18/12/19

DBMS

RELATIONAL DATABASE MANAGEMENT SYSTEMS :

→ RDBMS is a collection of information in the form of number of related tables.

→ The RDBMS is based on the relational model proposed by the person Edgar E codd in the year 1970.

→ He proposed 12 rules as part of relational models and all these rules has to be satisfied by a RDBMS.

→ The relational model bus is based on relational algebra.

→ It was first developed as a prototype System 'R' by IBM in 1974.

→ The first commercial RDBMS is the Multix Database.

→ The next commercial and most saled database is the Oracle 2 released by relation software company in 1979.

→ The next versions of company are

Oracle 3
Oracle 4
Oracle 5
⋮

Oracle 10g - Grid computing
Oracle 12c - Cloud computing
Latest version - 19c

→ (Oracle) The other types of RDM available are
i) SQL ii) MYSQL iii) DBL
iv) Teradata v) MS offices vi) Informics.
vii) MangoDB etc.

STRUCTURED QUERY LANGUAGE (SQL):

→ SQL is the database language that it was developed by IBM in 1970 for system and was originally called as "structured English Query Language" which has been changed to SQL now.

→ SQL supports a number of commands in order to write a query with which we can interact with the database.

QUERY:

A Query is a question given to the database in SQL.

→ SQL is a non-procedural language where we mention only what we want to retrieve from database without mentioning the procedure to be followed.

→ The other DBMS languages available are QL, CQL, XQUERY, YQL, QBE etc.

ORACLE DATA TYPES :

* char(size) :-

This data type is used to store character, string values of fixed length. The maximum number of characters it can hold is 255 characters.

* varchar(size) :-

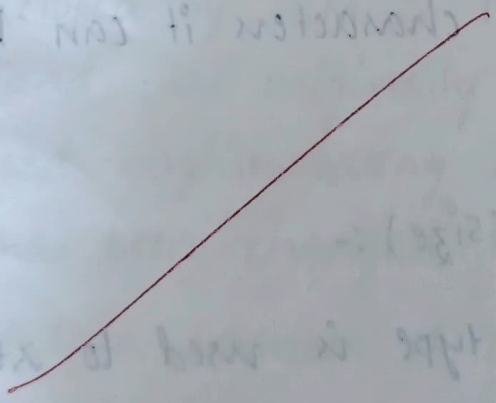
This data type is used to store variable length alpha numeric data. The maximum no. of characters it can hold is 4000 characters.

*Number (P, S) :-

used to store numbers of integers and floating point numbers. P is the precision which determines maximum length of data. S is the scale which determines the no. of places to the right of decimal values.

* Date :-

This data type will represent the date and type. The standard format of the date is "DD-MON-YY."



13/12/19

DDL : (Data Definition Language)

* CREATE :- used for creating data objects like tables, views, triggers.
* ALTER :- used to modify the structure of data objects.
* DROP :- Used to remove the data objects from database
* TRUNCATE :- used to remove the data object as well as the space occupied by the data
* RENAME :- used to rename the existing table.
* COMMENT :- used to add comment statements for data dictionary.

DDL commands used to define the structure of database and modify structure.

DML : (Data Manipulation Language)

These commands are used for inserting the data, modifying/ deleting existing data.

* INSERT :- used to insert data into a table
* UPDATE :- used to modify existing data
* DELETE :- used to delete data/rows/tuples from a table.

DCL :- (Data control Language)

These are used to control the access on the database.

* GRANT :- used to give access previliges for database users.
* REVOKE :- used to remove access previliges for DB users.

TCL :- (Transaction control Language).

These are used for controlling the operations of different transactions.

* COMMIT :- used to save the work done.
* ROLLBACK :- Restores the DB to original since last commit statement.
* Savepoint :- used to define a point in a transaction to which you can rollback later.

Other commands under TCL are,

* SET-TRANSACTION
* CALL

DQL : (Data Query Language)

Used for retrieving data from database.

* SELECT :- Retrieves the data

Under DML the other commands are MERGE, EXPLAIN-PLAN, LOCK and CALL

TABLE CREATION :

CREATE statement is used to create a table.

Syntax :

SQL > Create table <tablename>(
column1 type(size), ...
column2 type(size), ...);

Ex :

Create table student(name char(10),
roll-no varchar2(15), branch char(5));

Output : Table created.

Describe command :- used to describe the definition of relation.

Syntax :

SQL > desc tablename ;

(or)

describe tablename ;

Ex :

desc student ;

Inserting data into tables :-

A row / tuple can be inserted with different syntaxes of insert command.

Syntax 1 :

insert into <tablename> values (values seperated with commas) ;

Ex 1 :

insert into student values ('ABC', 19, 'CSE');

Output :-

1 row created.

↓

All attributes must be mentioned.

Syntax 2 :

In this syntax we will mention the column names and values that we want to insert.

insert into <tablename> (col1, col2, col3 .... coln)
values (v1, v2, v3 .... vn); ↓ Assign values to specific columns

Syntax 3 :

In order to add the row values interactively we can use the below syntax with the help of a substitution variable & → All attributes must be mentioned

insert into <tablename> values (&col1, &col2, &col3, ... , &coln);

RUN Command :-

This command is used to execute the previous commands.

Syntax :-

SQL > run
(or)
SQL > /

Selecting the data or retrieving the data :-

The SELECT command can be used to retrieve data from one or more tables.

Syntax 1 :-

select * from <tablename>

The above command displays all the rows/ tuples present in the mentioned table.

Ex :-

select * from student;

Syntax 2 :-

select * [OR] column names from <tablename>
[where clause]
[Group by clause]
[having clause]
[order by clause]

The above command is used to retrieve the data based on the condition specified in different clauses.

Ex :-

select * from student where branch = 'CSE';

Select statement can be used to retrieve the data from selected columns also.

Output :-

```
SQL > connect
Enter User-name : 19165A0504
Enter Password :
connected
SQL > create table student(Name char(15), Roll-no
      varchar2(15), Branch char(10));
Table created.
SQL > desc student
```

Name	Null ?	Type
NAME		CHAR(10)
ROLL-NO		VARCHAR2(15)
BRANCH		CHAR(10)

```
SQL > insert into student values('abc', 01, 'CSE');
1 row created
SQL > insert into student(Name, Roll-no, Branch)
      values('xyz', 02, 'ECE');
1 row created.
SQL > insert into student values(&Name, &ROLLNO
      &Branch);
```

Enter value for name : 'pqr'
Enter value for ROLL-NO : 03
Enter value for branch : 'EEE'

1 row created

SQL > run
1* insert into student values(&Name, &ROLL-NO, &Branch);

Enter value for name : 'aaa'
Enter value for ROLL-NO : 04
Enter value for branch : 'CSE'

1 row created

SQL > select * from employee student;

NAME	ROLL-NO	BRANCH
abc	1	CSE
xyz	2	ECE
pqr	3	EEE
aaa	4	CSE

SQL > select Name from student where (Branch = 'CSE');

NAME
abc
aaa

SQL > insert into student values('abd', 7);
ERROR : Not enough values.

* Display the details of the student who belong to CSE branch.

Query :

select * from student where (Branch = 'CSE');

NAME	ROLL-NO	BRANCH
abc	1	CSE
aaa	4	CSE

* COMMIT COMMAND :-

used to save the previous work done.

Syntax :-

SQL > commit ; → commit complete

* To know the date format supported by database

SQL > select Sysdate from dual ;

SYSDATE
23-DEC-19.

* Create a Employee table having following attributes,
Name, id, designation, Date of Joining and salary.

* Create a department table having department id, department name and location.

EMPLOYEE TABLE :

```
SQL > Create table employee (e_id number(5),
      e_name char(10), e_designation char(15),
      e_doj date, e_salary number(7,2));
```

Table created.

```
SQL> insert into employee values(1, 'Raju',
     'manager', '02-JAN-15', 60000.80);
```

1 row inserted.

```
SQL> insert into employee values(2, 'Ravi',
     'Head', '12-JUN-15', 65000.80);
```

1 row inserted.

```
SQL > select * from employee;
```

E_ID	E_NAME	E_DESIGNATION	E_DOJ	E_SALARY
1	Raju	Manager	02-JAN-15	65000.80
2	Ravi	Head	12-JUN-15	60000.80
3	Geetha	Accountant	23-FEB-15	25000
4	Geetha	HR	17-MAR-16	75000.5
5	Hari	Accountant	27-MAR-16	25000
6	Priya	Head	17-MAR-16	35000
7	Rupa	clerk	07-MAR-17	15000
8	Vinay	clerk	28-APR-16	25000
9	Shiva	A_Manager	18-DEC-16	55000
10	Mounika	Manager	04-JUL-16	35000.

DEPARTMENT TABLE :

```
SQL > Create table department (id number(3),
      name char(10), location char(10));
```

Table created

```
SQL > insert into department values (101,
      'HR_dept', 'Begumpet');
```

1 row inserted.

```
SQL > insert into department values (102,
      'Acnt_dept', 'Begumpet');
```

1 row inserted.

```
SQL > Select * from department;
```

ID	NAME	LOCATION
101	HR-dept	Begumpet
102	Acnt-dept	Begumpet
103	Exam-dept	Begumpet
104	Admin-dept	Begumpet
105	Training	Begumpet

* In order to see the various tables of present user

SQL > select * from tab;

T-NAME	TAB TYPE	CLUSTERID
STUDENT	TABLE	
EMPLOYEE	TABLE	

* In order to see the current user we can use

SQL > Select user from dual;

USER

192G5A0504

* In order to view all the tables present in the database we use

SQL > select table_name from all_tables

TABLE_NAME

DUAL
HELP
SYSTEM_PREVILIGE_MAP
AUDIT_ACTIONS ..... etc.

SQL > desc tab;

Name	Null ?	Type
TNAME	Not NULL	VARCHAR2(30)
TABTYPE		VARCHAR2(7)
CLUSTERID		NUMBER

SQL > desc dual

Name	Null ?	Type
DUMMY		VARCHAR2(1)

SQL > Select * from dual;

D
X

SQL > desc all_tables;

Name	Null ?	Type
OWNER	Not NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)

PCT-FREE	NUMBER
PCT-USED	NUMBER
LAST-ANALYSED	DATE
DROPPED	VARCHAR2(3)
.....	
.....	

Read
23/12/19

30/12/19

*Select statement can be used to retrieve the columns

Syntax :-
select column-names from table name ;

Ex :-
select Roll-no, Name from student ;

*In-order to display the contents of a table on a condition we can use where clause in the select statement.

Syntax :-
select columns from table-name where condition.

The rows which satisfy the condition will be displayed by the select statement. The where condition should be a boolean expression that returns true or false. We can combine multiple conditions using logical operators AND, OR, NOT. The * in select statement indicates all the columns of the table.

Queries :-

* Display the details of employee ID and designation of employees.

SQL > Select e-id, e-designation from employee;

Output :

E_ID	E_DESIGNATION
1	Manager
2	head
3	Accountant
4	HR
5	Accountant
6	Head
7	clerk.

* Display the details of the students who belong to ECE branch.

SQL > select * from student where (branch = 'ECE');

Output :

NAME	ROLL-NO	BRANCH
xyz	2	ECE

* Display the names of employees who are drawing salary more than 25,000

SQL > select e_name from employee where e_salary > 25000;

Output :

E_NAME
Raju
Ravi
Geetha
Hari
Priya
Shiva
Mounika

* In order to remove the duplicate data while retrieving and displaying from a table we can use "distinct" keyword in select statement.

Ex :-

Select e-designation from employee;

Output :

E_DESIGNATION
manager
head
Accountant
HR
Accountant
Head
clerk
clerk
A_manager
Manager.

SQL> select distinct e-designation from employee;

Output:

E-DESIGNATION
A-manager
Manager
Accountant
HR
head
Head
manager
clerk

* Display the details of employees who are managers.

SQL> select * from employee where designation = 'manager';

E-ID	E-NAME	E-DESIGNATION	E-DOJ	E-SALARY
1	Raju	manager	02-JAN-15	60000.8

* Display the details of students who got more than 500 marks.

SQL> select * from student where (marks > 500) roll-no = 2;

Output:

NAME	ROLL-NO	BRANCH
xyz	2	ECE

MODIFYING THE SCHEMA (Or) STRUCTURE OF RELATIONS:

In order to modify the structure of a relation we can use 'alter' command. This command allows to,

- Add or delete columns
- create or destroy indexes
- change the size of existing columns
- change the name of existing columns.

Adding new columns :-

Syntax :-

alter table <table name> add (<col. name 1> type (size), <col. name 2> type (size), ...);

Ex :-

alter table student add (dob date);

Output :- Table altered.

Dropping a column :-

alter table <tablenames> drop column-name;

Ex:

alter table student drop column dob;

Modifying the size of existing columns :-

Syntax :-

alter table <tablename> modify (col.name newdatatype (size));

Ex :-

alter table employee modify (e_salary number (10,3));

Renaming the existing column :-

Syntax :-

alter table <tablename> rename column oldname to newname;

Ex :-

SQL> alter table employee rename column e_salary to e_sal;

Queries Using Alter command :

* Modify the student table by adding Date of Birth details.

SQL> alter table student add (dob date);

Output :

Table altered.

* Alter the employee table by adding the dept_id column.

SQL> alter table employee add (dept_id number(3));

Output :

Table altered

* Alter the employee table by adding the HRA and DA attributes (where)

SQL> alter table employee add (HRA number varchar2(10), DA number varchar2(5));

Output :

Table altered.

DELETE COMMAND :

→ This command deletes rows of data from the table if no condition is specified.

→ when a condition is specified in delete the rows of data that satisfy the condition will be deleted.

Syntax 1 :-

```
delete from <tablename>;
```

Syntax 2 :-

```
delete from <tablename> where condition;
```

→ The delete is a DML command which can be rolled back (If commit is not used in between).

* SQL> delete from student;

Output :

4 rows deleted

* SQL> rollback;

Output :

Rollback complete.

SQL> delete from student where branch = 'EEE';

1 row deleted.

SQL> select * from student;

Output :

NAME	ROLL-NO	BRANCH	MARKS
abc	1	CSE	
xyz	2	ECE	
aaa	4	CSE	

SQL> commit;

Commit complete.

SQL> rollback;

Rollback complete;

SQL>

no rows selected.

UPDATING THE CONTENTS OF TABLE :

→ The update command is used to modify the content present in a table.

→ This command can be used to update all the rows of a table or some of the rows of a table based on a condition.

Syntax-1 :-

SQL > Update <tablename> set <col.name1> = <exp-1>,
<col.name2> = <exp-2>;

Syntax-2 :

SQL > Update <tablename> set
<col.name1> = <exp1>,
<col.name2> = <exp2> where condition;

→ Update is a DML statement and the expressions in the syntax can be values or arithmetic/logical expressions.

Ex :

SQL> Update employee set dept_id = 103;
10 rows updated.

SQL> select e-id, dept-id from employee;

E-ID	dept_ID
1	103
2	103
3	103
4	103
5	103
6	103
7	103
8	103
9	103
10	103

SQL > Update employee set dept-id = 101 where e-designation = 'Accountant';
2 rows updated.

SQL> select e-designation, dept-id from employee;

E-DESIGNATION	DEPT-ID
Manager	103
Head	103
Accountant	101
HR	103
Accountant	101
Head	103
clerk	103
clerk	103
A-manager	103
Manager	103

SQL> Update employee set dept-id = 102 where e-designation = 'Manager';
2 rows updated.

SQL > Update employee set dept-id = 104 where e-designation = 'clerk';
2 rows updated.

SQL> select * fd

SQL> select e-designation, dept-id from employee;

E-DESIGNATION	DEPT-ID
Manager	102
Head	103
Accountant	101
HR	103
Accountant	101
Head	103
Clerk	104
Clerk	104
A-manager	104
Manager	102

* Alter the student table schema with DOB and update the contents for this attribute with update command.
* Modify employee table by adding location attribute

OPERATORS IN SQL :

1) Arithmetic operators :-

The various arithmetic operators available are +, -, /, *, **, ()

2) Relational operators :-

The various relational operators available are <, <=, >, >=, ==, != . These are used to compare two expressions and the result of comparison will be true or false.

3) Logical operators :-

These operators will combine the result of two conditions to produce a single result based on them.
AND, OR, NOT.

Queries :-

* Update the student information by giving 5 extra masks for assignments for CSE student

SQL> Update student set masks = masks + 5
where branch = 'CSE';
2 rows updated.

SQL> select marks from student where branch='cse';

MARKS
583
461

change the salary of employee by giving a bonus of 1000.

SQL> update employee set e_salary=e_salary+1000;

10 rows updated.

SQL> select e_sal from employee;

E_SAL
61000.8
66000.8
26000
76000.5
26000.5
36000.5
16000.5

Modify the salary of employee by giving 10% hike for attenders.

SQL> update employee set e_sal=e_salary+(0.1*e_sal) where e_designation='clerk';

2 rows updated.

SQL> select e_sal from employee where e_designation='clerk';

E_SAL
19360.605
31460

* Delete the employee info whose salary is less than 35,000 and dept_id = 102.

SQL> delete from employee where e_sal<35000 AND dept_id=102;

2 rows deleted

SQL> select e_sal, dept_id from employee;

E_SAL	DEPT_ID
67100.88	102
72600.88	103
83600.50	103
39600.50	103

rollback.

* Update the HRA and DA of employees such that HRA = 30% of salary DA = 10% of salary

SQL> update employee set HRA=e_sal+(0.3*e_sal);

SQL> update employee set DA=e_sal+(0.1*e_sal);

SQL> select e_sal, HRA, DA from employee;

E_SAL	HRA	DA
67100.88	87231	73811
72600.88	94381	79861

28600	31180	31460
83600.55	108681	91961
20600.55	37181	31461
39600.55	51481	43561
19360.605	25169	21297
31460	40898	34606.

Student table :-

```
SQL> alter table student add(dob date);
Table altered.
SQL> update student set dob = '11-JAN-2001'
     where marks = '578';
1 row updated.
SQL> update student set dob = '02-feb-2001'
     where marks = '512';
1 row updated.
SQL> update student set dob = '12-jun-2001'
     where marks = '437';
1 row updated
SQL> update student set dob = '25-apr-2001'
     where marks = '456';
1 row updated.
```

```
SQL> Select * from student;
```

NAME	ROLL-NO	BRANCH	MARKS	DOB
Lakshmi	10	CSE	578	11-JAN-01
Sowjanya	15	CSE	456	25-APR-01
Pavani	19	EEE	437	12-JUN-01
Shirisha	17	ECE	512	02-FEB-01

Employee table :

```
SQL> alter table employee add(location char(10));
Table altered.
SQL> Update employee set location = 'Begumpet'
     where e_id = 1;
1 row updated.
SQL> select e_id, e_name, location from employee;
```

E-ID	E-NAME	LOCATION
1	Raju	Begumpet
2	Vinay	Secunderabad
3	Rupa	Begumpet
4	Geetha	Ameerpet
5	Mounika	Uppal.

* Update HRA and DA of employee salary such that HRA = 30% of salary
DA = 10% of salary

SQL> Update employee set HRA = 0.3 * e_sal,
DA = 0.1 * e_sal;

10 rows updated.

SQL> select e_sal, HRA, DA from employee;

E_SAL	HRA	DA
67100.88	20130	6710
72600.88	21780	7260
28600	8500	2860
83600.55	25080	8360
28600.58	8580	2860

Dml: (5)

6/1/2020

DROP & TRUNCATE COMMANDS :-

→ The Drop command is used to delete all the rows in a table. It also removes the structure of the table from the database. Once a table is dropped we cannot get back the table.

Syntax :
drop table <tablename>

→ The truncate command is used to delete all the rows present in a table. when we use truncate on a table the structure of table will remain in the database.

Syntax :
truncate table <tablename>

→ All the DDL statements are auto committed They cannot be rolled back.
→ Truncate and drop are DDL statements
→ The DML commands can be rolled back.

Differences between Drop and Truncate :-

Drop	Truncate
* It removes all the rows, indexes, constraints, triggers	* Removes all the rows from table.

defined on the table will be removed. * The structure of table will not exist * can't be rolled back	* The structure of table still exist. * Cant be rolled back.

SPECIAL OPERATORS IN SQL :

The various special operators available in sql are, in, not in, between, like, is NULL, any, all, exist operators.

Between :- It returns the set of rows from a table between and inclusive of the range of values specified.

* List the details of employees whose salary is between 35,000 & 50,000

SQL> select * from employee where salary between 35000 and 50,000;

Output :

E-ID	E-NAME	E-DESIGNATION	E-SAL	DEPT-ID
6	Priya	Head	39600.55	103
10	Mounika	Manager	39600	102

* **Not between :-** It returns the records from the table not between the specified inclusive range.

* Find the names of students whose marks are not between 400 & 550

SQL> select name from student where marks not between 400 and 550;

Output :

NAME
Lakshmi

In :- This operator is used to test the membership of an element in a set of values when a match is found for a value the in operator returns true which in turn will return the particular row from the table.

* Find the names of employees whose designation is manager and clerk.

SQL> select e-name from employee where e-designation in ('Manager', 'Clerk');

Output :

E-NAME
Raju
Rupa
Vinay
Mounika

* Modify the marks of the students for the roll-nos 10, 19, 17

SQL> Update student set marks = marks + 5 where roll-no in (10, 19, 17);

Output:

3 rows updated

Not in :- It returns all the records except the values that we have mentioned in the list.

* Find the names of students who does not belong to CSE, ECE, EEE departments.

SQL> select name from student where branch not in ('CSE', 'ECE', 'EEE');

Output:

NAME
Anjali

* change the salary of employees by giving 20% of hike who are not living at Begumpet, Uppal, Secunderabad.

SQL> Update employee set salary = salary + 0.2 * salary where location not in ('Begumpet', 'Uppal', 'secunderabad');

Output:

10 rows updated.

is NULL :- It returns the records of a table for which the value of specified column is NULL.

* List the names of employees who didn't provide the details of their location.

SQL> select E-name from employee where location is null;

Output:

E-NAME
Hari

is not NULL :- Returns the records of table for which the value of specified column is not NULL.

* List the details of students whose marks are ~~not~~ given

SQL> Select * from student where marks is not null;

Output:

NAME	ROLL-NO	BRANCH	MARKS	DOB
Lakshmi	10	CSE	588	11-JAN-01
Sowjanya	15	CSE	461	25-APR-01
Pavani	19	EEE	442	12-JUN-01
Shirisha	17	ECE	517	02-FEB-01

Like :- This operator is used for pattern matching in a tabular data. It uses 2 wild card characters % , _ for pattern matching.

- % represents 0 (or) more no. of unknown characters.
- _ represents 1 unknown character.

* List the names of employees who are living in a location which begins with letter A.

SQL> select e-name from employee where location like 'A%';

Output:

E-NAME
Ravi
Geetha

* List the names of student whose name will end with letter 'i' and has exactly 4 letters.

SQL> select name from student where name like '___i%';

Output:

NAME
Ravi

QUERIES ON ABOVE OPERATORS :

* Find the details of employees whose designation is clerk.

SQL> select * from employee where e-designation = 'clerk';

Output:

E-ID	E-NAME	E-DESIGNATION	E-DOJ	E-SAL
7	Rupa	Clerk	07-MAR-17	23232
8	Vinay	clerk	28-APR-16	37752

* Find the names of employees who are not living in Begumpet.

SQL> Select e-name from employee where location not in('Begumpet');

E-NAME
Ravi
Geetha
Hari
Rupa

* Display the name & id of employee who is not a manager and has salary more than 90,000 and lives in the location Ameerpet.

Output:-

E-NAME	E-ID
Geetha	4

select e.name, e.id from employee where e.designation != 'Manager' and e.sal > 90000 and location = 'Ameerpet';

* Find the details of employee who reside at Ameerpet (or) Uppal.

SQL > select * from employee where location in ('Ameerpet', 'Uppal');

E-ID	E-NAME	E-DESIGNATION	LOCATION
2	Ravi	Head	Ameerpet
3	Geetha	Accountant	Uppal
10	Mounika	Manager	Uppal
7	Rupa	Clerk	Ameerpet

* List the names of employees who joined after june and before dec-31 in 2015.

SQL > select e.name from employee where e.doj > '01-JUN-15' and e.doj < '31-DEC-15';

Output:

E-NAME
Ravi

* select e.name from employee where e.doj between '01-JUL-15' and '30-NOV-15';

* Display the names of employees who are not HR & Manager.

SQL > select e.name from employee where e.designation not in ('HR', 'Manager');

Output:

E-NAME
Ravi
Geetha
Hari
Priya
Rupa

(1) Find the details of employee → whose years of experience is between 5 and 8. & belong to dept id = 4.

SQL > select * from employee where ((sysdate - doj)/365 between 5 and 8) and deptid = '4';

Queries on derived attributes

* Find the experience of employees from the employee table.

SQL > Select sysdate - e.doj as exp from employee;

Output:

EXP
1830.60428
1669.60428
1778.60428
1390.60428
1380.60428

Days

(Sysdate - e.doj)/30 or 31 → Months

(Sysdate - e.doj)/365 → Years

* Display the age of students present in the student relation.

SQL > Select sysdate - dob as age from student;

Output:

AGE
6934.60729
6830.60729
6782.60729
6759.60729

* Find the gross salary of employees where gross salary = salary + HRA + DA.

```
SQL> select (e.sal+HRA+DA) as g_sal from
     employee;
```

Output:

G_SAL
107361.056
116161.056
45760
133760.66
45760.66
63360.66

* Display the id, name, PF of employees present in employee relation where PF is given as 10% of salary.

```
SQL> Select e_id, e_name, (0.1* e_sal) as PF from
     employee;
```

Output:

E_ID	E_NAME	PF
1	Raju	8052.1056
2	Ravi	8712.1056
3	Geetha	3432
4	Geetha	10032.066
5	Hari	3432.066
6	Priya	4752.066
7	Rupa	2323.2726
8	Vinay	3775.2

* Display the details of students whose branch name ends and starts with E

```
SQL> select * from student where branch
     like 'E_E%';
```

Output:

NAME	ROLL_NO	BRANCH	MARKS	DOB
Pavani	19	EEE	442	12-JUN-01
Shinisha	17	ECE	512	02-FEB-01

* Display the details of employee whose name starts with r and ends with i

```
SQL> select * from employee where e_name
     like 'R%i%';
```

Output:

E_ID	E_NAME	E_DESIGNATION	E_DOJ
2	Ravi	Head	12-JUN-15

* List the names of employees who have the second character in their name a or s.

```
SQL> select e_name from employee where
     e_name like '_s%' or e_name like '_a%';
```

Output:

```
E-NAME
-------
Raju
Ravi
Hani
```

```
SQL> truncate table std;
Table truncated.
SQL> desc std;
 Null ?      Type
 ------      -----
 S-ID        NUMBER(3)
 S-NAME      VARCHAR2(5)

SQL> drop table std;
Table dropped
SQL> desc std;
Object std does not exist.
```

Ravi 6/1/20

10

20/1/2020

SORTING OF DATA PRESENT IN A TABLE :

→ The data tuples that are retrieved can be sorted either in ascending or descending order based on the condition specified in the select statement.

→ By default the retrieved data is sorted in ascending order of the key field.

Syntax:

```
Select * (or) column-names from table-name
order by <col-name1> <col-name2> [desc];
```

→ If we want to sort the retrieved data in descending order of column value we use the keyword desc.

→ We can sort the data based on more than one column also.

→ Instead of using the column names we can use column numbers also.

* Display the details of employee in the ascending order of their salaries.

```
SQL> select * from employee order by
     e_sal;
```