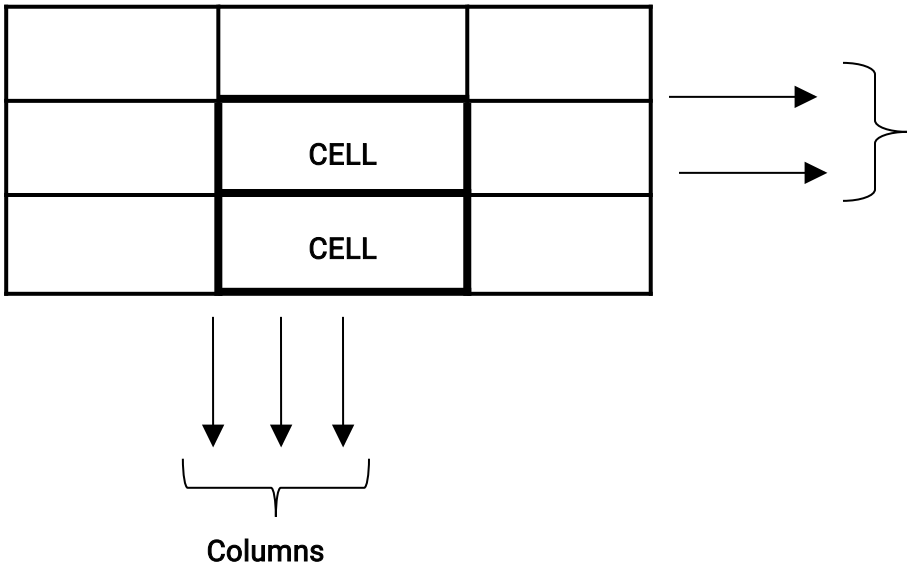
Pradeepamr007@gmail.com

# *CHAPTER 1*

# RDBMS Concepts

**Database**
A database is the place of storage of the data in the form of tables
Data means information which is very useful. A database is also collection of 1 or more tables.

**Table** – a table is a collection of rows and columns.




A cell is an intersection of a row and a column
A column is also called as a field / attribute
A record is also called as a row / tuple.
A table is also called as an entity / relation.

***Note*** **:-**

- If we install any of the database related software(s) – we can create our own database, we can create our own tables and we can store the data inside it.
- When we install any database s/w(s) – a part of hard disk will be designated / reserved to perform database related activities
- A database can also contain other database objects like views, indexes, stored procedures, functions, triggers etc, apart from tables.

Some of the database software(s) we have are,
Oracle, SQL Server, DB2, Sybase, Informix, MySQL, MS – Access, Foxbase, FoxPro

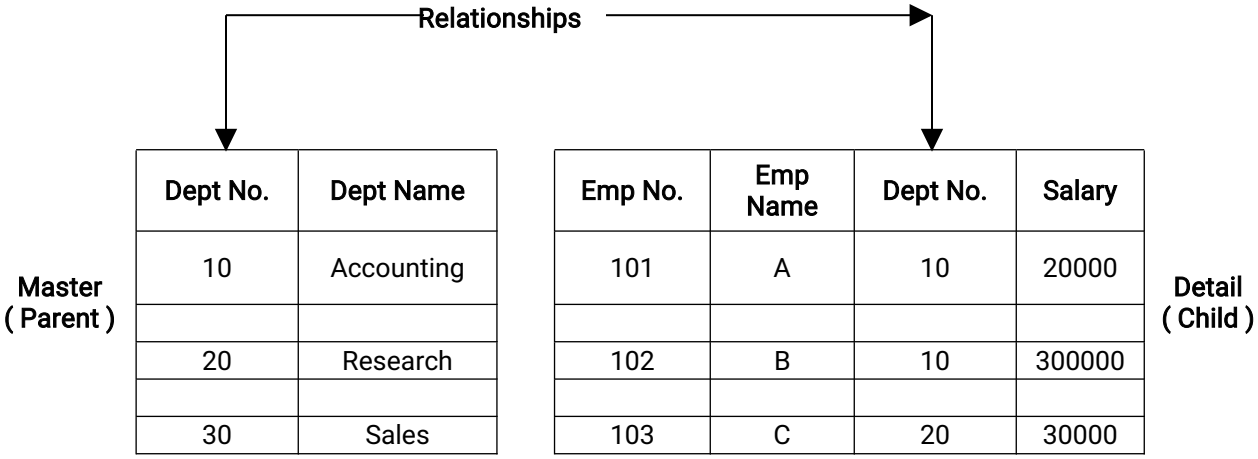
Among the above database software – some of them are DBMS and some of them are RDBMS

The s/w which is widely used today is Oracle. The different versions of Oracle starting from the earliest

to the latest are – Oracle 2, Oracle 3, Oracle 4, Oracle 5, Oracle 6, Oracle 7, Oracle 8i, Oracle 9i, Oracle 10g, and the latest to hit the market is Oracle 11g. here 'i' stands for Internet and 'g' stands for Grid / Grid computing.

## RELATIONSHIPS

A relationship is the association between any two tables which preserves data integrity.




| Dept No. | Dept Name |
|---|---|
| 10 | Accounting |
| | |
| 20 | Research |
| | |
| 30 | Sales |

| Emp No. | Emp Name | Dept No. | Salary |
|---|---|---|---|
| 101 | A | 10 | 20000 |
| | | | |
| 102 | B | 10 | 300000 |
| | | | |
| 103 | C | 20 | 30000 |

Relationship helps to prevent the incorrect data in the child tables
Once the relationship is created, one table becomes master (or parent) and the other one becomes the child ( or detail ).
Whatever we insert into the child should be present in the master, else the record will be rejected from the child.
The master table contains the master data which will not change frequently.
The child table contains the transactional data which will change quite often.

## DBMS & RDBMS

**DBMS** – stands for Database Management System
DBMS is a database s/w which allows us to store the data in the form of tables.

**RDBMS** – stands for Relational DBMS
RDBMS is also a database s/w which has facility to handle more data volume, good performance, enhanced security features etc when compared against DBMS.
Any DBMS to qualify as a RDBMS should support the Codd rules / Codd laws

**Ex** for DBMS – FoxPro, FoxBase, Dbase
**Ex** for RDBMS – Oracle, Sybase, DB2, Teradata, SQL Server, MySQL

## CONSTRAINTS

A constraint is a condition which restricts the invalid data in the table.
A constraint can be provided for a column of a table.

## Types of Constraints

- ❖ NOT NULL
- ❖ UNIQUE
- ❖ Primary Key
- ❖ Foreign Key
- ❖ Check

**NULL**

- → NULL is nothing, it is neither zero nor blank space
- → It will not occupy any space in the memory
- → Two NULLS are never same in Oracle.
- → NULL represents unknown value
- → Any arithmetic operation we perform on NULL will result in NULL itself. **For ex,** 100000 + NULL = NULL ; 100000 * NULL = NULL

**NOT NULL**

- NOT NULL will ensure atleast some value should be present in a column

**UNIQUE**

- → It will not allow any duplicates in a column
- → UNIQUE column can take multiple NULL (s)

**Primary Key**

- → It is the combination of **NOT NULL** and **UNIQUE**
- → Only one PK is allowed in a table
- → PK identifies a record uniquely in a table
- → Creation of PK is not mandatory, but it is highly recommended to create

**Foreign Key**

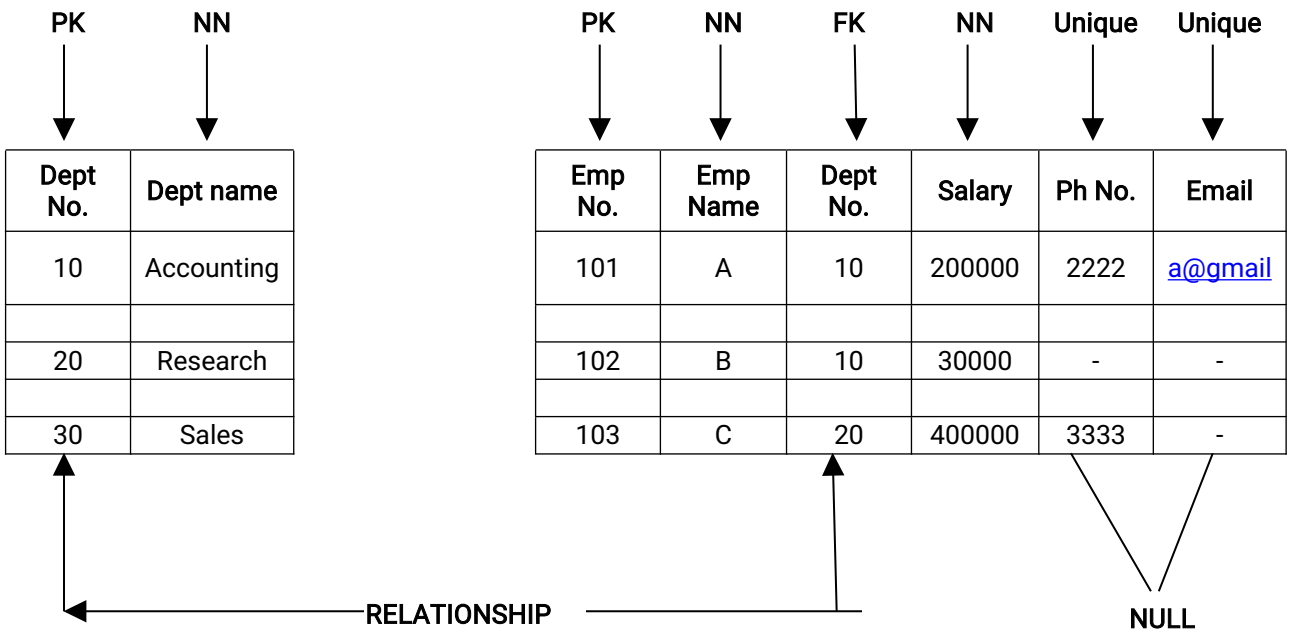
- → FK creates relationship between any two tables
- → FK is also called as referential integrity constraints
- → FK is created on the child table
- → FK can take both NULL and duplicate values
- → To create FK, the master table should have PK defined on the common column of the master table
- → We can have more than 1 FK in a given table

**CHECK**

It is used to provide additional validations as per the customer requirements.

**Ex -** 1) sal > 0
2) empnum should start with 1
3) commission should be between 1000 & 5000

**Check (sal > 0)**




*CHAPTER 2*

# SQL – Structured Query Language

SQL – Structured Query Language
SQL – it is a language to talk to the database / to access the database

SQL – it is a language, whereas SQL server is a database.
To work on SQL , a DB software (RDBMS) is required.
SQL is not case sensitive

**Username** - Scott
**Password** – Tiger

**Troubleshooting Oracle**

*Error 1*
The account is locked

*Steps to rectify the error*

- Login as username – 'system' & password – 'manager' or 'password – 'tiger'
- SQL > show user ;
  User is "SYSTEM"

  SQL > alter user scott account unlock ;
  User altered

  SQL > exit ;

*Error 2*
TNS : protocol adapter error

*How to troubleshoot this*
Cause is "oracle service has not started"

How to go here,
Settings – Control Panel – Administrative Tools – Services
Sort the list
There is an "Oracle Service ORCL" & "start the service"

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
```

This query gives the list of tables.
* - selects all

```
SQL> desc dept ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

This query gives the description of the table “department”.
The description of the table has **column names, constraints, datatypes**

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

This query gives the description of the table “department”

```
SQL> select * from emp ;

     EMPNO 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


     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM
---------- ---------- --------- ---------- --------- ---------- ----------
    DEPTNO
----------
      7566 JONES      MANAGER         7839 02-APR-81       2975
        20
```

The above query gives the description of the “employee” table. But we see that all the data is in different lines which makes it very difficult to analyse.
So we use the following command to see the data in a more orderly fashion,

```
SQL> set linesize 120 ;
SQL> select * from emp ;

     EMPNO 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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      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

14 rows selected.
```

The **“set linesize”** command helps in increasing the line size , thus the data is arranged in a orderly fashion.

```
SQL>  set pagesize 20 ;
SQL> select * from emp ;

     EMPNO 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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       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

14 rows selected.
```

The above command **“set pagesize 20”** increases the page size, thus accommodating more number of rows in a single page.

```
SQL> select ename, job, sal
  2  from emp ;

ENAME      JOB              SAL
---------- --------- ----------
SMITH      CLERK             800
ALLEN      SALESMAN         1600
WARD       SALESMAN         1250
JONES      MANAGER          2975
MARTIN     SALESMAN         1250
BLAKE      MANAGER          2850
CLARK      MANAGER          2450
SCOTT      ANALYST          3000
KING       PRESIDENT        5000
TURNER     SALESMAN         1500
ADAMS      CLERK            1100
JAMES      CLERK             950
FORD       ANALYST          3000
MILLER     CLERK            1300

14 rows selected.
```

The above query gives the value of only these 3 columns from the table “employee”.

```
SQL> select * from emp where sal = 3000 ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

**‘where’** clause is used to restrict the number of records displayed. It gives only the records of the specified condition.

```
SQL> select * from emp where job='MANAGER' ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

Any string data should be enclosed within **single quotes** ( ‘ ‘ ) and the same becomes **case sensitive**.

Assignment

1) List the employees in dept 20

```
SQL> select * from emp where deptno = 20 ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

2) List the employees earning more than Rs 2500.

```
SQL> select * from emp where sal > 2500 ;
```

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |

3) Display all salesmen

```
SQL> select * from emp where job= 'SALESMAN' ;
```

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |

*CHAPTER 3*

# OPERATORS

Operators are classified into,

- **Arithmetic Operators** ( +, - , * , / )
- **Relational Operators** ( > , < , >= , <= , = , < > **or** != - not equals to )
- **Logical Operators** ( NOT, AND, OR )
- **Special Operators** ( IN , LIKE , BETWEEN , IS )

SPECIAL OPERATORS

**1) IN** – it is used for evaluating multiple values.

**Ex – 1)** List the employees in dept 10 & 20

```
SQL> select * from emp where deptno in (10 , 20 ) ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

8 rows selected.
```

**2)** List all the clerks and analysts

```
SQL> select * from emp where job in ('CLERK', 'ANALYST' ) ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       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

6 rows selected.
```

We can provide upto 1000 values at the max

**2) LIKE** – used for pattern matching

**% (percentage) -** matches 0 or 'n' characters
**_ (underscore)** - matches exactly one character

Ex – 1) List all the employees whose name starts with 'S'

```
SQL> select * from emp where ename like 'S%' ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
```

Whenever we use % or _ , always ensure that it is preceded by the word **'like'**

2) List the employees whose name is having letter 'L' as 2nd character

```
SQL> select * from emp where ename like '_L%' ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

ASSIGNMENT
1) List the employees whose name is having atleast 2 L's

```
SQL> select * from emp where ename like '%_LL_%';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

2) List the employees whose name is having letter 'E' as the last but one character

```
SQL> select * from emp where ename like '%_E_';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
```

3) List all the employees whose name is having letter 'R' in the 3rd position

```
SQL> select * from emp where ename like '__R%';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
```

In the above query, we give 2 underscores before R%.

4) List all the employees who are having exactly 5 characters in their jobs

```
SQL> select ename, job from emp where job like '_____';

ENAME      JOB
---------- ---------
SMITH      CLERK
ADAMS      CLERK
JAMES      CLERK
MILLER     CLERK
```

Here , in single quotes – we give 5 underscores.

5) List the employees whose name is having atleast 5 characters

```
SQL> select ename from emp where ename like '_____';

ENAME
----------
SMITH
ALLEN
JONES
BLAKE
CLARK
SCOTT
ADAMS
JAMES

8 rows selected.
```

Here, also in single quotes – we give 5 underscores ( _____ )

3) **BETWEEN** operator – used for searching based on range of values.
**Ex – 1)** List the employees whose salary is between 200 and 300

```
SQL> select * from emp where
  2  sal between 2000 and 3000 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
```

**4) IS** operator – it is used to compare nulls

**Ex – 1)** List all the employees whose commission is null

```
SQL> select * from emp where comm is null ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
      7876 ADAMS      CLERK            7788 23-MAY-87       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

10 rows selected.
```

ASSIGNMENT

1) List all the employees who don't have a reporting manager

```
SQL> select * from emp where mgr is null ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
```

LOGICAL OPERATORS

1) List all the salesmen in dept 30

```
SQL> select * from emp where job = 'SALESMAN' and deptno = 30 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
```

2) List all the salesmen in dept number 30 and having salary greater than 1500

```
SQL> select * from emp
  2  where job = 'SALESMAN'
  3  and deptno = 30
  4  and sal > 1500 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
```

3) List all the employees whose name starts with ‘s’ or ‘a’

```
SQL> select * from emp
  2  where ename like 'S%' or ename like 'A%' ;

     EMPNO 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
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7876 ADAMS      CLERK            7788 23-MAY-87       1100                    20
```

4) List all the employees except those who are working in dept 10 & 20.

```
SQL> select * from emp
  2  where deptno not in (10,20) ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7900 JAMES      CLERK            7698 03-DEC-81        950                    30

6 rows selected.
```

5) List the employees whose name does not start with ‘S’

```
SQL> select * from emp
  2  where ename not like 'S%' ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      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 09-JUN-81       2450                    10
      7839 KING       PRESIDENT             17-NOV-81       5000                    10
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK            7788 23-MAY-87       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

12 rows selected.
```

6) List all the employees who are having reporting managers in dept 10

```
SQL> select * from emp
  2  where mgr is not null
  3  and deptno = 10 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10
```

ASSIGNMENT

1) List the employees who are not working as managers and clerks in dept 10 and 20 with a salary in the range of 1000 to 3000

```
SQL> select * from emp
  2  where job not in ('MANAGER','CLERK')
  3  and deptno in (10,20)
  4  and sal between 1000 and 3000;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
```

2) List the employees whose salary not in the range of 1000 to 2000 in dept 10,20,30 except all salesmen

```
SQL> select * from emp where
  2  sal not between 1000 and 2000
  3  and deptno in (10,20,30)
  4  and job <>'SALESMAN';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20

8 rows selected.
```

3) List the department names which are having letter 'O' in their locations as well as their department names

```
SQL> select * from dept
  2  where loc like '%_O_%' and
  3  dname like '%_O_%';

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        40 OPERATIONS     BOSTON
```

SORTING

It arranges the data either in ascending / descending order
Ascending – ASC / Descending – DESC
We can sort the data using **ORDER BY**

By default, the data is always arranged in ASC order

For ex – 1) Arrange all the employees by their salary

```
SQL> select * from emp
  2  order by sal;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
```

```
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10

14 rows selected.
```

2) Arrange all the employees by their salary in the descending order

```
SQL> select * from emp
  2  order by sal desc;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20

14 rows selected.
```

3) Arrange ename, sal, job, empno and sort by descending order of salary

```
SQL> select ename, sal, job, empno
  2  from emp
  3  order by 2 desc ;

ENAME             SAL JOB              EMPNO
---------- ---------- --------- ----------
KING             5000 PRESIDENT        7839
FORD             3000 ANALYST          7902
SCOTT            3000 ANALYST          7788
JONES            2975 MANAGER          7566
BLAKE            2850 MANAGER          7698
CLARK            2450 MANAGER          7782
ALLEN            1600 SALESMAN         7499
TURNER           1500 SALESMAN         7844
MILLER           1300 CLERK            7934
WARD             1250 SALESMAN         7521
MARTIN           1250 SALESMAN         7654
ADAMS            1100 CLERK            7876
JAMES             950 CLERK            7900
SMITH             800 CLERK            7369

14 rows selected.
```

In the above query we have – **order by 2** – thus it arranges only the 2nd column 'salary' in the descending order.
Thus to arrange the specific columns in order – we must have to specify the column number.

*NOTE :-* **ORDER BY** should be used always as the last statement in the SQL query.

## Selecting DISTINCT VALUES

```
SQL> select distinct deptno
  2  from emp ;

    DEPTNO
----------
        30
        20
        10
```

The above query arranges all the distinct values of department number.

*CHAPTER 4*

# GROUP functions and Grouping

We have **5 GROUP** functions,

1) Sum
2) Max
3) Min
4) Avg
5) Count

**Sum –** returns total value
**Max** – returns maximum value
**Min** – returns minimum value
**Avg** – returns average value
**Count –** returns number of records

**Ex – 1)** display the maximum salary, minimum salary and total salary from employee

```
SQL> select max(sal), min(sal), sum(sal) from emp;

  MAX(SAL)   MIN(SAL)   SUM(SAL)
---------- ---------- ----------
      5000        800      29025
```

*To give aliases for the columns :-*

```
SQL> select max(sal) "high",
  2  min(sal) "low",
  3  sum(sal) "total"
  4  from emp ;

      high        low      total
---------- ---------- ----------
      5000        800      29025
```

3) The below query gives the total number of employees

```
SQL> select count(*), count(empno)
  2  from emp ;

  COUNT(*) COUNT(EMPNO)
---------- ------------
        14           14
```

4) The below query gives the number of employees who have commission

```
SQL> select count(*), count(comm)
  2  from emp ;

  COUNT(*) COUNT(COMM)
---------- ------------
        14            4
```

5) List the number of employees in department 30

```
SQL> select count(*) from emp
  2  where deptno = 30 ;

  COUNT(*)
----------
         6
```

ASSIGMENT

1) Display the total salary in department 30

```
SQL> select sum(sal) "total" from emp
  2  where deptno = 30;

     total
----------
      9400
```

2) List the number of clerks in department 20

```
SQL> select count(*) from emp
  2  where deptno = 20
  3  and job = 'CLERK' ;

  COUNT(*)
----------
         2
```

3) List the highest and lowest salary earned by salesmen

```
SQL> select max(sal), min(sal) from emp
  2  where job = 'SALESMAN';

  MAX(SAL)   MIN(SAL)
---------- ----------
      1600       1250
```

GROUPING

It is the process of computing the aggregates by segregating based on one or more columns. Grouping is done by using **'group by'** clause.

For ex – 1) Display the total salary of all departments

```
SQL> select deptno, sum(sal)
  2  from emp
  3  group by deptno ;

    DEPTNO   SUM(SAL)
---------- ----------
        30       9400
        20      10875
        10       8750
```

2) Display the maximum of each job

```
SQL> select job, max(sal)
  2  from emp
  3  group by job ;

JOB         MAX(SAL)
--------- ----------
CLERK           1300
SALESMAN        1600
PRESIDENT       5000
MANAGER         2975
ANALYST         3000
```

## HAVING

**'Having'** is used to filter the grouped data.
**'Where'** is used to filter the non grouped data.

**'Having'** should be used after **group by** clause
**'Where'** should be used before **group by** clause

For ex – 1) Display job-wise highest salary only if the highest salary is more than Rs1500

```
SQL> select job, max(sal)
  2  from emp
  3  group by job
  4  having max(sal) > 1500 ;

JOB         MAX(SAL)
--------- ----------
SALESMAN        1600
PRESIDENT       5000
MANAGER         2975
ANALYST         3000
```

2) Display job-wise highest salary only if the highest salary is more than 1500 excluding department 30. Sort the data based on highest salary in the ascending order.

```
SQL> select job, max(sal)
  2  from emp
  3  where deptno <>30
  4  group by job
  5  having max(sal) >1500
  6  order by 2 ;

JOB         MAX(SAL)
--------- ----------
MANAGER         2975
ANALYST         3000
PRESIDENT       5000
```

## RESTRICTIONS ON GROUPING

- we can select only the columns that are part of '**group by**' statement
If we try selecting other columns, we will get an error as shown below,

```
SQL> select deptno, job, sum(sal), sum(comm)
  2  from emp
  3  group by deptno ;
select deptno, job, sum(sal), sum(comm)
               *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

The above query is an error because '**job**' is there in the **select** query but not in the **group by** query.
If it is enclosed in any of the **group functions like sum(sal)** etc – then it is not an error. But whatever table is included in the **select** query must also be included in the **group by** query.

The above problem can be overcome with the following query as shown below,

```
SQL>  select deptno, job, sum(sal), sum(comm)
  2   from emp
  3   group by deptno, job ;

    DEPTNO JOB         SUM(SAL)  SUM(COMM)
---------- --------- ---------- ----------
        20 CLERK            1900
        30 SALESMAN         5600       2200
        20 MANAGER          2975
        30 CLERK             950
        10 PRESIDENT        5000
        30 MANAGER          2850
        10 CLERK            1300
        10 MANAGER          2450
        20 ANALYST          6000

9 rows selected.
```

The below query is also correct to rectify the above error,

```
  1  select deptno, sum(sal), sum(comm)
  2  from emp
  3  group by deptno, job
  4* order by deptno
SQL> /

    DEPTNO   SUM(SAL)  SUM(COMM)
---------- ---------- ----------
        10       1300
        10       2450
        10       5000
        20       6000
        20       1900
        20       2975
        30        950
        30       2850
        30       5600       2200

9 rows selected.
```

Whatever is there in the **select** statement must be there in the **group by** statement. But, whatever is there in the **group by** statement need not be present in the **select** statement. This is shown in the above two corrected queries.

ASSIGNMENT

1) Display the department numbers along with the number of employees in it

```
SQL> select deptno, count(*)
  2  from emp
  3  group by deptno
  4  order by deptno ;

    DEPTNO   COUNT(*)
---------- ----------
        10          3
        20          5
        30          6
```

2) Display the department numbers which are having more than 4 employees in them

```
SQL> select deptno from emp
  2  group by deptno
  3  having count(*) >4
  4  order by deptno ;

    DEPTNO
----------
        20
        30
```

3) Display the maximum salary for each of the job excluding all the employees whose name ends with 'S'

```
SQL>  select ename, job, min(sal)
  2   from emp
  3   where ename not like '%S'
  4   group by ename, job
  5  order by 3 ;

ENAME      JOB         MIN(SAL)
---------- --------- ----------
SMITH      CLERK            800
MARTIN     SALESMAN        1250
WARD       SALESMAN        1250
MILLER     CLERK           1300
TURNER     SALESMAN        1500
ALLEN      SALESMAN        1600
CLARK      MANAGER         2450
BLAKE      MANAGER         2850
FORD       ANALYST         3000
SCOTT      ANALYST         3000
KING       PRESIDENT       5000

11 rows selected.
```

4) Display the department numbers which are having more than 9000 as their departmental total salary

```
SQL>  select deptno, sum(sal)
  2   from emp
  3   group by deptno
  4   having sum(sal) >9000
  5  order by 1 ;

    DEPTNO   SUM(SAL)
---------- ----------
        20      10875
        30       9400
```

**NOTE :**
To clear the screen, the command used is,
**cl scr ;**

if it is a large query and we cannot type it repeatedly, then type in – **SQL > ed ;**
when we type **ed ;** - we get the notepad – after making the necessary changes – then click on the **'x'** i.e, the close button at the top right corner – then click on **yes** when a dialog box asking whether to overwrite the file comes – after this it comes to the oracle screen – in the next line , enter '**/** ' and hit on **enter** button – another way of ending the query is by typing ' **/** ' in the next line of the query – this indicates the end of the query.

## *CHAPTER 5*

# STATEMENTS

**Statements –** they help us to create the table and insert the data.

There are 3 types of statements,

- **DDL** – Data Definition Language – the various commands in DDL are :- Create, Drop, Truncate, Alter, Rename
- **DML** – Data Manipulation Language – the various commands in DML are :- Insert, Update, Delete
- **TCL** – Transaction Control Language – the various commands in TCL are :- Rollback, Commit, Savepoint

**CREATE** – It creates the table.

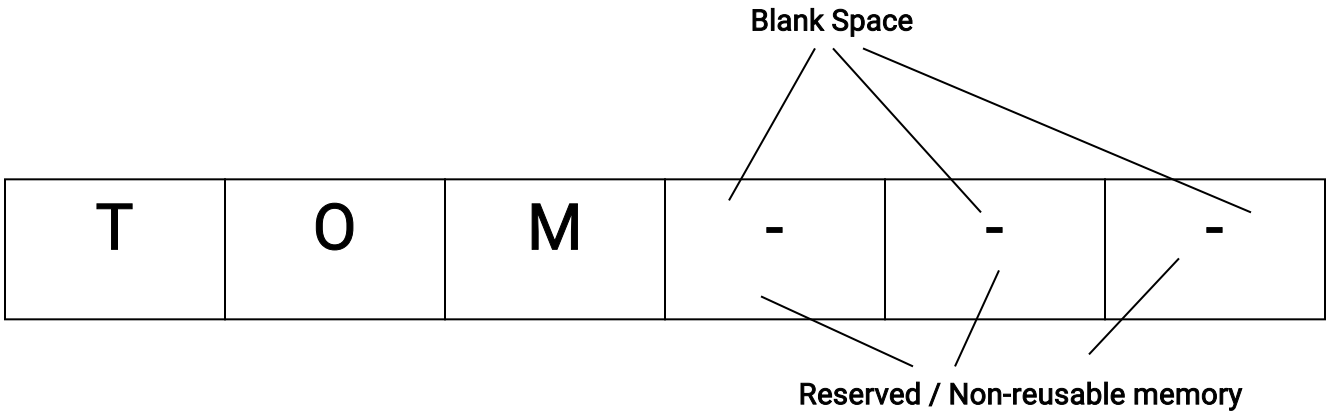
Before we study the **Create** command, let us first study the some of the basic **datatypes** we use in SQL.

**1) CHAR** :-
It stores the fixed length character data.
It can store the alphanumeric data (i.e, numbers and characters).

**2) VARCHAR**
It stores the variable length character data
It can store alphanumeric data.

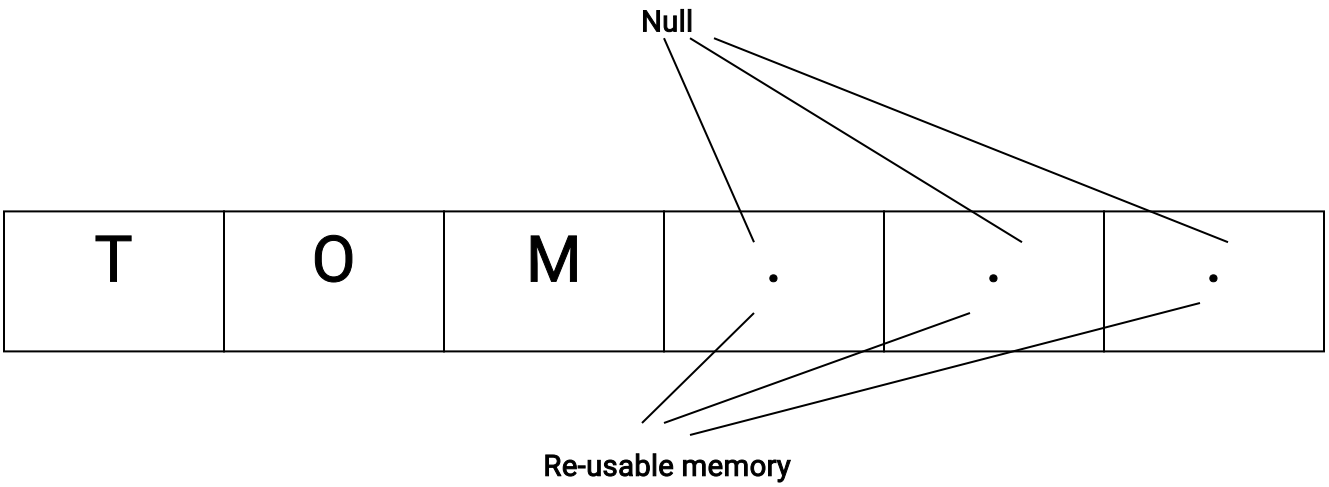
**Difference between CHAR & VARCHAR**
Let us consider an example as shown below to explain the difference.

***Name char (6) ;***
Here we are defining **name** which is of 6characters in length.
Now, let us store '*Tom*' in the name field. Let us understand how the memory is allocated for this,




When we declare anything of type **char**, the memory is allocated as of the size given and its fixed length – hence it cannot be altered.
Now, when we give ***tom***, it allocates 6 bytes for **name char** – only the 1st 3bytes are used to store **Tom** – the rest becomes waste as it is a blank space and it is reserved memory.
The **length(name) = 6**.

**Name varchar (6) ;**
Here we are defining **name** which is of 6characters in length.
Now, let us store '*Tom*' in the name field. Let us understand how the memory is allocated for this,




When we declare anything of type **varchar**, the memory is allocated as shown above and it is variable length
When we give ***tom***, it allocates 6bytes for **name varchar** – only the 1st 3bytes are used to store **tom** – the remaining 3 fields becomes **null**. As we know the property of **null** – null does not occupy any memory space **– thus the memory is not wasted here.**

The **length(name) = 3**.

**Another difference is** : -
In **char**, maximum value we can store is 2000 characters
In **varchar**, maximum value we can store is 4000 characters.

**3) NUMBER**
- it stores numeric data.
**For ex – 1) sal number(4) ;**
Here the maximum possible value is 9999.

**2) sal number (6, 2) ;**
Here, 2 – scale (total number of decimal places)
6 – precision (total number of digits including decimal places)
Maximum value is 9999.99
**sal number (4, 3) ;**
maximum value is 9.999
**sal number (2, 2)**
maximum value is .99

**4) DATE**
- it stores date and time
- no need to specify any length for this type.

**For ex,** SQL > order_dt DATE ;

Date is always displayed in the default format :- **dd – month – yy**

**NOTE :-**
**varchar2** – from 10g, varchar & varchar2 are the same.
Earlier, varchar was supporting upto 2000 characters and varchar2 was supporting upto 4000 characters.

**5) BLOB**
Stands for – Binary Large Object
It stores binary data (images, movies, music files) within the database. It stores upto 4GB.

**6) CLOB**
Stands for – Character Large Object
It stores plain character data like **varchar** field upto 4GB.

**Create the following tables**

| PRODUCTS |
| --- |
| ProdID ( PK ) |
| ProdName ( Not Null ) |
| Qty ( Chk > 0 ) |

| Description |
|---|

| ORDERS |
|---|
| ProdID ( FK from products ) |
| OrderID ( PK ) |
| Qty_sold ( chk > 0 ) |
| Price |
| Order_Date |

```
SQL> CREATE TABLE products
  2  (
  3   prodid  NUMBER(4) PRIMARY KEY ,
  4   prodname  VARCHAR(10) NOT NULL ,
  5   qty  NUMBER(3) CHECK (qty > 0) ,
  6   description VARCHAR(20)
  7  ) ;

Table created.
```

We can see that the table has been created.
Now, let us verify if the table has really been created and also the description of the table,

```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
```

The new table **products** has been added to the database.

```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                                NUMBER(3)
 DESCRIPTION                                        VARCHAR2(20)
```

Thus, we get the description of the table **products**.

```
SQL> CREATE TABLE orders
  2  (
  3   prodid  NUMBER(4) REFERENCES  products (prodid) ,
  4   orderid  NUMBER(4) PRIMARY KEY ,
  5   qty_sold NUMBER(3) CHECK (qty_sold > 0),
  6   price NUMBER(8, 2) ,
  7   order_dt DATE
  8  ) ;

Table created.
```

The new table **orders** has been created. We can see from the above query how to reference a child table to the parent table using the **references** keyword.

```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE

6 rows selected.
```

Thus we can verify that **orders** table has ben created and added to the database.

```
SQL> desc orders ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                             NUMBER(4)
 ORDERID                                   NOT NULL NUMBER(4)
 QTY_SOLD                                           NUMBER(3)
 PRICE                                              NUMBER(8,2)
 ORDER_DT                                           DATE
```

Thus, we get the description of the **orders** table.

Creating a table from another table :-

Now, we will see how to create a table from another table – i.e, it duplicates all the records and the characterstics of another table.

The SQL query for it is as follows,

```
SQL> CREATE TABLE temp
  2  AS
  3  select * from dept ;

Table created.
```

Thus we can see that we have created another table **temp** from the table **dept**.
We can verify it as shown below,

```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP                           TABLE

7 rows selected.
```

Thus, we can see that the **table temp** has been created.

```
SQL> desc temp ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

Thus, we can see that the table **temp** has copied the structure of the table **dept**. Here, we must observe that **temp** copies all the columns, rows and NOT NULL constraints only from the table **dept**. It never copies PK, FK, Check constraints.
**Thus, when in the interview somebody asks you "I have a table which has about 1million records. How do I duplicate it into another table without using Insert keyword and without inserting it individually all the records into the duplicated table ?**
**Answer is - Use the above query of creating a table from another table and explain it.**

```
SQL> select * from temp ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Thus, from the above query – we can see that all the records of the table **dept** has been copied into the table **temp**.

## TRUNCATE

It removes all the data permanently, but the structure of the table remains as it is.
**Ex – SQL > TRUNCATE TABLE test ;**

## DROP

It removes both data and the structure of the table permanently from the database.
**Ex – SQL > DROP TABLE test ;**

Let us understand the difference between **drop & truncate** using the below shown example,

```
SQL> CREATE TABLE test1
  2  AS
  3  select * from dept ;

Table created.
```

```
SQL> CREATE TABLE test2
  2  AS
  3  select * from dept ;

Table created.
```

Let us create 2 tables Test1 and Test2 as shown above.

```
SQL> desc test1 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

```
SQL> select * from test1 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

The above shows the description of the table test1.

```
SQL> desc test2 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

```
SQL> select * from test2 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

The above gives the description of the table Test2.

Now, let us use the **Truncate query on Test1** and **Drop query on Test2** and see the difference.

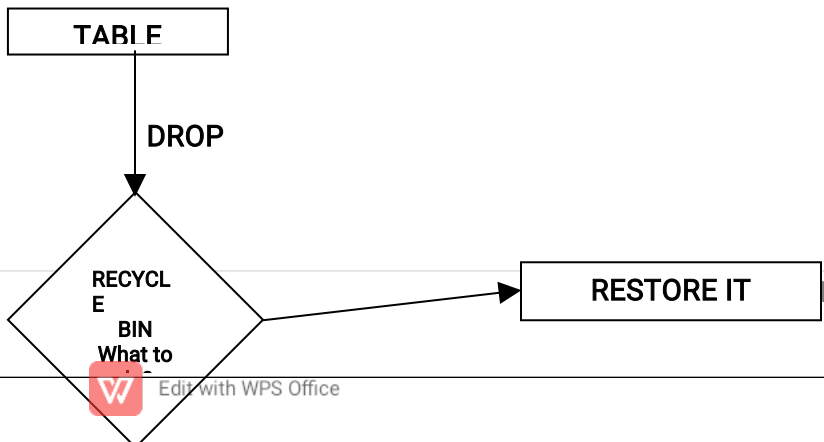
```
SQL> truncate table test1 ;

Table truncated.

SQL> select * from test1 ;

no rows selected

SQL> desc test1 ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                             NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

The above 3 queries show that – 1st query has the table test1 truncated.
2nd query – it shows **no rows selected** – thus only the records from the table has been removed. 3rd query – it shows that the structure of the table is still present. Only the records will be removed.
Thus, this **explains the truncate query.**

```
SQL> drop table test2 ;

Table dropped.

SQL> select * from test2 ;
select * from test2
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> desc test2 ;
ERROR:
ORA-04043: object test2  does not exist
```

Thus from the above queries we can explain how **drop** works. 1st query – it drops the table. Thus – the entire structure and records of the table are dropped.
2nd and 3rd query – since, there is no table – **select & desc** query for **test2** will throw an error.
Thus, this **explains the drop query.**
Hence, we have seen the difference between **drop & truncate** query.

<u>10g Recycle Bin</u>



## FLASHBACK

## PURGE

The functionality of Recycle Bin was introduced in Oracle 10G version only. Thus even though the table has been dropped, we can still restore it using **flashback command** or we can permanently remove it using the **purge** command.
This concept of Recycle bin was not there in the earlier versions of Oracle.

### RENAME

It renames a table.
**For ex,** let us see the query of how we do this renaming a table.

```
SQL> CREATE TABLE temp
  2  AS
  3  select * from dept ;

Table created.

SQL> select * from temp ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP                           TABLE

7 rows selected.
```

In the above 3queries – we have created a table **temp** which copies table **dept** – we see the records of the table temp – and also check if the table has really been created.

Now let us **rename temp to temp23** as shown below,

```
SQL> RENAME temp TO temp23 ;

Table renamed.
```

The above query is used to rename a table.
Now let us verify the contents of the table and check if it has really been modified,
See next page,

```
SQL> select * from tab ;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEPT                           TABLE
EMP                            TABLE
BONUS                          TABLE
SALGRADE                       TABLE
PRODUCTS                       TABLE
ORDERS                         TABLE
TEMP23                         TABLE

7 rows selected.

SQL> select * from temp23 ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Thus the table has been renamed and its contents are verified.

## ALTER

- this query alters / changes the structure of the table (i.e, - adding columns, removing columns, renaming columns etc ).
Now let us **alter** the table **products** (which we have created earlier).
**1) Let us add a new column *'model_no'* to the table.**

```
SQL> ALTER TABLE products
  2  ADD model_no VARCHAR(10) NOT NULL ;

Table altered.
```

Thus, a new column has been added. Lets verify it with the query shown below,

```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                               NUMBER(3)
 DESCRIPTION                                       VARCHAR2(20)
 MODEL_NO                                  NOT NULL VARCHAR2(10)
```

2) Now let us drop the column model_no from products.

```
SQL> ALTER TABLE products
  2  DROP COLUMN model_no ;

Table altered.
```

Thus, the column has been dropped.

```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY                                               NUMBER(3)
 DESCRIPTION                                       VARCHAR2(20)
```

Thus, we can see from the description of the table – the column **model_no** has been dropped.

3) Let us rename the column *qty* to *qty_available*.

```
SQL> ALTER TABLE products
  2  RENAME column qty to qty_available ;

Table altered.
```

Let us verify if it has been renamed,

```
SQL> desc products ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODID                                    NOT NULL NUMBER(4)
 PRODNAME                                  NOT NULL VARCHAR2(10)
 QTY_AVAILABLE                                     NUMBER(3)
 DESCRIPTION                                       VARCHAR2(20)
```

NOTE : *SELECT* is neither DML nor DDL. It does not belong to any group because it does not alter anything, it just displays the data as required by the user.

# DML

## INSERT

It inserts a record to a table.
Let us observe how it is done,

```
SQL>  INSERT INTO products
  2   values (1001, 'CAMERA' , 10, 'Digital') ;

1 row created.

SQL> INSERT INTO products
  2  values (1002, 'Laptop', 23, 'Dell') ;

1 row created.
```

This is how we insert values into a table. All characters and alpha-numeric characters(ex – 10023sdf78) must be enclosed in single quotes (‘ ‘ ) and each value must be separated by comma. Also we must be careful in entering the data without violating the primary key, foreign key , unique constraints.

Now let us see the table in which the data in has been inserted,

```
SQL> select * from products ;

    PRODID PRODNAME   QTY_AVAILABLE DESCRIPTION
---------- ---------- ------------- --------------------
      1001 CAMERA                10 Digital
      1002 Laptop                23 Dell
```

Now, let us insert data into the table **orders** in which a foreign key is referencing primary key,

```
SQL> INSERT INTO orders
  2  values (1001, 9001, 2, 9867.1, sysdate ) ;

1 row created.
```

Here, we see that 1001 is the same prodid as of the earlier table. Sysdate – it displays the current date set in the system .

```
SQL> INSERT INTO orders
  2  values (1002, 9023, 2, 98756.23, ' 02 - Oct - 2010 ' ) ;

1 row created.
```

Now, let us see the table,

```
SQL> select * from orders ;

    PRODID    ORDERID   QTY_SOLD      PRICE ORDER_DT
---------- ---------- ---------- ---------- ---------
      1001       9001          2     9867.1 06-APR-11
      1002       9023          2   98756.23 02-OCT-10
```

Another way of inserting data into the table is shown below,

```
SQL> INSERT INTO orders (prodid, orderid, qty_sold, price, order_dt)
  2  values (1002, 99, 7, 23678.9, '02 - Oct - 1987' ) ;

1 row created.
```

Now, let us see the table,

```
SQL> select * from orders ;

    PRODID    ORDERID   QTY_SOLD      PRICE ORDER_DT
---------- ---------- ---------- ---------- ---------
      1001       9001          2     9867.1 06-APR-11
      1002       9023          2   98756.23 02-OCT-10
      1002         99          7    23678.9 02-OCT-87
```

UPDATE :-

It updates one or more records.

**For ex – 1)** Let us update salary by increasing it by Rs200 and also give commission of Rs100 where empno = 7369.

```
SQL> select * from emp ;

     EMPNO 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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       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

14 rows selected.
```

Now, let us **update** the said record as shown below,

```
SQL> update emp set sal = sal + 200, comm = 100 where empno = 7369 ;

1 row updated.
```

Let us verify if the record has been updated,

```
SQL> select * from emp ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80       1000        100         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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       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

14 rows selected.
```

Thus, the record(empno – 7369) has been updated.

**2) Increase all salary by 10%**

```
SQL> update emp set sal = sal + sal * 0.1 ;

14 rows updated.
```

Let us verify it,

```
SQL> select * from emp ;
```

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 1100 | 100 | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1760 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1375 | 500 | 30 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 3272.5 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1375 | 1400 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 3135 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2695 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3300 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5500 | | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1650 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1210 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 1045 | | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3300 | | 20 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1430 | | 10 |

```
14 rows selected.
```

## DELETE

It deletes one / some / all the records.
Let us create a table test from table emp – and see how to delete 1 record and how to delete all records from it,

```
SQL> select * from test ;
```

| EMPNO | 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 | 09-JUN-81 | 2450 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 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 |

```
14 rows selected.
```

Thus, we have created the table test.

```
SQL> delete from test where empno = 7934 ;

1 row deleted.
```

Thus 1 row, 'miller' has been deleted.

```
SQL> select * from test ;

     EMPNO 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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20

13 rows selected.
```

Thus, the deletion has been confirmed.

# TCL

Any DML change on a table is not a permanent one.
We need to save the DML changes in order to make it permanent
We can also undo (ignore) the same DML changes on a table.

The DDL changes cannot be undone as they are implicitly saved.

## ROLLBACK

It undoes the DML changes performed on a table.
Let us see in the below example how **rollback** works,

```
SQL> delete from emp ;

14 rows deleted.

SQL> select * from emp ;

no rows selected
```

Let us delete the employee table. When we perform **select** operation on emp, we can see that all the rows have been deleted.

We now perform the **rollback** operation,

```
SQL> rollback ;

Rollback complete.
```

Now let us perform the **select** operation,

```
SQL> select * from emp ;

     EMPNO 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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       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

14 rows selected.
```

Thus performing the **rollback** operation, we can retrieve all the records which had been deleted.

## COMMIT

It saves the DML changes permanently to the database.

**Committing after rollback & vice versa will not have any effect**
Let us explain the above statement with an example,

```
SQL> select * from test ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> delete from test ;

4 rows deleted.

SQL> select * from test ;

no rows selected

SQL> rollback ;

Rollback complete.

SQL> commit  ;

Commit complete.

SQL> select * from test ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

We can see that **commit** has no effect after **rollback** operation.

```
SQL> select * from test ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> delete from test ;

4 rows deleted.

SQL> commit ;

Commit complete.

SQL> rollback ;

Rollback complete.

SQL> select * from test ;

no rows selected
```

Thus, from above – we can see that **rollback** has no effect after **commit** operation.

During an abnormal exit – i.e, shutdown or if the SQL window is closed by mouse click – then all the DML's will be rolled back automatically.

During a normal exit – **exit ;** - all the DML's will be auto-committed – and there will be no rollback.

**Ex – 1)** INSERT
UPDATE
ALTER
DELETE
ROLLBACK

When we perform the following operations in the same order for a table – then INSERT, UPDATE will be committed – because ALTER is a DDL – and thus all the DML's above it will also be committed – because DDL operations cannot be undone.
Here – only DELETE will be rolled back because it's a DML.

**2)** INSERT
UPDATE
DELETE
ROLLBACK

Here, all are rolled back.

**SAVEPOINT** :

It is like a pointer (break-point) till where a DML will be rolled back.
**Ex :-**
Insert …

Save point x ;
Update …
Delete ..

Rollback to x ;
…
…

Here, only DELETE & UPDATE are rolled back.
INSERT is neither rolled back nor committed.

**Assignments**
**1) Create the following tables**

a) Table name :- STUDENTS
regno (PK)
name (NN)
semester
DOB

Phone

b) Table name :- BOOKS
bookno (PK)
bname
author

c) Table name :- LIBRARY
regno (FK from students)
bookno (FK from books)
DOI –date of issue
DOR – date of return

**2) Insert 5 records to each of these tables**

**3) Differentiate between,**
**a) Delete and Truncate**
**b) Truncate and Drop**
**c) Char and Varchar**
**d) Drop and Delete**

Delete and Truncate

a) Delete – deletes whichever records we want to delete from the table
Truncate – deletes all the records whether we want it or not

b) Delete – can be undone
Truncate – cannot be undone.

NOTE – The Primary Key created using more than 1 column is called as ***composite primary key***.
Ex – **alter table lib**
**Add primary key (regno, bookno, DOI)** ;

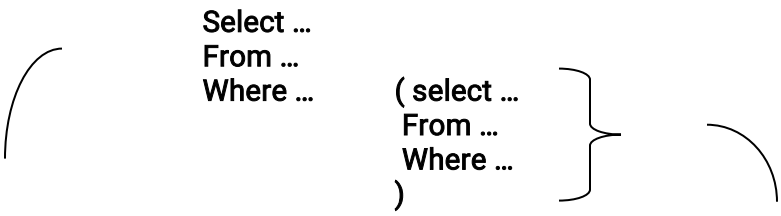
---

*CHAPTER 6*

# SUB - QUERIES

A sub-query is also called as a nested query.

**Syntax of a sub-query**




OUTER QUERY INNER QUERY

Here, the **inner query** will be executed first.
The output of **inner query** is passed as input to the **outer query.**

To write a sub-query, atleast 1 common column should be existing between the tables.

For ex :-

1) List the employees working in 'Research' department.

```
SQL> select * from emp
  2  where deptno = (select deptno
  3                  from dept
  4                  where dname = 'RESEARCH'
  5                  ) ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7566 JONES      MANAGER          7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7876 ADAMS      CLERK            7788 23-MAY-87       1100                    20
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20
```

2) List the department names that are having analysts

```
SQL>  select * from dept
  2   where deptno IN (select deptno
  3                    from emp
  4                    where job = 'ANALYST'
  5                   ) ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        20 RESEARCH       DALLAS
```

3) List the employees in Research and Sales department

```
SQL>  select * from emp
  2   where deptno IN (select deptno
  3                    from dept
  4                    where dname IN ('RESEARCH','SALES')
  5                    )
  6  order by deptno ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30

11 rows selected.
```

Assignment

1) List the department names which are having salesmen in it.

```
SQL> select * from dept
  2  where deptno in (select deptno from emp
  3                    where job = 'SALESMAN'
  4                   ) ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        30 SALES          CHICAGO
```

2) Display the employees whose location is having atleast one ‘O’ in it.

```
SQL>  select * from emp
  2   where deptno in (select deptno from
  3                    dept where loc like '%O%')
  4  order by deptno ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30

9 rows selected.
```

3) List the department names that are having atleast 1 employee in it.

```
SQL>  select dname from dept
  2   where deptno IN (select deptno from emp
  3                    group by deptno
  4                    having count(*) >0 ) ;

DNAME
--------------
SALES
RESEARCH
ACCOUNTING
```

4) List the department names that are having atleast 4 employees in it

```
SQL>  select dname from dept
  2   where deptno in (select deptno from emp
  3                    group by deptno
  4                    having count(*) >=4);

DNAME
--------------
SALES
RESEARCH
```

5) Display the department names which are having atleast 2clerks in it

```
SQL>  select dname from dept
  2   where deptno in (select deptno from emp
  3                    where job = 'CLERK'
  4                    group by deptno
  5                    having count('CLERK') =2 ) ;

DNAME
--------------
RESEARCH
```

6) Display the 2nd maximum salary

```
SQL> select max(sal) from emp
  2  where sal < (select max(sal) from emp) ;

  MAX(SAL)
----------
      3000
```

7) Display the 3rd maximum salary

```
SQL> select max(sal) from emp
  2  where sal < (select max(sal) from emp
  3  where sal < (select max(sal) from emp) ) ;

  MAX(SAL)
----------
      2975
```

8) Display the 4th least salary

```
SQL> select min(sal) from emp
  2  where sal > (select min(sal) from emp
  3  where sal > (select min(sal) from emp
  4  where sal > (select min(sal) from emp ) ) ) ;

  MIN(SAL)
----------
      1250
```

This method is not efficient to find the maximum and minimum salary. The limit is 32. This is not efficient if you want to find the 100th maximum salary.
**We can have upto 32 levels of sub-queries only.**

9) List the department names that are having no employees at all

```
SQL> select * from dept
  2  where deptno not in (select deptno from emp) ;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        40 OPERATIONS     BOSTON
```

---

*CHAPTER 7*

JOIN

**Joins** are used when we need to fetch the data from multiple tables

**Types of JOIN(s)**

- Cartesian Join (product)
- Inner (Equi) Join
- Outer Join - Left Outer Join, Right Outer Join, Full Outer Join
- Self Join

**CARTESIAN JOIN**

- It is based on Cartesian product theory.

> **Cartesian Product Theory** in Mathematics states that :-
> Let there be two sets – A {1, 2, 3} & B {4, 5}
> Thus the Cartesian product (A*B) will be,
> A * B = { (1,4), (1,5), (2,4), (2,5), (3,4), (3,5) }
> Thus there are 6 sets – order of A is 3 & order of B is 2. Therefore, 2*3 = 6 is the Cartesian product.

Here, each and every record of the 1st table will combine with each and every record of the 2nd table.
If a table A is having 10 records & B is having 4 records – the Cartesian join will return 10*4 = 40 records.

**For ex,** let us consider the following query
**Display employee name along with the department name**

```
SQL> select A.ename, A.sal, B.dname
  2  from emp A, dept B ;
```

| ENAME | SAL | DNAME |
|---|---|---|
| SMITH | 800 | ACCOUNTING |
| ALLEN | 1600 | ACCOUNTING |
| WARD | 1250 | ACCOUNTING |
| JONES | 2975 | ACCOUNTING |
| MARTIN | 1250 | ACCOUNTING |
| BLAKE | 2850 | ACCOUNTING |
| CLARK | 2450 | ACCOUNTING |
| SCOTT | 3000 | ACCOUNTING |
| KING | 5000 | ACCOUNTING |
| TURNER | 1500 | ACCOUNTING |
| ADAMS | 1100 | ACCOUNTING |
| JAMES | 950 | ACCOUNTING |
| FORD | 3000 | ACCOUNTING |
| MILLER | 1300 | ACCOUNTING |
| SMITH | 800 | RESEARCH |
| ALLEN | 1600 | RESEARCH |
| WARD | 1250 | RESEARCH |
| JONES | 2975 | RESEARCH |
| MARTIN | 1250 | RESEARCH |
| BLAKE | 2850 | RESEARCH |
| CLARK | 2450 | RESEARCH |
| SCOTT | 3000 | RESEARCH |
| KING | 5000 | RESEARCH |
| TURNER | 1500 | RESEARCH |
| ADAMS | 1100 | RESEARCH |
| JAMES | 950 | RESEARCH |
| FORD | 3000 | RESEARCH |
| MILLER | 1300 | RESEARCH |
| SMITH | 800 | SALES |
| ALLEN | 1600 | SALES |
| WARD | 1250 | SALES |
| JONES | 2975 | SALES |
| MARTIN | 1250 | SALES |
| BLAKE | 2850 | SALES |

```
SCOTT              3000 RESEARCH
KING               5000 RESEARCH
TURNER             1500 RESEARCH
ADAMS              1100 RESEARCH
JAMES               950 RESEARCH
FORD               3000 RESEARCH
MILLER             1300 RESEARCH
SMITH               800 SALES
ALLEN              1600 SALES
WARD               1250 SALES
JONES              2975 SALES
MARTIN             1250 SALES
BLAKE              2850 SALES
```

```
ENAME                SAL DNAME
---------- ---------- --------------
CLARK              2450 SALES
SCOTT              3000 SALES
KING               5000 SALES
TURNER             1500 SALES
ADAMS              1100 SALES
JAMES               950 SALES
FORD               3000 SALES
MILLER             1300 SALES
SMITH               800 OPERATIONS
ALLEN              1600 OPERATIONS
WARD               1250 OPERATIONS
JONES              2975 OPERATIONS
MARTIN             1250 OPERATIONS
BLAKE              2850 OPERATIONS
CLARK              2450 OPERATIONS
SCOTT              3000 OPERATIONS
KING               5000 OPERATIONS

ENAME                SAL DNAME
---------- ---------- --------------
TURNER             1500 OPERATIONS
ADAMS              1100 OPERATIONS
JAMES               950 OPERATIONS
FORD               3000 OPERATIONS
MILLER             1300 OPERATIONS

56 rows selected.
```

From above – we can see that the above query returns 56 records – but we are expecting 14 records. This is because each and every record of employee table will be combined with each & every record of department table.
Thus, Cartesian join should not be used in real time scenarios.
The Cartesian join contains both correct and incorrect sets of data. We have to retain the correct ones & eliminate the incorrect ones by using the **inner join**.

## INNER JOIN

Inner join are also called as **equijoins**.
They return the matching records between the tables.
In the real time scenarios, this is the most frequently used Join.

**For ex,** consider the query shown below,

```
Select A.ename, A.sal, B.dname
From emp A, dept B
Where A.deptno = B.deptno          - JOIN condition
And A.sal > 2000                   - FILTER condition
Order by A.sal ;
```

Let us see the output shown below,

```
SQL> Select A.ename, A.sal, B.dname
  2  From emp A, dept B
  3  Where A.deptno = B.deptno
  4  And A.sal > 2000
  5  Order by A.sal ;

ENAME              SAL DNAME
---------- ---------- --------------
CLARK            2450 ACCOUNTING
BLAKE            2850 SALES
JONES            2975 RESEARCH
FORD             3000 RESEARCH
SCOTT            3000 RESEARCH
KING             5000 ACCOUNTING

6 rows selected.
```

JOIN condition is mandatory for removing the Cartesian output.

Let us consider the following 2 scenarios shown below,

**Scenario 1**

| A | | |
|---|---|---|
| P | Q | R |
| | | |

| B | | |
|---|---|---|
| P | S | T |
| | | |

| C | | |
|---|---|---|
| P | X | Y |
| | | |

| We want | | | |
|---|---|---|---|
| P | Q | S | X |
| | | | |

**The SQL query will be,**

**Select** A.P, A.Q, B.S, C.X
From A, B, C
Where A.P = B.P
And A.P = C.P

} **Number of joins = 2**

| Therefore, Number of JOINS = Number of tables - 1 |
|---|

**Scenario 2**

| A | | |
|---|---|---|
| P | Q | R |
| | | |

| B | | | |
|---|---|---|---|
| P | Q | S | T |
| | | | |

| C | | |
|---|---|---|
| P | X | Y |
| | | |

| We want | | | | |
|---|---|---|---|---|
| P | Q | R | S | X |
| | | | | |

The **SQL query is ,**

```
Select A.P, A.Q, A.R, B.S, C.X
From A, B, C
Where A.P = B.P
And A.Q = B.Q
And A.P = C.P ;
```

→ **Number of Joins = 3**

**Therefore, Number of JOINS = Number of common columns**

If there are no common columns, then reject it saying that the two tables can be joined.

But there are some cases – where the 2 columns will be same but having different column names.
**For ex –** customerid & cid

**Display employee name, his job, his dname and his location for all the managers living in New York or Chicago**

```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.job = 'MANAGER'
  5  and B.loc in ('NEW YORK', 'CHICAGO') ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
BLAKE      MANAGER   SALES          CHICAGO
CLARK      MANAGER   ACCOUNTING     NEW YORK
```

**ANSI style JOINS**
This was introduced from Oracle 9i.

It is another way of writing inner joins with a few modifications.

```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A join dept B
  3  on A.deptno = B.deptno
  4  and A.job = 'MANAGER'
  5  and B.loc in ('NEW YORK', 'CHICAGO') ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
BLAKE      MANAGER   SALES          CHICAGO
CLARK      MANAGER   ACCOUNTING     NEW YORK
```

Thus we, can see the changes ,

- In the 2nd line - ,(comma) has been replaced by the word 'join'
- In the 3rd line – 'where' has been replaced with 'on'

Assignment

1) Display employee name and his department name for the employees whose name starts with 'S'

```
SQL> select A.ename, B.dname
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.ename not like 'S%' ;

ENAME      DNAME
---------- --------------
ALLEN      SALES
WARD       SALES
JONES      RESEARCH
MARTIN     SALES
BLAKE      SALES
CLARK      ACCOUNTING
KING       ACCOUNTING
TURNER     SALES
ADAMS      RESEARCH
JAMES      SALES
FORD       RESEARCH
MILLER     ACCOUNTING

12 rows selected.
```

OUTER JOIN

It returns both matching and non-matching records

Outer join = inner join + non-matching records

Non-matching records means data present in one table, but absent in another table w.r.to common columns.

**For ex,** 40 is there in deptno of dept table, but not there in deptno of emp table.

Display all the department names irrespective of any employee working in it or not. If an employee is working – display his name.

*Using right join*

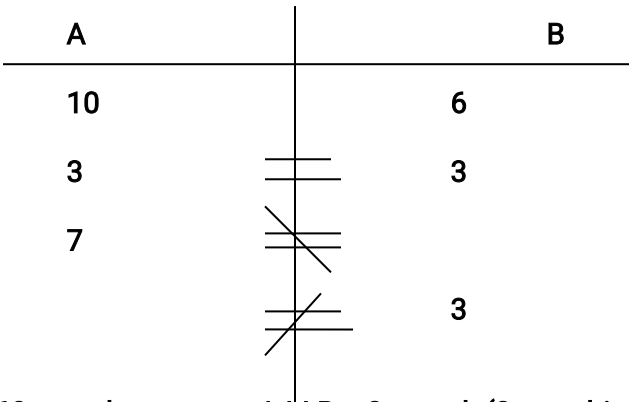
```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A right join dept B
  3  on A.deptno = B.deptno ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
CLARK      MANAGER   ACCOUNTING     NEW YORK
KING       PRESIDENT ACCOUNTING     NEW YORK
MILLER     CLERK     ACCOUNTING     NEW YORK
JONES      MANAGER   RESEARCH       DALLAS
FORD       ANALYST   RESEARCH       DALLAS
ADAMS      CLERK     RESEARCH       DALLAS
SMITH      CLERK     RESEARCH       DALLAS
SCOTT      ANALYST   RESEARCH       DALLAS
WARD       SALESMAN  SALES          CHICAGO
TURNER     SALESMAN  SALES          CHICAGO
ALLEN      SALESMAN  SALES          CHICAGO
JAMES      CLERK     SALES          CHICAGO
BLAKE      MANAGER   SALES          CHICAGO
MARTIN     SALESMAN  SALES          CHICAGO
                     OPERATIONS     BOSTON

15 rows selected.
```

*Using left join*

```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from dept B left join emp A
  3  on A.deptno = B.deptno ;
```

*Using full join*

```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from dept B full join emp A
  3  on A.deptno = B.deptno ;
```




A CJ B = 60records A IJ B = 3records(3 matching)
A LJ B = 10records (3matching + 7non matching of A)
A RJ B = 6records (3matching + 3non matching of B)
A FJ B = 13records (3matching of A & B + 7nonmatching of A + 3nonmatching of B)

Assignment

1) Display employee name and his department name for the employees whose name starts with 'S'

```
SQL> select A.ename, B.deptno
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.ename like 'S%' ;

ENAME          DEPTNO
---------- ----------
SMITH              20
SCOTT              20
```

2) Display employee name and his department name who is earning 1st maximum salary

```
SQL> select A.ename, B.dname
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.sal = (select max(sal) from emp) ;

ENAME      DNAME
---------- --------------
KING       ACCOUNTING
```

SELF JOIN

Joining a table to itself is called self join

The **FROM** clause looks like this,

FROM emp A, emp B

Or

FROM emp A join emp B - *ANSI style*

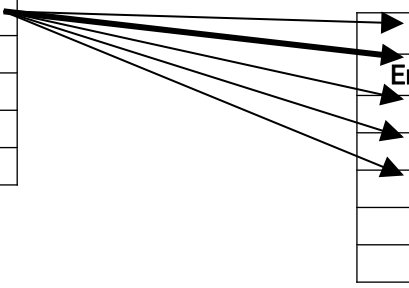
For ex, - Display employee name along with their manager name

```
SQL> select A.ename "EMP",
  2         B.ename "MANAGER"
  3  from emp A, emp B
  4  where A.mgr = B.empno ;

EMP          MANAGER
---------- ----------
SMITH        FORD
ALLEN        BLAKE
WARD         BLAKE
JONES        KING
MARTIN       BLAKE
BLAKE        KING
CLARK        KING
SCOTT        JONES
TURNER       BLAKE
ADAMS        SCOTT
JAMES        BLAKE
FORD         JONES
MILLER       CLARK

13 rows selected.
```

Now, let us see how this i.e the logic (the above query) works,

Emp (A)

| EmpNo | Ename | Mgr |
|---|---|---|
| 101 | Scott | 102 |
| 102 | Blake | 103 |
| 103 | King | - |
| 104 | Smith | 103 |
| 105 | Jones | 104 |



| Emp (B) | | |
|---|---|---|
| EmpNo | Ename | Mgr |
| 101 | Scott | 102 |
| 102 | Blake | 103 |
| 103 | King | - |
| 104 | Smith | 103 |
| 105 | Jones | 104 |

Now, when we give the above query – in Oracle – it starts matching the '**mgr**' column of **emp A** with the '**empno**' of **emp b** – we get two tables because in **self join** – a duplicate of the table required is created.

Now let us consider the **first employee Scott** – it starts the **mgrid** of **Scott** with the **empno** of all the records in **emp B** – when two **ids** match, then the **empno** in **emp B** becomes the **mgr** of the **empno** in **emp A**. Thus, we can see that – **mgr id** 102 is matching with **empno** 102 **Blake** in **emp B**. Therefore, Blake is the manager of Scott.
Similarly we do the same for all the other records of **emp A** and thus find the employees and their respective managers.

**Display the employees who are getting the same salary**

```
SQL> select A.ename, A.sal
  2  from emp A join emp B
  3  on A.sal = B.sal
  4  and A.empno <> B.empno ;

ENAME             SAL
---------- ----------
MARTIN            1250
WARD              1250
FORD              3000
SCOTT             3000
```

**Co – related Queries** :

- → They are special type of sub – queries
- → Here, both outer & inner queries are inter-dependent
- → For each & every record of outer query, the entire inner query will be executed
- → They work on the principles of both **sub – queries & JOIN(s)**.

For ex, **Display the employee who is earning the highest salary**

```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal ) ;
```

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |

Thus, if an outer query column is being accessed inside the inner query, then that query is said to be co-related.

Let us see the logic i.e, how we get the 1st max salary :-

| Emp (A) | | |
|---|---|---|
| **EmpNo** | **Ename** | **Sal** |
| 101 | Scott | 3000 |
| 102 | Blake | 4000 |
| 103 | King | 5000 |
| 104 | Smith | 2000 |
| 105 | Jones | 1000 |

| Emp (B) | | |
|---|---|---|
| **EmpNo** | **Ename** | **Sal** |
| 101 | Scott | 3000 |
| 102 | Blake | 4000 |
| 103 | King | 5000 |
| 104 | Smith | 2000 |
| 105 | Jones | 1000 |

Since co-related queries are a combination of Joins and sub-queries.
It follows the concept of Joins and creates multiple copies of the same table.
Then it takes 1st record i.e, - Blake – sal is 3000. It starts comparing with the sal in the emp table,
3000 = 3000 - count starts from 0 – thus, 0 = 0
3000 < 4000 – thus, 0 ! = 1
3000 < 5000 – thus, 0 ! = 2
3000 > 2000 – thus , 0! = 2
3000 > 1000 – thus, 0 ! = 2 if the condition becomes false, then the count increments by 1. Here 3000 is less than 4000 & 5000, thus 0 ! = 2. Thus , Blake does not have the highest salary.

Similarly, it does for the next records,
Blake – salary of 4000 – but 4000 < 5000 – thus, 0 ! = 1. This is also false.
King – salary of 5000 – it is greater than everything – thus, 0 = 0. Thus, King has the highest salary.
But the query doesn't stop here, it checks for Smith & Jones as well.

Similarly, if we want to find the 2nd maximum salary,
Then in the query, change '0' to '1' & here, the logic is – it compares until it gets 1 = 1.
For 3rd maximum salary – change 0 to 2 and so on – here, the logic is – it compares until it gets 2 = 2.

**For any highest, always put it as '0' in the query.**

**If you want n(th) salary, pass (n-1).**

**In interview – this is a definite question. They will ask you what is co-related queries. And then they'll ask you find, 1st or max or 3rd maximum salary – after you write the query – they will ask you to explain the logic as to how it gets the same – draw the table and explain it to them just as shown above.**

**Assignment**

1) Display the least salary from the employee table.

```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal > B.sal ) ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

2) Display top 3 person's salaries from the employee table.

```
SQL> select * from emp A
  2  where 2 >= (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal ) ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

3) Write a query to display bottom 3 salaries

```
SQL> select * from emp A
  2  where 2 >= (select count(distinct(B.sal)) from emp B
  3  where A.sal > B.sal )
  4  order by sal asc ;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
```

4) Display 1st and 4th maximum salary

```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal )
  4  UNION
  5  select * from emp A
  6  where 3 = (select count(distinct(B.sal)) from emp B
  7  where A.sal < B.sal )
  8  /

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

5) Display 1st, 4th & 6th highest salaries in a single query

```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal )
  4  UNION
  5  select * from emp A
  6  where 3 = (select count(distinct(B.sal)) from emp B
  7  where A.sal < B.sal )
  8  UNION
  9  select * from emp A
 10  where 5 = (select count(distinct(B.sal)) from emp B
 11  where A.sal < B.sal )
 12  /

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

---

*CHAPTER 8*

# FUNCTIONS

Functions – it is a re-usable program that returns a value.

There are **2 types**,

- Pre – defined
- User defined

**Pre – defined**

→ GROUP functions
→ CHARACTER functions
→ NUMERIC functions
→ DATE functions
→ SPECIAL functions

These are used both in SQL and PL/SQL. PL – Procedural Language (it's a extension to SQL, can contain IF statements, loops, exceptions, OOPs, etc .. )

**User – defined**
Used only in PL/SQL and we will not study it here.

We have already learnt about GROUP functions.

Now, let us study the various CHARACTER functions.
**CHARACTER functions**
a) Upper
b) Lower
c) Length

For ex :-

```
SQL> select upper ('oracle'), lower ('ORacLE')
  2  from dual ;

UPPER( LOWER(
------ ------
ORACLE oracle

SQL> select ename, lower(ename) from emp ;

ENAME      LOWER(ENAM
---------- ----------
SMITH      smith
ALLEN      allen
WARD       ward
JONES      jones
MARTIN     martin
BLAKE      blake
CLARK      clark
SCOTT      scott
KING       king
TURNER     turner
ADAMS      adams
JAMES      james
FORD       ford
MILLER     miller

14 rows selected.
```

In the 1st query, we see something called as **dual**.

**Dual** – is a dummy table which is used for performing some independent operations which will not depend on any of the existing tables.

For ex,

1)

```
SQL> select sysdate from dual ;

SYSDATE
---------
09-APR-11
```

This gives the system date.

2)

```
SQL> select 100 + 200 from dual ;

   100+200
----------
       300

SQL> select 100 + 200 " ADDITION "
  2  from dual ;

 ADDITION
----------
       300
```

3)

```
SQL> select ename, sal + 100 from emp ;

ENAME          SAL+100
---------- ----------
SMITH             900
ALLEN            1700
WARD             1350
JONES            3075
MARTIN           1350
BLAKE            2950
CLARK            2550
SCOTT            3100
KING             5100
TURNER           1600
ADAMS            1200
JAMES            1050
FORD             3100
MILLER           1400

14 rows selected.
```

We use dual – when the data is not present in any of the existing tables. Then we use dual.

Length – it returns the length of a given string.

For ex,

1)

```
SQL> select length ('oracle') from dual ;

LENGTH('ORACLE')
----------------
               6
```

2)

```
SQL> select ename, length(ename) from emp ;
```

```
ENAME      LENGTH(ENAME)
---------- -------------
SMITH                  5
ALLEN                  5
WARD                   4
JONES                  5
MARTIN                 6
BLAKE                  5
CLARK                  5
SCOTT                  5
KING                   4
TURNER                 6
ADAMS                  5
JAMES                  5
FORD                   4
MILLER                 6

14 rows selected.
```

3) Display all the employees whose name & job is having exactly 5 characters

```
SQL> select * from emp
  2  where length(ename) =5
  3  and length(job) =5 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 17-DEC-80        800                    20
      7876 ADAMS      CLERK            7788 23-MAY-87       1100                    20
      7900 JAMES      CLERK            7698 03-DEC-81        950                    30
```

REPLACE

It replaces the old value with a new value in the given string.

For ex,

```
SQL> select replace ('oracle','a','p') from dual ;

REPLAC
------
orpcle
```

Here, **a** – is the old value to be replaced with **p** – which is the new value.

```
SQL> select ename, replace(ename, 'A', 'B')
  2  from emp ;
```

This query replaces all the names which has ‘A’ in it with ‘B’.

Let us see the output as shown below,

```
ENAME      REPLACE(EN
---------- ----------
SMITH      SMITH
ALLEN      BLLEN
WARD       WBRD
JONES      JONES
MARTIN     MBRTIN
BLAKE      BLBKE
CLARK      CLBRK
SCOTT      SCOTT
KING       KING
TURNER     TURNER
ADAMS      BDBMS

ENAME      REPLACE(EN
---------- ----------
JAMES      JBMES
FORD       FORD
MILLER     MILLER

14 rows selected.
```

```
SQL> select ename, replace (ename, 'A', NULL)
  2  from emp ;

ENAME      REPLACE(EN
---------- ----------
SMITH      SMITH
ALLEN      LLEN
WARD       WRD
JONES      JONES
MARTIN     MRTIN
BLAKE      BLKE
CLARK      CLRK
SCOTT      SCOTT
KING       KING
TURNER     TURNER
ADAMS      DMS

ENAME      REPLACE(EN
---------- ----------
JAMES      JMES
FORD       FORD
MILLER     MILLER

14 rows selected.
```

SUBSTR

This is called **substring**.

It extracts 'n' characters from x(th) position of a given string.

**For ex,**

```
SQL> select job,
  2  substr (job,1,3) "1 - 3",
  3  substr (job,2,4) "2 - 4",
  4  substr (job,3) "3 - n",
  5  substr (job, -4) "last"
  6  from emp ;

JOB       1 - 2 -  3 - n   last
--------- --- ---- ------- ----
CLERK     CLE LERK ERK     LERK
SALESMAN  SAL ALES LESMAN  SMAN
SALESMAN  SAL ALES LESMAN  SMAN
MANAGER   MAN ANAG NAGER   AGER
SALESMAN  SAL ALES LESMAN  SMAN
MANAGER   MAN ANAG NAGER   AGER
MANAGER   MAN ANAG NAGER   AGER
ANALYST   ANA NALY ALYST   LYST
PRESIDENT PRE RESI ESIDENT DENT
SALESMAN  SAL ALES LESMAN  SMAN
CLERK     CLE LERK ERK     LERK

JOB       1 - 2 -  3 - n   last
--------- --- ---- ------- ----
CLERK     CLE LERK ERK     LERK
ANALYST   ANA NALY ALYST   LYST
CLERK     CLE LERK ERK     LERK

14 rows selected.
```

Here , **(job, '1' , '3')** – means from **job** – extract **from 1st position , 3 characters**.

1) Display the employees whose job starts with 'man'

```
SQL> select * from emp
  2  where substr (job,1,3) = 'MAN';

     EMPNO ENAME      JOB              MGR HIREDATE         SAL        COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

2) Display the employees whose job ends with 'man'

```
SQL> select * from emp
  2  where substr (job,-3) = 'MAN' ;
```

```
     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN         7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN         7698 28-SEP-81       1250       1400         30
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
```

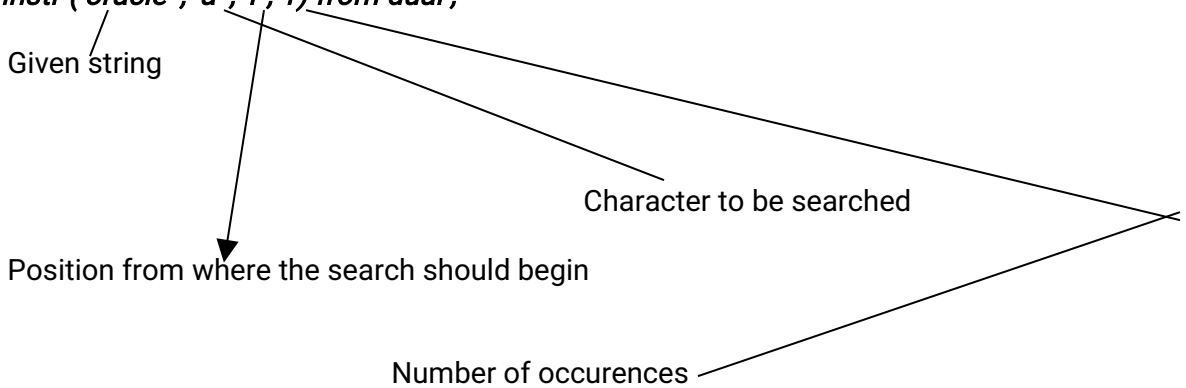
INSTR

This is also called as **instring.**

It returns position of a given character in a given string.

**For ex,**

*Select instr ('oracle', 'a', 1 , 1) from dual ;*




```
SQL> select instr ('oraclea','a',1,1),
  2         instr ('oraclea','a',1,2),
  3         instr ('oraclea','a')
  4  from dual ;

INSTR('ORACLEA','A',1,1) INSTR('ORACLEA','A',1,2) INSTR('ORACLEA','A')
------------------------ ------------------------ --------------------
                       3                        7                    3
```

Display all the employees whose name is having 'L'

```
SQL> select * from emp
  2  where instr (ename,'L',1,1) >0 ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN         7698 20-FEB-81       1600        300         30
      7698 BLAKE      MANAGER          7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER          7839 09-JUN-81       2450                    10
      7934 MILLER     CLERK            7782 23-JAN-82       1300                    10
```

List the employees whose job is having atleast 2 A's in it

```
SQL> select * from emp
  2  where instr(job,'A',1,2) >=2;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      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 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST          7566 19-APR-87       3000                    20
      7844 TURNER     SALESMAN         7698 08-SEP-81       1500          0         30
      7902 FORD       ANALYST          7566 03-DEC-81       3000                    20

9 rows selected.
```

CONCAT

It concatenates any two values or columns.
It is represented by - ||

For ex,

```
SQL> select ename ||' Works as '||job "statement" from emp ;

statement
-----------------------------
SMITH Works as CLERK
ALLEN Works as SALESMAN
WARD Works as SALESMAN
JONES Works as MANAGER
MARTIN Works as SALESMAN
BLAKE Works as MANAGER
CLARK Works as MANAGER
SCOTT Works as ANALYST
KING Works as PRESIDENT
TURNER Works as SALESMAN
ADAMS Works as CLERK
JAMES Works as CLERK
FORD Works as ANALYST
MILLER Works as CLERK

14 rows selected.
```

NUMERIC FUNCTIONS

1) Mod :- it returns the remainder when 1 number is divided by the other.

```
SQL> select mod(7,2) "REM", 7/2 "QUO" from dual ;

       REM        QUO
---------- ----------
         1        3.5
```

Display the employees earning odd numbered salaries.

```
SQL> select * from emp
  2  where mod(sal,2)<>0;

     EMPNO ENAME      JOB              MGR HIREDATE          SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
```

Round
It rounds off a given number to the nearest decimal place.

Trunc
It truncates the given number to the given decimal place. Truncate does not do any rounding.

For ex,

```
SQL> select round(34.76,1),
  2         trunc(34.76,1)
  3  from dual ;

ROUND(34.76,1) TRUNC(34.76,1)
-------------- --------------
          34.8           34.7
```

Here, '1' indicates the number of positions.

DATE FUNCTIONS

1) Sysdate
Stands for System date.
It returns both date & time, but by default – only date is displayed.
The default format is,

dd – mon – yy

```
SQL> select sysdate from dual;

SYSDATE
---------
10-APR-11
```

2) Systimestamp
Introduced from Oracle 9i
Returns date, time and timezone.

```
SQL> select systimestamp from dual
  2  /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.49.08.914000 AM +05:30
```

Here, .914000 – gives the fraction of millisecond which keeps changing as shown below,

```
SQL> select systimestamp from dual
  2  /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.49.08.914000 AM +05:30

SQL> /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.50.25.614000 AM +05:30

SQL> /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.50.26.726000 AM +05:30

SQL> /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.50.27.697000 AM +05:30

SQL> /

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.50.29.109000 AM +05:30
```

In interview – if they ask you – *" which function contains fractions of a second " OR "how to see the system time "* – then answer is "SYSTIMESTAMP".

## SPECIAL FUNCTIONS

### 1) TO – CHAR

Used for displaying the date in different formats.

For ex,

```
SQL> select to_char(sysdate, 'mm/dd/yyyy') from dual ;

TO_CHAR(SY
----------
04/10/2011
```

```
SQL> select to_char (sysdate, 'day, dd-month')from dual ;

TO_CHAR(SYSDATE,'DAY,DD
-----------------------
sunday   , 10-april
```

```
SQL> select ename, to_char(hiredate, 'mm/dd/yyyy') from emp;

ENAME      TO_CHAR(HI
---------- ----------
SMITH      12/17/1980
ALLEN      02/20/1981
WARD       02/22/1981
JONES      04/02/1981
MARTIN     09/28/1981
BLAKE      05/01/1981
CLARK      06/09/1981
SCOTT      04/19/1987
KING       11/17/1981
TURNER     09/08/1981
ADAMS      05/23/1987
JAMES      12/03/1981
FORD       12/03/1981
MILLER     01/23/1982

14 rows selected.
```

```
SQL> select to_char(sysdate,'mm-yyyy hh:mi:ss') from dual ;

TO_CHAR(SYSDATE,
----------------
04-2011 06:56:30
```

Now, let us see how to add 5 hrs to the existing time,

```
SQL> select to_char(sysdate + (5/24),'hh:mi') from dual ;

TO_CH
-----
11:59

SQL> select systimestamp from dual;

SYSTIMESTAMP
---------------------------------------------------------------------------
10-APR-11 06.59.44.909000 AM +05:30
```

We can see that 5 hrs has been added to the current time.

**NVL**
It substitutes a value for a null.

**For ex,**

```
SQL> select ename,sal,comm,sal+NVL(comm,0) "total Sal" from emp;

ENAME             SAL       COMM  total Sal
---------- ---------- ---------- ----------
SMITH              800                  800
ALLEN             1600        300       1900
WARD              1250        500       1750
JONES             2975                  2975
MARTIN            1250       1400       2650
BLAKE             2850                  2850
CLARK             2450                  2450
SCOTT             3000                  3000
KING              5000                  5000
TURNER            1500          0       1500
ADAMS             1100                  1100
JAMES              950                   950
FORD              3000                  3000
MILLER            1300                  1300

14 rows selected.
```

The above query means – if the employee has commission, then add sal + comm. To get total salary – else add 0 to the sal and display total salary.

**DECODE**
It works like '**if – then – else**' statement.
**For ex,**

```
SQL> select ename,job,
  2  decode (job,'CLERK','C','SALESMAN','S','O')
  3  from emp;

ENAME      JOB       D
---------- --------- -
SMITH      CLERK     C
ALLEN      SALESMAN  S
WARD       SALESMAN  S
JONES      MANAGER   O
MARTIN     SALESMAN  S
BLAKE      MANAGER   O
CLARK      MANAGER   O
SCOTT      ANALYST   O
KING       PRESIDENT O
TURNER     SALESMAN  S
ADAMS      CLERK     C
JAMES      CLERK     C
FORD       ANALYST   O
MILLER     CLERK     C

14 rows selected.
```

The above query states that – in job, if clerk is there, replace with C – else if salesman is there, replace it with S – else replace with 'O'.

Display employee name, job, salary and commission. If the commission is NULL, then display -100

```
SQL> select ename, job, sal, NVL(comm, -100) from emp ;

ENAME      JOB              SAL NVL(COMM,-100)
---------- --------- ---------- --------------
SMITH      CLERK             800           -100
ALLEN      SALESMAN         1600            300
WARD       SALESMAN         1250            500
JONES      MANAGER          2975           -100
MARTIN     SALESMAN         1250           1400
BLAKE      MANAGER          2850           -100
CLARK      MANAGER          2450           -100
SCOTT      ANALYST          3000           -100
KING       PRESIDENT        5000           -100
TURNER     SALESMAN         1500              0
ADAMS      CLERK            1100           -100

ENAME      JOB              SAL NVL(COMM,-100)
---------- --------- ---------- --------------
JAMES      CLERK             950           -100
FORD       ANALYST          3000           -100
MILLER     CLERK            1300           -100

14 rows selected.
```

Display all employees whose name is having exactly 1 'L' in it

```
SQL> select * from emp
  2  where instr (ename, 'L',1,1) >0
  3  and instr (ename, 'L',1,2) =0;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
```

*CHAPTER 9*

# NORMALIZATION

**Normalization** is the process of splitting the bigger table into many small tables without changing its functionality.
It is generally carried out during the design phase of SDLC.

**Advantages**
1) it reduces the redundancy (unnecessary repeatation of data)
2) avoids problem due to delete anamoly (inconsistency)

Normalization is a step-by-step process and in each step, we have to perform some activities.

**STEPS IN NORMALIZATION**
1) 1NF – 1st Normal form
2) 2NF – 2nd Normal form
3) 3NF – 3rd Normal form

**1NF**
- We should collect all the required attributes into 1 or more bigger entities.
- We have to assume no 2 records are same (i.e, records should not be duplicated)
- Identify the probable primary key

At the end of 1NF, our data looks like this,

| COLLEGE |
|---|
| RegNo - **PK** |
| Sname |
| Semester |
| DOB |
| MailID |
| Phone |
| BookNo - **PK** |
| Bname |
| Author |
| DOI |
| DOR |
| Fine |

**2NF**
To perform 2NF,
- The tables have to be in 1NF
- Here, we identify all the complete dependencies and move them separately into different tables.

At the end of 2NF, our data looks like this,

| STUDENTS |
|---|

| BOOKS |
|---|

| RegNo - PK |
|---|
| Sname |
| Semester |
| DOB |
| MailID |
| Phone |

| BookNo - PK |
|---|
| RegNo - FK |
| Bname |
| Author |
| DOI |
| DOR |
| Fine |

**3NF**

The table will have to be in 2NF

Here, we identify all the partial dependencies and move such columns to a separate table.

| STUDENTS |
|---|
| RegNo - PK |
| Sname |
| Semester |
| DOB |
| MailID |
| Phone |

| BOOKS |
|---|
| BookNo - PK |
| Bname |
| Author |

| LIBRARY |
|---|
| BookNo - FK |
| RegNo - FK |
| DOI |
| DOR |
| Fine |

**Disadvantage of Normalization**

The only minor disadvantage is we may have to write complex queries as we have more number of tables to be accessed.

**Denormalization** is the process of combining more than 1 smaller table to form 1 bigger table is called as denormalization.

**CODD rules** ( **Differentiates between DBMS & RDBMS** )

1) should support NULL values
2) should support creation of relationship between tables
3) should support DDL, DML, TCL
4) should support constraints like PK, Unique, CHK
5) should support query techniques like sub – queries, joins, grouping etc.

**Oracle 9i Features (i means internet)**

- TIMESTAMP datatype
- SYSTIMESTAMP function
- ANSI style joins
- Renaming a column

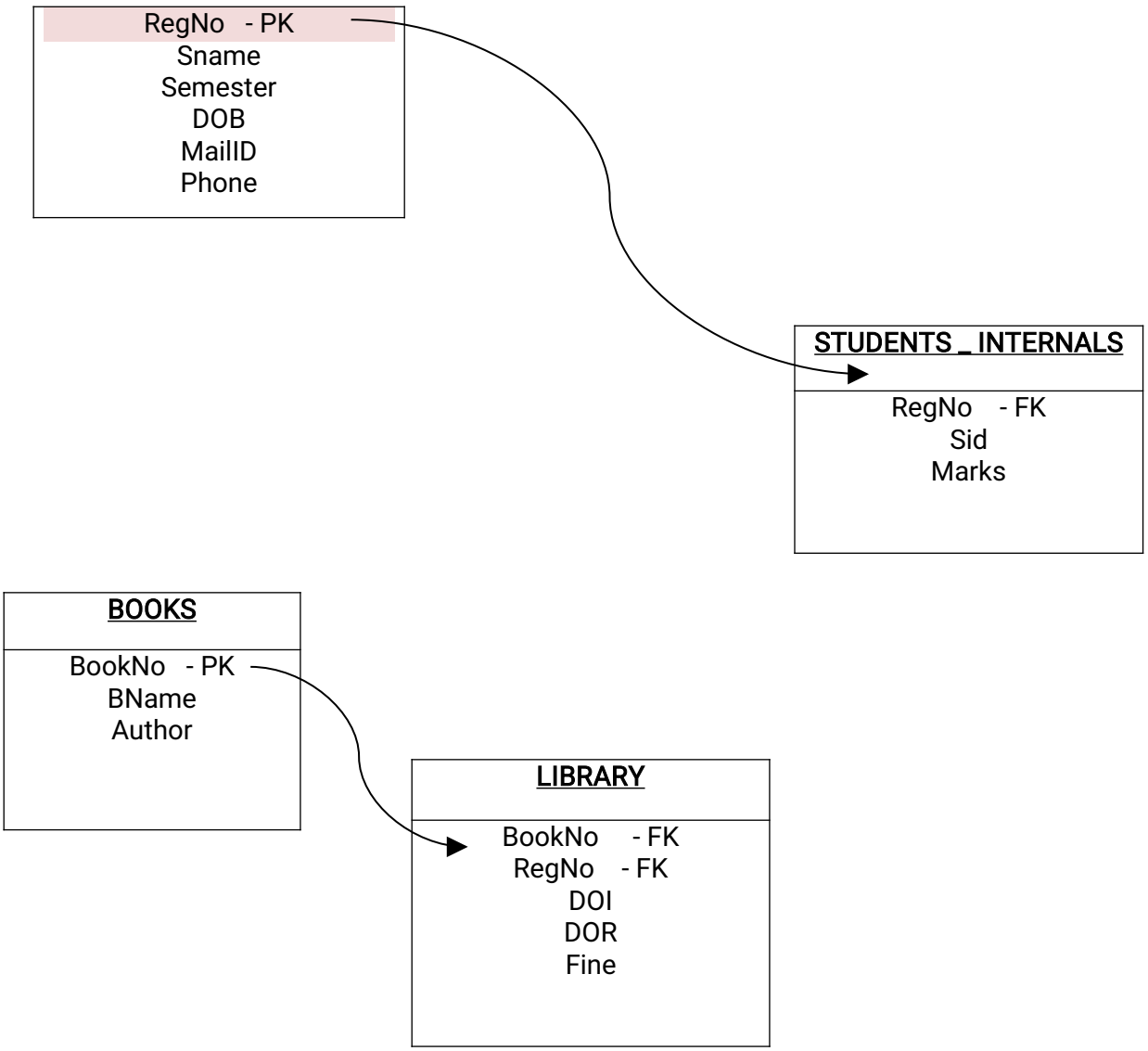
**Oracle 10g features (g means grid)**

- Recycle Bin

**ERD - Entity Relationship Diagram**

It is the pictorial representation of all the entities and their relationships (tables).

| STUDENTS |
|---|




******* THE END *******

**Pradeepamr007@gmail.com**

---