------------------------------

1 A 1 TNL

2 B 1 TNL

3 C 1 TNL

4 D 1 TNL

1 A 2 VJA

2 B 2 VJA

3 C 2 VJA

4 D 2 VJA

1 A 1 GNT

2 B 1 GNT

3 C 1 GNT

4 D 1 GNT

1 A 3 VJA

2 B 3 VJA

3 C 3 VJA

4 D 3 VJA

1 A 2 GNT

2 B 2 GNT

3 C 2 GNT

4 D 2 GNT

1 A 3 TNL

2 B 3 TNL

3 C 3 TNL

4 D 3 TNL

1 A 1 VJA

2 B 1 VJA

3 C 1 VJA

4 D 1 VJA

There are four joining methods

1. EQUI JOIN (or) INNER JOIN (or) NATURAL JOIN (or) SIMPLE JOIN

2. NON-EQUI JOIN

3. OUTER JOIN

4. SELF JOIN (or) RECURSIVE JOIN

***1. EQUI JOIN (or) INNER JOIN (or) NATURAL JOIN (or) SIMPLE JOIN: -*** This method join the tables by using the EQUAL TO operator.

Example:-

***AIM:-***

Write a query to list the persons along with their visited places.

***QUERY:-***

SQL>SELECT \* FROM PERSON,VISITED

WHERE PERSON.ID = VISITED.ID;

***OUTPUT:-***

ID NAME ID PLACE

-----------------------------------------------------------------

1 A 1 TNL

1 A 1 GNT

1 A 1 VJA

2 B 2 VJA

2 B 2 GNT

3 C 3 VJA

3 C 3 TNL

***2. NON-EQUI JOIN: -*** This method join the tables by using the other than EQUAL TO operator.

Example:-

***QUERY:-***

SQL>SELECT \* FROM PERSON,VISITED

WHERE PERSON.ID > VISITED.ID;

***OUTPUT:-***

ID NAME ID PLACE

-----------------------------------------------------------------

2 B 1 TNL

2 B 1 GNT

2 B 1 VJA

3 C 1 TNL

3 C 1 GNT

3 C 1 VJA

3 C 2 VJA

3 C 2 GNT

4 D 1 TNL

4 D 1 GNT

4 D 1 VJA

4 D 2 VJA

4 D 2 GNT

4 D 3 VJA

4 D 3 TNL

***3. OUTER JOIN: -*** In the outer join method the system will list all the rows from the first table whether they are not having combined rows in other table. The outer join method denoted by the operator ‘+’.

The operator plus (+) sign enclosed in parenthesis.

There are two outer join methods.

a) Left outer join.

b) Right outer join.

a)Left outer join:-

Example:-

***QUERY:-***

SQL>SELECT \* FROM PERSON,VISITED

WHERE PERSON.ID = VISITED.ID(+);

***OUTPUT:-***

ID NAME ID PLACE

-----------------------------------------------------------------

1 A 1 TNL

1 A 1 GNT

1 A 1 VJA

2 B 2 VJA

2 B 2 GNT

3 C 3 VJA

3 C 3 TNL

-----------------------------------------------------------------

4 D Outer information

-----------------------------------------------------------------

b)Right outer join:-

Example:-

***QUERY:-***

SQL>SELECT PERSON.ID,NAME,PLACE FROM PERSON,VISITED

WHERE PERSON.ID(+) = VISITED.ID;

***OUTPUT:-***

ID NAME PLACE

----------------------------------------------------

1 A TNL

1 A GNT

1 A VJA

2 B VJA

2 B GNT

3 C VJA

3 C TNL

***4. SELF JOIN (or) RECURSIVE JOIN: -*** In the self join method join a table to itself. This is called a self join or recursive join.

Example:-

***QUERY:-***

SQL>SELECT \* FROM PERSON,PERSON P;

***OUTPUT:-***

ID NAME ID NAME

------------------------------------------------------------------

1 A 1 A

2 B 1 A

3 C 1 A

4 D 1 A

1 A 2 B

2 B 2 B

3 C 2 B

4 D 2 B

1 A 3 C

2 B 3 C

3 C 3 C

4 D 3 C

1 A 4 D

2 B 4 D

3 C 4 D

4 D 4 D

Initially it creates a two copies of the tables named as PERSON and P and also called as aliases. Now these two tables works like a separate tables.

***RELATIONAL SET OPERATORS***

To mix the rows retrieved from the both of the queries.

PRODUCT PURCHASE

PRODUCTNAME ITEMNAME

--------------------- ------------------

Hard disk Hard disk

Floppy Floppy

Tape Ribbon

Ribbon CD drive

Printer

The relational set operators are

1. a)UNION

b)UNION ALL

2. INTERSECT

3. MINUS

1. a)UNION:- To return all the distinct rows retrieved by the both of the queries.

The general syntax is

QUERY1

UNION

QUERY2;

Example:-

SELECT \* FROM PRODUCT

UNION

SELECT \* FROM PURCHASE;

PRODUCTNAME

----------------------

Hard disk

Floppy

Tape

Ribbon

CDdrive

Printer

1. b)UNION ALL:- To return all the rows(including duplicates) retrieved by the both of the queries.

The general syntax is

QUERY1

UNION ALL

QUERY2;

Example:-

SELECT \* FROM PRODUCT

UNION ALL

SELECT \* FROM PURCHASE;

PRODUCTNAME

-----------------------

Hard disk

Floppy

Tape

Ribbon

Hard disk

Floppy

Ribbon

CD drive

printer

2) INTERSECT: - To return only common rows retrieved by the both of the queries.

The general syntax is

QUERY1

INTERSECT

QUERY2;

Example:-

SELECT \* FROM PRODUCT

INTERSECT

SELECT \* FROM PURCHASE;

PRODUCTNAME

----------------------

Hard disk

Floppy

Ribbon

3) MINUS: - To return all the rows retrieved by first query that are not in the second query.

The general syntax is

QUERY1

MINUS

QUERY2;

Example1:-

SELECT \* FROM PRODUCT

MINUS

SELECT \* FROM PURCHASE;

PRODUCTNAME

----------------------

Tape

Example2:-

SELECT \* FROM PURCHASE

MINUS

SELECT \* FROM PRODUCT;

ITEMNAME

----------------------

CDdrive

Printer

***CONSTRAINTS***

A constraint is a rule that controls the accessing of data from a table.

The constraints are

1. a)NOT NULL constraint

b) NULL constraint

2. CHECK constraint

3. PRIMARY KEY constraint

4. UNIQUE constraint

5. FOREIGN KEY constraint

6. DEFAULT constraint

***1. a)NOT NULL constraint:-*** This constraint is used to ensure that a column does not accept nulls.

(or)

This constraint controls the accessing of null values into the column.

Example:-

SQL>CREATE TABLE SAMPLE1

(ID NUMBER(3) NOT NULL,

NAME VARCHAR2(20));

It creates a table with the name sample1 and define the column ID with the constraint NOT NULL.

i) SQL>INSERT INTO SAMPLE1

VALUES(100,’A’);

This is a valid transaction.

ii) SQL>INSERT INTO SAMPLE1

VALUES(NULL,’B’);

This is invalid transaction. It violates the NOT NULL constraint because the value for the ID column value is NULL.

iii) SQL>INSERT INTO SAMPLE1(NAME)

VALUES(‘C’);

This is invalid transaction. It violates the NOT NULL constraint because the value for the ID column value is NULL.

***1.b)NULL constraint:-*** This constraint is default constraint for every column in a table structure. A NULL constraint allows a null values into the column.

***2) CHECK constraint: -*** This constraint to allow only the valid data that satisfies the given condition. The genera syntax is

CHECK(condition)

Example:-

SQL>CREATE TABLE SAMPLE2

(ID NUMBER(3) CHECK(ID BETWEEN 200 AND 400),

NAME VARCHAR2(20));

It creates a table with the name sample2 and define the column ID with the constraint CHECK.

i) SQL>INSERT INTO SAMPLE2

VALUES(250,’A’);

This is a valid transaction because it satisfies the check constraint.

ii) SQL>INSERT INTO SAMPLE2

VALUES(150,’B’);

This is invalid transaction. It violates the check constraint because the value for the ID column value 150 will not satisfied the check condition.

iii) SQL>UPDATE SAMPLE2

SET ID=180

WHERE ID=250;

This is invalid transaction. It violates the check constraint because the value for the ID column value 180 will not satisfied the check condition.

***3. PRIMARY KEY constraint: -*** This constraint control the accessing of duplicate values and null values into the column.

Example:-

SQL>CREATE TABLE SAMPLE3

(ID NUMBER(3) PRIMARY KEY,

NAME VARCHAR2(20));

It creates a table with the name sample3 and define the column ID with the constraint PRIMARY KEY.

i) SQL>INSERT INTO SAMPLE3

VALUES(100,’A’);

This is a valid transaction.

ii) SQL>INSERT INTO SAMPLE3

VALUES(100,’B’);

This is invalid transaction. It violates the primary key constraint because the value for the ID column value 100 is duplicate.

iii) SQL>INSERT INTO SAMPLE3

VALUES(NULL‘C’);

This is invalid transaction. It violates the primary key constraint because the value for the ID column value is NULL.

If you want to define the both the columns as primary key. Use the primary key constraint as below

SQL>CREATE TABLE SAMPLE4

(ID NUMBER(3) ,

DID NUMBER(3),

NAME VARCHAR2(20),PRIMARY KEY(ID,DID));

It creates a table with the name sample4 and define the columns ID and DID with the constraint PRIMARY KEY.

i) SQL>INSERT INTO SAMPLE4

VALUES(100,200,’A’);

This is a valid transaction.

ii) SQL>INSERT INTO SAMPLE4

VALUES(100,201,’B’);

This is valid transaction because the system will check the duplicates while the combination of the ID and DID(ID is same but DID is different).

iii) SQL>INSERT INTO SAMPLE4

VALUES(100,200,‘C’);

This is invalid transaction because the system will check the duplicates while the combination of the ID and DID(ID and DID both column values are same).

***4. UNIQUE constraint: -*** This constraint similar to primary key constraint. But this constraint allows a NULL values into the column.

Example:-

SQL>CREATE TABLE SAMPLE5

(ID NUMBER(3) UNIQUE,

NAME VARCHAR2(20));

It creates a table with the name sample5 and define the column ID with the constraint UNIQUE.

i) SQL>INSERT INTO SAMPLE5

VALUES(100,’A’);

This is a valid transaction.

ii) SQL>INSERT INTO SAMPLE5

VALUES(100,’B’);

This is invalid transaction. It violates the UNIQUE constraint because the value for the ID column value 100 is duplicate.

iii) SQL>INSERT INTO SAMPLE5

VALUES(NULL‘C’);

This is valid transaction because this constraint allows null values into the column.

***5. FOREIGN KEY constraint: -*** This constraint references to the other table whether the given value into the column existed in the reference table.

The general syntax is

FOREIGN KEY(columnname)

REFERENCES tablename(columnname)

[ON DELETE CASCADE]

Tablename -> the name of the table we want to use for reference to check whether a given column

Value is existed in this table.

Columnname -> the name of the column in the reference table and must be defined with the constraint

PRIMARY KEY.

ON DELETE CASCADE -> this clause delete all rows in the child table when you delete a corresponding

row in the master table.

Example:-

CUSTOMER

PRIMARY KEY --------> CUSTCODE CUSTNAME CUSTADDR

---------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

103 D HYD

SQL>CREATE TABLE BILLS

(BILLNO NUMBER(3),

BILLDATE DATE,

CUSTCODE NUMBER(3),FOREIGN KEY(CUSTCODE)

REFERENCES CUSTOMER(CUSTCODE));

OUTPUT:-

Table created.

i) SQL>INSERT INTO BILLS

VALUES(400,’10-NOV-22’,100);

Before going to insert this record into the table BILLS its verifies the CUSTOMER table for the existence of the CUSTCODE value(100) in the given list of values(100,101,102,103). If a corresponding record is located in the reference table CUSTOMER then the new record is added to the table BILLS, otherwise it gives an error message.

ii) SQL>UPDATE BILLS

SET CUSTCODE=104

WHERE CUSTCODE=100;

This is invalid transaction. It violates the foreign key constraint because the CUSTCODE value 104 is not found in the CUSTOME table.

iii) SQL>DELETE CUSTOMER

WHERE CUSTCODE=100;

The record is not deleted from the table CUSTOMER because the corresponding record is referenced by other table BILLS.

But if you want to delete this record from the table CUSTOMER and also delete the corresponding record in the child table called BILLS. To create ON DELETE CASCADE clause in the table BILLS.

SQL>DROP TABLE BILLS;

OUTPUT:-

Table dropped.

SQL>CREATE TABLE BILLS

(BILLNO NUMBER(3),

BILLDATE DATE,

CUSTCODE NUMBER(3),FOREIGN KEY(CUSTCODE)

REFERENCES CUSTOMER(CUSTCODE)

ON DELETE CASCADE);

OUTPUT:-

Table created.

i) SQL>INSERT INTO BILLS

VALUES(400,’10-NOV-22’,100);

OUTPUT:-

1 row created.

ii) SQL>DELETE CUSTOMER

WHERE CUSTCODE=100;

Now delete the record from the table CUSTOMER whose CUSTCODE value is 100 and also delete the corresponding record in the child table called BILLS.

NOTE: - We can use the foreign key constraint on more than one column.

***6. DEFAULT constraint: -*** This constraint sets a default value to the column when the column value is null.

Example:-

SQL>CREATE TABLE SAMPLE

(ID NUMBER(3) DEFAULT 0,

NAME VARCHAR2(20)

BDATE DATE DEFAULT SYSDATE);

i) SQL>INSERT INTO SAMPLE(ID,NAME)

VALUES(100,’A’);

In this transaction the first two columns are set to the values 100 and A. The last column BDATE is set by default value SYSDATE(current system date).

***LAB SECTION 5***

EMP

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7369 smith clerk 7902 17-dec-80 800 20

7499 allen salesman 7698 20-feb-81 1600 300 30

7521 ward salesman 7698 22-feb-81 1250 500 30

7566 jones manager 7839 02-apr-81 2975 20

7654 martin salesman 7698 28-sep-81 1250 1400 30

7698 blake manager 7839 01-may-81 2850 30

7782 clark manager 7839 04-jun-81 2450 10

7788 scott analyst 7566 09-dec-82 3000 20

7839 king president 17-nov-81 5000 10

7844 turner salesman 7698 08-sep-81 1500 30

7876 adams clerk 7788 12-jan-83 1100 20

7900 james clerk 7698 03-dec-81 950 30

7902 ford analyst 7566 03-dec-81 3000 20

7934 miller clerk 7782 23-jan-82 1300 10

DEPT

DEPTNO DNAME LOC

--------------------------------------------------------------------------

10 ACCOUNTING NEWYORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

***AIM1:-***

Write a query to display employ name and their department name.

***QUERY:-***

SQL>SELECT EMP.ENAME,DEPT.DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO;

***OUTPUT:-***

ENAME DNAME

---------------------------------------------

smith research

allen sales

ward sales

jones research

martin sales

blake sales

clark accounting

scott research

king accounting

turner sales

adams research

james sales

ford research

miller accounting

***AIM2:-***

Write a query to display employ name, department number and department name.

***QUERY:-***

SQL>SELECT ENAME,EMP.DEPTNO,DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO;

(or)

SQL>SELECT ENAME,E.DEPTNO,DNAME FROM EMP E,DEPT D

WHERE E.DEPTNO=D.DEPTNO;

***OUTPUT:-***

ENAME DEPTNO DNAME

----------------------------------------------------------

smith 20 research

allen 30 sales

ward 30 sales

jones 20 research

martin 30 sales

blake 30 sales

clark 10 accounting

scott 20 research

king 10 accounting

turner 30 sales

adams 20 research

james 30 sales

ford 20 research

miller 10 accounting

***AIM3:-***

Write a query to display the employ name, location and department name of employees whose salary is greater than 1500.

***QUERY:-***

SQL>SELECT ENAME,LOC,DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO AND SAL>1500;

***OUTPUT:-***

ENAME LOC DNAME

----------------------------------------------------------------

clark newyork accounting

king newyork accounting

jones dallas research

ford dallas research

scott dallas research

allen chicago sales

blake chiago sales

***AIM4:-***

Write a query to display the employ name, job, department number and department name of all employees who work in newyork.

***QUERY:-***

SQL>SELECT ENAME,JOB,EMP.DEPTNO,DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO AND LOC=’NEWYORK’;

***OUTPUT:-***

ENAME JOB DEPTNO DNAME

---------------------------------------------------------------------------

clark manager 10 accounting

king president 10 accounting

miller clerk 10 accounting

***AIM5:-***

Write a query to display the department number and department name that department has no employees(using joining methods).

***QUERY:-***

SQL>SELECT DEPT.DEPTNO,DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO(+)=DEPT.DEPTNO AND EMPNO IS NULL;

***OUTPUT:-***

DEPTNO DNAME

------------------------------------------

40 operations

***AIM6:-***

Write a query to display the department number and department name that department has no employees(using relational set operators).

***QUERY:-***

SQL>SELECT DEPTNO,DNAME FROM DEPT

MINUS

SELECT EMP.DEPTNO,DNAME FROM EMP,DEPT

WHERE EMP.DEPTNO=DEPT.DEPTNO;

***OUTPUT:-***

DEPTNO DNAME

------------------------------------------

40 operations

***AIM7:-***

Find the job that was filled in the first half of 1983 and the same job that was filled during the same period in 1984.

***QUERY:-***

SQL>SELECT JOB FROM EMP

WHERE HIREDATE BETWEEN ’01-JAN-83’ AND ’30-JUN-83’

INTERSECT

SELECT JOB FROM EMP

WHERE HIREDATE BETWEEN ’01-JAN-84’ AND ’30-JUN-84’

***OUTPUT:-***

JOB

-------------

clerk

***AIM8:-***

Find all the distinct job types in department number 10 and all the distinct job types in department number 20.

***QUERY:-***

SQL>SELECT JOB FROM EMP

WHERE DEPTNO=10

UNION

SELECT JOB FROM EMP

WHERE DEPTNO=20;

***OUTPUT:-***

JOB

-------------

anlyst

clerk

manager

president

***AIM9:-***

Create a table called PROJECT, with the columns as given below. In addition, define its PROJECT-ID column as primary key, and ensure that PROJECT-END-DATE dates are not earlier than PROJECT-START-DATE dates.

Column name Data type size

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_

PROJECT\_ID NUMBER 4

PROJECT\_NAME VARCHAR2 20

PROJECT\_START\_DATE DATE

PROJECT\_END\_DATE DATE

BUDGET\_AMOUNT NUMBER 8

MAX\_NO\_STAFF NUMBER 2

***QUERY:-***

SQL>CREATE TABLE PROJECT

(PROJECT\_ID NUMBER(4) PRIMARY KEY,

PROJECT\_NAME VARCHAR2(20),

PROJECT\_START\_DATE DATE,

PROJECT\_END\_DATE DATE,

BUDGET\_AMOUNT NIMBER(8),

MAX\_NO\_STAFF NUMBER(2),

CHECK(PROJECT\_START\_DATE <= PROJECT\_END\_DATE));

***OUTPUT:-***

Table created.

***AIM10:-***

Create a table called ASSIGNMENTS, with the columns as given below. In addition, define its PROJECT-ID column as foreign key which references to the table PORJECT and EMPNO column is a further foreign key which references to the table EMP. These two columns should not allow NULL values(PROJECT\_ID and EMPNO).

Column name Data type size

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_

PROJECT\_ID NUMBER 4

EMPNO NUMBER 4

ASSIGNMENT\_START\_DATE DATE

ASSIGNMENT\_END\_DATE DATE

***QUERY:-***

SQL>CREATE TABLE ASSIGNMENTS

(PROJECT\_ID NUMBER(4) NOT NULL, FOREIGN KEY(PROJECT\_ID) REFERENCES PROJECT(PROJECT\_ID),

EMPNO NUMBER(4) NOT NULL,FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO),

ASSIGNMENT\_START\_DATE DATE,

ASSIGNMENT\_END\_DATE DATE);

***OUTPUT:-***

Table created.

***VIEWS (VIRTUAL TABLES)***

View is a logical table that holds the information related to a particular query. The view does not require a storage area. A view is a virtual table. It has no data of its own.

***CREATING A VIEW:-***

The general syntax is

CREATE [OR REPLACE] VIEW viewname

AS QUERY

[WITH CHECK OPTION];

Example:-

CUSTOMER

CUSTCODE CUSTNAME CUSTADDRESS

------------------------------------------------------------------------

100 A TNL

101 B GNT

102 C VJA

***AIM:*** - To create a view which consist of CUSTCODE,CUSTNAME,CUSTADDRESS of CUSTOMER table who are belongs to Guntur.

***QUERY:-***

SQL>CREATE VIEW VIEW1

AS SELECT \* FROM CUSTOMER

WHERE CUSTADDRESS=’GNT’;

***OUTPUT:-***

View created.

It creates a simple view with the name VIEW1. In this view we can use any type of database transactions like SELECT, INSERT.

Ex:

SQL>SELECT \* FROM VIEW1;

O/P

CUSTCODE CUSTNAME CUSTADDRESS

-----------------------------------------------------------------

101 B GNT

In this example the query defined under the view is executed.

Ex:

SQL>INSERT INTO VIEW1

VALUES(103,’D’,’GNT’);

O/P:-

1 row created.

It adds the new record on the base table of the VIEW1. The base table of the VIEW1 is CUSTOMER.

Ex:

SQL>INSERT INTO VIEW1

VALUES(104,’E’,’VJA’);

O/P:-

1 row created.

This transaction will be successfully executed because the view does not have WITH CHECK OPTION.

***ALTERING VIEWS:-*** To alter the view definition use ‘OR REPLACE’ command that allows to redefine the view definition.

Example:-

**AIM:-**Write a query to replace the existing view VIEW1 with the clause WITH CHECK OPTION.

***QUERY:-***

SQL>CREATE OR REPLACE VIEW VIEW1

AS SELECT \* FROM CUSTOMER

WHERE CUSTADDRESS=’GNT’

WITH CHICK OPTION;

***OUTPUT:-***

View replaced.

SQL>INSERT INTO VIEW1

VALUES(105,’F’,’VJA’);

This transaction is failed because whose CUSTADDRESS value ‘VJA’ will not satisfied the condition in the query under defined the view.

***DROPPING VIEWS:-*** This command to remove a view from the database permanently.

The general syntax is

DROP VIEW viewname;

Example:-

**AIM:-** Write a query to remove a view VIEW1.

**QUERY:-**

SQL>DROP VIEW VIEW1;

**OUTPUT:-**

View dropped.

***LISTING THE VIEWS FROM THE CURRENT USER:-***

SQL>SELECT TNAME FROM TAB

WHERE TABTYPE=’VIEW’;

O/P

TNAME

----------

VIEW1

(or)

SQL>SELECT VIEW\_NAME FROM USER\_VIEWS;

O/P

VIEW\_NAME

-----------------

VIEW1

USER\_VIEWS🡪 It is a view defined by the ORACLE that maintain all the views created in the current user.

***SUBQUERIES***

Subquery means a query defined under another query. The main activity is that first subquery is executed and send the output of subquery as an input to the main query.

***WHEN TO USE SUBQUERIES (or) WHERE SUBQUERIES:-***

EMPS

EMPNO NAME JOB SAL DEPTNO

------------------------------------------------------------------------------------------------------

1 srinivas manager 5000 10

2 jones manager 5000 20

3 vasu manager 4000 30

4 raju clerk 2000 10

5 ramu clerk 1000 20

6 suresh clerk 1000 30

7 ford analyst 3000 10

8 blake clerk 2000 20

Suppose we want to find all the employees who have the same job as JONES. You might do this with two queries, one to find JONE’S job and another to find all employees with the same job.

To find jones job, enter

SQL> SELECT JOB FROM EMPS

WHERE ENAME= ‘jones’;

OUTPUT:-

JOB

---------

manager

Now, you know that jones job is manager, we can make a second query to find all the managers.

To find all the managers, enter

SQL>SELECT ENAME,JOB FROM EMPS

WHERE JOB=’manager’;

OUTPUT:-

ENAME JOB

-------------------------------------------

Jones manager

Srinivas manager

Vasu manager

We can reach the same result with one query by using a subquery in the WHERE clause to find the jones job. Always enclose subqueries in parentheses.

SQL>SELECT ENAME, JOB FROM EMPS (single row subquery)

WHERE JOB= (SELECT JOB FROM EMPS

WHERE ENAME=’jones’);

OUTPUT:-

ENAME JOB

-------------------------------------------

Jones manager

Srinivas manager

Vasu manager

The subquery return a single value. (In this case manager)

***SUBQUERIES THAT RETURN MORE THAN ONE ROW (or) MULTIPLE ROW SUBQUERY (or) IN SUBQUERIES:-***

If a subquery can return more than one value. We need to use a multi-row comparison operator. In the case the IN operator must be used because it expects a list of values.

Example1:-

AIM:-

Write a query to display employees in department number 10 with the same jobs as anyone in department number 30.

QUERY:-

SQL>SELECT ENAME,JOB,SAL FROM EMPS

WHERE DEPTNO=10

AND JOB IN(SELECT JOB FROM EMPS

WHERE DEPTNO=30);

OUTPUT:-

ENAME JOB SAL

-------------------------------------------------------

Srinivas manager 5000

Raju clerk 2000

The subquery returns a list of jobs hold by employees in department number 30. The main query selects employee in department number 10 and whose job returned by the subquery.

Example2:-

AIM:-

Write a query to display employees in department number 10 with a job not found in department number 30.

QUERY:-

SQL>SELECT ENAME,JOB,SAL FROM EMPS

WHERE DEPTNO=10

AND JOB NOT IN(SELECT JOB FROM EMPS

WHERE DEPTNO=30);

OUTPUT:-

ENAME JOB SAL

-------------------------------------------------------

Ford analyst 3000

***CORRELATED SUBQUERIES***

In a correlated subquery which processing the inner query depends on data from the outer query.

Example:-

AIM:-

Write a query to find the employees who earn a salary greater than the average salary for their department.

QUERY:-

SQL>SELECT EMPNO,ENAME,SAL,DEPTNO FROM EMPS E

WHERE SAL > (SELECT AVG(SAL) FROM EMPS

WHERE DEPTNO=E.DEPTNO);

OUTPUT:-

EMPNO ENAME SAL DEPTNO

---------------------------------------------------------------------------------

1 srinivas 5000 10

2 jones 5000 20

3 vasu 4000 30

Let us analyses the above example by using the EMPS table.

1. Select first candidate row srinivas in department number 10 earning 5000

2. EMPS FROM clause has alias E which qualifies DEPTNO(10) column reference in inner query’s WHERE clause.

3. WHERE clause compares 5000 against value returned by inner query.

***LAB SECTION 6***

EMP

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7369 smith clerk 7902 17-dec-80 800 20

7499 allen salesman 7698 20-feb-81 1600 300 30

7521 ward salesman 7698 22-feb-81 1250 500 30

7566 jones manager 7839 02-apr-81 2975 20

7654 martin salesman 7698 28-sep-81 1250 1400 30

7698 blake manager 7839 01-may-81 2850 30

7782 clark manager 7839 04-jun-81 2450 10

7788 scott analyst 7566 09-dec-82 3000 20

7839 king president 17-nov-81 5000 10

7844 turner salesman 7698 08-sep-81 1500 30

7876 adams clerk 7788 12-jan-83 1100 20

7900 james clerk 7698 03-dec-81 950 30

7902 ford analyst 7566 03-dec-81 3000 20

7934 miller clerk 7782 23-jan-82 1300 10

DEPT

DEPTNO DNAME LOC

--------------------------------------------------------------------------

10 ACCOUNTING NEWYORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

***AIM1:-***

Write a query to find the employees who earn the minimum salary.

***QUERY:-***

SQL>SELECT \* FROM EMP

WHERE SAL=(SELECT MIN(SAL) FROM EMP);

***OUTPUT:-***

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7369 smith clerk 7902 17-dec-80 800 20

***AIM2:-***

Write a query to find all the employ names whose department not found in DEPT table.

***QUERY:-***

SQL>SELECT ENAME FROM EMP

WHERE DEPTNO NOT IN(SELECT DEPTNO FROM DEPT);

***OUTPUT:-***

No rows selected.

***AIM3:-***

Write a query to display all the employ names who earn more than the lowest salary in department number 20.

***QUERY:-***

SQL>SELECT ENAME FROM EMP

WHERE SAL > (SELECT MIN(SAL) FROM EMP

WHERE DEPTNO=20);

***OUTPUT:-***

ENAME

-----------------

**allen**

**ward**

jones

martin

blake

clark

scott

king

turner

adams

james

ford

miller

***AIM4:-***

Write a query to find the employees who earn the maximum salary.

***QUERY:-***

SQL>SELECT \* FROM EMP

WHERE SAL=(SELECT MIN(SAL) FROM EMP);

***OUTPUT:-***

ENPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

------------------------------------------------------------------------------------------------------------------------------------

7839 king president 17-nov-81 5000 10

***AIM5:-***

Write a query to display the employ name, job, salary for employees whose salary is greater than the highest salary in sales department.(SUBQUERY WITH MULTIPLE TABLES)

***QUERY:-***

SQL>SELECT ENAME,JOB,SALRY FROM EMP

WHERE SAL > (SELECT MAX(SAL) FROM EMP

WHERE DEPTNO=(SELECT DEPTNO FROM DEPT

WHERE DEPTNO=’SALES’));

***OUTPUT:-***

ENAME JOB SAL

-------------------------------------------------------

Jones manager 2975

Scott analyst 3000

King president 5000

Ford analyst 3000

***AIM6:-***

Write a query to create a view which contains employ name, job and annual salary of all managers.

***QUERY:-***

SQL>CREATE VIEW MANAGEVIEW

AS SELECT ENAME,JOB,SAL\*12 ANNUAL\_SAL FROM EMP

WHERE JOB=’manager’;

***OUTPUT:-***

View created.

***AIM7:-***

Write a query to create a view which contains employ name, job and department number of all clerks.

***QUERY:-***

SQL>CREATE VIEW CLERKVIEW

AS SELECT ENAME,JOB,DEPTNO FROM EMP

WHERE JOB=’clerk’;

***OUTPUT:-***

View created.

***AIM8:-***

Write a query to display the views names in the current user.

***QUERY:-***

SQL>SELECT VIEW\_NAME FROM USER\_VIEWS;

***OUTPUT:-***

VIEW\_NAME

------------------

VIEW1

MANAGEVIEW

CLERKVIEW

***AIM9:-***

Write a query to display the department number, employ name and their salary of all the employees who earn the lowest salary in each department.

***QUERY:-***

(USING IN OPERATOR COMPARE WITH MORE THAN ONE COLUMN)

SQL>SELECT DEPTNO,ENAME,SAL FROM EMP

WHERE (DEPTNO,SAL) IN (SELECT DEPTNO,MIN(SAL) FROM EMP

GROUP BY DEPTNO);

(or)

SQL>SELECT DEPTNO,ENAME,SAL FROM EMP E (USING CORRELATED SUBQUERY)

WHERE SAL = (SELECT MIN(SAL) FROM EMP

WHERE DEPTNO = E.DEPTNO);

***OUTPUT:-***

DEPTNO ENAME SAL

-----------------------------------------------------------

10 miller 1300

20 smith 800

30 james 950

***AIM10:-***

Write a query to display the department number, which have an average salary greater than the average salary of department number 30. (HAVING SUBQUERIES)

***QUERY:-***

SQL>SELECT DEPTNO FROM EMP

GROUP BY DEPTNO

HAVING AVG(SAL) > (SELECT AVG(SAL) FROM EMP

WHERE DEPTNO = 30);

***OUTPUT:-***

DEPTNO

------------

10

20