**DBMS:** A database management system (DBMS) is a software package with [computer programs](http://en.wikipedia.org/wiki/Computer_program) that control the creation, maintenance, and use of a [database](http://en.wikipedia.org/wiki/Database).

To open Oracle in Command prompt just type sqlplus in RUN.

**RDBMS:** RDBMS stands for relational database management system. A relational database is the one in which data is stored in the form of tables i.e. rows and columns. And a system that manages this database is known as database management system. The major functions performed by RDBMS are creating, editing and deleting tables and inserting, deleting and editing data into that tables.  
The data items are recognized by a unique entry known as primary key. The main examples of RDBMS are Oracle, SQL server, DBase.

**SQL (Structured query language)**

1. It is non procedural language which is used to communicate any type of database such as Oracle DB2, SQL server e.t.c.,
2. This language was developed by a German scientist MR MF Codd in 1968. ANSI approved this language in 1970 and it was released at 1972.

**Query:** Query is a mechanism by using query we can perform language operations. Those are DML,DDL,DRL,DCL,TCL.

**Features of SQL:**

1. Sql is not a case sensitive language because all the commands of SQL we can write in capital letters or small letters.
2. Every command of SQL should ends with semicolon.
3. SQL can also Pronounced as SEQUEL (Structured English query language).
4. SQL can call as NLI (Natural language interface) because every command of SQL almost similar reassembles to normal English language.
5. SQL can be called as common language interface because this is the only language to communicate with any type of database such as Oracle, Sybase, DB2, Teradata e.t.c.,
6. Structure query language was mainly subdivided into five sublanguages those are fallowing types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Definition Language**  **(DR CAT)** | **Data Manipulation Language** | **Data Retrieval Language** | **Transition Control Language(Three)** | **Data Control Language(Double**) |
| DROP | INSERT | SELECT | COMMIT | GRANT |
| Rename | DELETE |  | ROLLBACK | REVOKE |
| CREATE | UPDATE |  | SAVEPOINT |  |
| ALTER |  |  |  |  |
| TRUNCATE |  |  |  |  |

**Data types in Oracle 10g:**

These data types are totally divided into two types.

1. Number
2. Alphabet.
3. **NUMBER:** This data type allows us to enter numeric values such as integer values or decimal values. In SQL these data types further divided into NUMBER(SIZE)
4. **NUMBER(P,S) (P=Precision, S= Scale)**
5. **NUMBER (SIZE):** This data type allows us to enter only integer values student roll numbers employee numbers e.t.c., Here size represents no of digits.

**EX:** EID NUMBER(3)

1. **NUMBER(P,S) :** This data type allows us to enter decimal values such as Employee salaries student average marks e.t.c., Here p represents precision and s represents scale. Here Precision means no of digits including decimal points where as scale means no of decimal places.

**EX:**

ESAL NUMBER(6,2)

AM NUMBER(3,2)

1. **Alphabets**: The data type allows us to enter character values such as student names employee names e.t.c., this data type is further divided into three types 1. Char(Size) 2. Varchar(Size), 3. Varchar2(Size)

**Char (Size):** This data type allows us to enter character values the maximum length of this data type is 1 to 2000 bytes of characters this data type always allocates the memory in a static fashion.

**EX:** Ename char(15)

**Varchar (Size)/Varchar2(Size):** This data type also allows us to enter character values such as student names, employee names e.t.c., The maximum length of this data type is 1 to 4000 characters or bytes. It always allocates the memory in a dynamic fashion.

**Note:** The difference between varchar data type and varchar2 data type is – Varchar data type is developed along with SQL where as varchar2 data type developed by Oracle corporation varchar data type always allocates the memory in a dynamic fashion where as varchar2 data type supports garbage values.

**DATE:** The data type allows us to enter date values such as student joining date employee hiring data e.t.c., the default date format is Oracle is DD/MM/.YYYY.

**Ex:** Jdate date

**Timestamp:** The data type allows to enter both date and time values. The default format of timestamp is DD/MM/YYYY HH/MM/SS.

**EX:** Logintime timestamp

**Miscellunious datatypes:** These data types are further divided into three types CLOB, BLOB, Bfile.

1. **CLOB (Character Large Object):** This data type allows us to all types of characters and the minimum length of this data type is 1 to 4GB.

Whenever we need to enter employee history student conduct e.t.c., then we need to use this data type.

1. **BLOB (Binary large Object):** This data type allows us to enter any type of photos, graphical pictures clippings, sounds, multimedia messages e.t.c., the minimum length of this data type is 1 to 4 GB.
2. **BFILE:** This data type also allows us to enter BLOB types of values and also binary data, XML data e.t.c., the minimum length of data type is 1 to 4 GB.
3. **Data definition Language:** This is the first sub language in SQL which is used to define the database objects such as table, view, synonym, index and sequence. This language contains five commands those are

i)CREATE ii) ALTER iii) RENAME iv) TRUNCATE v)DROP.

i) **CREATE:** This command is used to creating the database objects such as Table, View e.t.c.,

**Syntax:** create table <table name>

(col1 datatype(Size),

col2 datatype(Size),

coln datatype(size));

ex: create table emp(eid varhcar(3), ename varchar2(12), sal number(6,2));

**Rules for naming a table:**

1. Table name should start with alphabets which contains minimum 1 and maximum 30 characters it should not allows any spaces or any special characters such as except \_# and 0 to 9.
2. A table can have minimum 1 column maximum thousand columns.
3. A table can have 0 no of records and maximum n no of records up to hard disk capacity.
4. Oracle reserved keywords and words should not be used column names or table names.
5. The rules which we are following for table name the same rules are applicable for column names.
6. The numeric precision for column must be 1 to 38.
7. **ALTER:** This command is used to modify the structure of the table. Using this command we can perform four different operations. This command contains four subprograms those are

ALTER – Modify

ALTER – ADD

ALTER – RENAME

ALTER – DROP

**ALTER – Modify:** This command is used to increase or decrease the size of the data type and also we can use the data type from all data type to new data type.

**Syntax:** ALTER TABLE <Table name> MODIFY <Column name> data type(size)

**EX:** ALTER TABLE EMP MODIFY ENAME VARCHAR2(20);

**Syntax for modify more than one column:**

ALTER TABLE <Table name> MODIFY (col1 data type(size), col2 data type(size)----,coln data type(size))

**EX:** Alter table emp modify (eid number(5), ename char(10));

**ALTER ADD:** This command is used to add a new column to the existing table.

**Syn:** ALTER TABLE<Table name>add column name datatype(size)

**EX:** ALTER TABLE EMP ADD DEPTNO NUMBER(3)

**Syntax to add more than one column:**

ALTER TABLE<table name> add (col1 datatype(size), col2 datatype(size)----,coln datatype(size))

**EX:** ALTER TABLE EMP ADD( mbnno number(10), addrs varchar2(10))

**Note:** When ever we add a new column to a table the new column is always added to the end of the table only.

**ALTER – RENAME:** This command is used to change the column name from old column name to new column name.

**Syn:** ALTER TABLE<table name> rename column<old column name> to <new column name)

**EX:** ALTER TABLE EMP RENAME COLUMN SAL TO SLARY.

**Syntax change table name:**

ALTER TABLE<old table name> RENAME TO <New table name>

**EX:** ALTER TABLE EMP RENAME TO EMP1;

**ALTER – DROP:** This command is used to remove the column from the existing table.

**Syn:** ALTER TABLE< table name>DROP COLUMN COLUMN NAME.

**EX:** ALTER TABLE EMP DROP COLUMN DEPTNO.

**Syntax to drop more than one column:**

ALTER TABLE EMP DROP(eid, ename, salary, mbno, addrs)// Here we can’t drop all columns.

1. **RENAME:** This command is used to change the table name from old table name to new table name.

**Syn:** RENAME <old table name> to <new table name>

**EX:** RENAME EMP TO EMP1;

1. **TRUNCATE:** This command is use delete records permanently from the existing table.

**Syn:** TRUNCATE TABLE<table name>

**Syn:** TRUNCATE TABLE EMP;

1. **DROP:** This command is used to drop entire data from database.

**Syn:** DROP TABLE< table name>

**EX:** DROP TABLE EMP;

**Difference between TRUNCATE and DELETE:**

1. TRUNCATE is DDL command, DELETE is DML command.
2. We can delete data permanently through TRUNCATE; Through DELTE it is temporarily.
3. We can’t delete single record Through TRUNCATE, but it is possible through DELTE.

**Recycle bin:** To retrieve data from recycle bin

Syn: Select \* from recycle bin

**Flashback:** This command is used to retrieve back the table from the recycle bin.

**Syn:** FLASHBACK TABLE<Table name> TO BEFORE DROP

**EX:** Flashback table emp to before drop

**Syntax to drop the table permanently from recycle bin:**

**Syn:** PURGE TABLE <Table name>

**EX:** PURGE TABLE EMP;

**Syntax drop table permanently from database:**

**Syn:** DROP TABLE < Table name> PURGE

**EX:** DROP TABLE EMP PURGE

1. **Data Manipulation Language:** This is the second sub language in SQL which contain four commands those are INSERT, UPDATE, SELECT, DELETE and which is used for manipulate the data within database.

**INSERT:** This command is used to insert the records in the table using this command we can insert the data into the table in two methods

1. Explicit method 2. Implicit method
2. **Explicit Method:** In this method user need to enter all the values into the columns without left any column data.

**Syn:** INSERT INTO <TABLE NAME> VALUES (Val1, Val2,------,Valn)

**EX:** INSERT INTO EMP Values(101,’RAJU’,500)

**Note:** We can use && in front of any column. By placing like this the use is – “It takes particular column values as default for remaining all values.

**Syntax to insert the records into the table using insertion operation:**

INSERT INTO<Table name> VALUES(&col1,&col2,--------,&coln)

**EX:** INSERT INTO EMP Values(&Eid,’&Ename’,&Sal’)

**Implicit method:** This method we can enter the values at required columns in the table.

**EX:** INSERT INTO < Table name> (Col1,Col2,------,Coln) values(Val1, Val2,--------,Valn)

**EX:** INSERT INTO EMP(EID,SAL) VALUES(115,9800)

**Syntax to insert record using & Symbol:**

INSERT INTO < TABLE NAME>(col1,col2,-------,coln) values(&col1,&col2------,&coln)

**EX:** INSERT INTO EMP(Eid,Sal) VALUES(&Eid,&Sal) VALUES(&Eid,&Sal)

**UPDATE:** This command is used to modify the data existing table. Using this command we can modify all the records in the table also we can modify specific records in the table using WHERE clause.

Syn: UPDATE <Table name> SET COLUMN NAME = “VALUE’

Syn: UPDATE EMP SET ENAME = ‘RAJU’

**Syntax to modify more than one column data at a time:**

UPDATE <Table name> SET COL1=VALUE,COL2=VALUE,-------,COLN=VALUE

**EX:** UPDATE EMP SET EID = 007, ENAME = ‘JAMES’

**DELETE:** This command is used to delete the records from the existing table. Using this command we can delete all the records and also we can delete specific records from the table.

**Syn:** DELETE FROM <Table Name> <Column Name>

**EX:** delete from rose no;

**EX:** delete from rose where name='UPASANA';

**EX3:** delete rose name where name='upasana' or name='vijay';

1. **Data Retrieval Language:**

**SELECT:** This command is used to retrieve the data from the existing table. Using this command we can retrieve all records and also we can retrieve some specific records in the table(Using where clause).

**Syn:** SELECT \* FROM < Table Name>

**EX:** SELECT \* FROM EMP;

Here \* represents all columns.

By using select command we can retrieve the data in three ways 1. Projection 2. Selection 3. Joins.

1. **Projection:** Retrieve the data from specific columns in the table is called projection.

**Syn:** SELECT col1,col2,--------------------,coln from <table name>

**ALIASES:** Alias is duplicate name or an alternative name for the original column name or table name or expression name. Whenever we need to submit meaningful or understanding reports then we need to use alias names.

We can provide the alias names in three levels i) Column level ii) Table level iii) Expression level.

1. **Column level Alias:** Providing the alias name for the columns is called column level alias.

**Syn:** SELECT COL1 AS “ALIAS NAME”, COL2 AS “ ALIAS NAME” ,---------,COLUMN NAME N AS “ALIAS NAME” FROM <Table name>

**EX:** SELECT EID AS “EMPLOYEEID”, ENAME AS “EMPLOYEE NAME” FROM EMP;

In the above example the keyword as is an optional keyword.

**Syn:** SELECT COL1 “ALIAS NAME”, COL2 “ALIAS NAME”, -------,COLN “ALIAS NAME” FROM <Table name>

**EX:** SELLECT EID “EMPLOYEEID”, ENAME “ EMPLOYEE NAME” FROM EMP;

In the above example we no need to use double quotation.

**Syn:** SELECT COL1 ALIASNAME, COL2 ALIASNAME,--------,COLN ALIASNAME FROM <Table name>

**EX:** SELECT EID EMPLOYEEID, ENAME EMPLOYEENAME FROM EMP;

1. **Expression Level alias:**

Providing the alias names for expressions is known as expression level alias.

**EX:** SELECT EID EMPLOYEEID, ENAME EMPLOYEENAME, SAL SALARY, SAL\*12 ANNUALSALARY FROM EMP;

**EX:** SELECT EID EMPLOYEEID, ENAME EMPLOYEENAME, SAL ANNUALSALARY, SAL\*12 ANNUALSALARY FROM EMP WHERE ANNUALSALARY>150000

**O/P:** “ANNUAL SALARY” in valid identifier because Alias names are not identifiers. Identifier means column name.

**Note:** The above example the fallowing error message- “Annual salary invalid identifier”. So we cant check the conditions of Alias names.

**EX:** SELECT EID EMPLOYEEID, ENAME EMPLOYEENAME, SAL SALARY, SAL\*12 ANNUALSALARY FROM EMP WHERE SAL\*12>150000

**EX:** SELECT EMP.\*, SAL\*12 ANNUALSALARY FROM EMP WHERE SAL\*12>150000

**Where clause:** This clause is used to check the condition based on the condition we can retrieve, we can modify, and we can remove some specific records in the table.

**Operators in SQL:** Operator means special symbol which performs some specific operations on operators.

1. Arithmetic operator: +, -,\*, /, MOD
2. Logical: AND, OR, NOT.
3. Relational: > , < ,>= ,<= ,!=,<>,^=,=
4. Concatenation: ||
5. Assignment: =
6. Miscellaneous: Between, not between, like, not like, in, not in, is null, is not null, exists, not exists, any, all, some, union, union all, intersect, minus.

**Update – Where Clause:**

Q) Write a query that changes the employee’s salaries as 2500 whose empno is 7788.

A) update emp set sal = 2500 where empno=7788;

Q) write a query change the employee’s salaries as 5000 who are working under 10 th department.

A) update emp set sal=5000 where deptno in(10);

(OR)

update emp set sal=5000 where deptno = 10;

Q) write the query hike the employee’s salaries 30% who are working in the 10th Department.

A) update emp set sal=sal+sal\*0.3 where deptno in(20);

(OR)

update emp set sal=sal+sal\*(30/100) where deptno in(20);

Q) write the query increase the employee’s salaries 1000 whose names start with A.

A) update emp set sal=sal+1000 where ename like 'A%';

Q) Write a query increases the employee’s salary 1000/- whose name is SCOTT working under 20th DEPT.

A) update emp set sal=sal+1000 where ename='SCOTT' and deptno = 20;

Q) Increase the employee salary by 1000 whose name starts with ‘S’ and ends with ‘T’ and third letter is ‘O’.

A) update emp set sal=sal+1000 where ename like 'S%\_O%T';

Q) Change deptno=30 whose emp sal between 1500 to 3000

A) update emp set deptno=30 where sal between 1500 and 3000;

(OR)

update emp set deptno=30 where sal >=1500 and sal<=3000

Q) Change job as clerk whose job is manager are joined before 1982.

Update emp set job = 'CLERK' where job = 'MANAGER' and hiredate< '01-JAN-1982';

Q) Write a query to increase the employee’s salaries 1000 who are having more than 10 years.

A) update emp set sal = sal+1000 where (sysdate-hiredate)/365>10;

Q) Write a query change the employee’s salaries as 7000 who are having commission.

A) update emp set sal=sal+7000 where comm>=0;

Q) Write a query change the employee’s salary 10000 who are not working under anybody.

A) update emp set sal = sal+10000 where mgr is null;

Q) Write a query change employee name as Srinivas and Salary 9000 whose empno=7566.

A) update emp set ename = 'SRINIVAS' and sal = 9000 where empno =7566;

Q) Write a query change the employees salary 5000 whose name contain more than 6 Character.

A) update emp set sal = 5000 where length(ename)>6;

Q) Write a query change the employees salaries as 6000 who is less than king.

A) update emp set sal = 6000 where ename < 'KING';

Q) Change employee salary as 5799 whose name contains to As.

A) update emp set sal=5799 where ename like '%A%';

Q) Change employee salary 2344 where employee name ending with K.

A) update emp set sal = 2344 where ename like '%K';

Q) Increase employee salaries as 1000 where employee name last but one letter end with A.

A) update emp set sal = sal+1000 where ename like '\_A%';

Q) Increase the the employee salaries as 1000 where employee name’s second letter starts with A.

A) update emp set sal = sal+1000 where ename like 'A%\_';

Q) Change the employee salary 4500 where who’s name contains two As

A) update emp set sal = 4500 where ename like '%A%A%';

Q) Change employee salary 4000 whose hiredate is 23rd MAY 87.

A) update emp set sal =4000 where hiredate = ’23-MAY-87’

Q) Change the employee salary as 6000 whose joining month in start with M.

A) Update emp set sal=6000 where hiredate like ‘\_\_\_M%’;

Q) Change employee salary 1234 where hiredate is in the month MAY or which starts with M and ends with Y

A) update emp set sal=1234 where hiredate like '\_\_\_M%\_Y%';

Q)Change the employee salary 4567 where hiredate is 23rd.

A) update emp set sal=4567 where hiredate like '2%3%';

Delete with where clause:

Syn: DELETE FROM <Table name> where <condition>

EX: DELETE FROM EMP WHERE EMPNO=7788

Q) Write a query delete employee details who are working under 10th DEPT.

A) delete from emp where deptno in (10);

Q) Delete emp details whose number and job ends with n?

A) delete emp where ename like '%N' and job like '%N';

Q) Write a query delete the employee details who are looking under 10th and 20th DEPT.

A) delete from emp where deptno =10 or deptno =20;

**NOTE:** Here we didn’t use and because an employee doesn’t work in two departments. So we can write this query two ways.

delete emp where deptno in(10,20);

Q) Write a query delete the employee who are joined in the month april?

A) delete from emp where hiredate like '%APR%';

(OR)

delete from emp where hiredate like '\_\_\_A%P%';

Q) Write a query delete the employee details who are not working under 20th ?

A) delete from emp where deptnonot in (20);

(OR)

delete from emp where deptno <> 20;

Q) write a query delete emp details whose annual salary is more than 25000

A) delete from emp where (sal\*12)>2000;

Q) Write a query delete employee details whose name contains letters only 4?

A) delete from emp where length(ename) = 4;

(OR)

delete from emp where ename like '\_\_\_\_';

Q) Write a query to delete the emp details whose salary not in range of 2000 and 3000

A) delete from emp where sal not between 20000 and 30000;

Q) Write a query delete emp details whose name contains letter A?

A) delete from emp where ename like '%I';

Q) Write the emp details who are having commission and commission more than 1000?

A) delete from emp where comm is not null and comm>1000;

Select with Where clause:

Select with Where clause:

**Seelction:** Retrieve the data based on some condition is known as selection.

Syn: SELECT \* FROM EMP WHERE <Condition>

Q) Write a query display the employee details whose salary is less than 3000 and working under dept 10?

A) select \* from emp where sal<=3000 and deptno in(10);

Q) write the query display the employee details whose empno and mgr ends with no 9?

A) select \* from emp where empno like '%9' and mgr like '%9';

Q) Write a query display details whose name and job contains only 5 letters?

A) select \* from emp where length(ename)= 5 and length(job)=5;

(OR)

select \* from emp where ename like '\_\_\_\_\_\_\_' and job like '\_\_\_\_\_';

Q) Write a query display employee details whose total salary is more than 2500.

A) select\* from emp where (sal+comm)>2000;

Q) write a query display the employee details whose sal and comm. Ends with zero?

A) select \* from emp where sal like '%0' and comm like '%0';

Q) write a query display the employee details who are having more than 30 years exp?

A) select \* from emp where (sysdate-hiredate)/365>30 and deptno in(10) and sal>3000;

Q) Write a query to display the employee details who doesn’t have commission and their names contains A and department no is 20.

A) select \* from emp where comm is null and ename like '%A%' and deptno in(20);

**Properties of the RDBMS:**

1. Data can be stored in the form of the table.
2. A table can contain collection of rows and columns.
3. The horizontal things are called rows and vertical things are called as columns.
4. The intersection of rows/columns called table.
5. A cell place where we can store data.
6. The other name of row also called as record/tuple.
7. The other name of column can be called as field and other name is entity.
8. Tables should not contain duplicate columns.
9. While inserting records in the table Programmer no need to follow any order.
10. Database should not contain duplicate objects.
11. When we defined the columns in the table programmer need to follow order.

**Order By:**

This clause is used to arrange the data either ascending or descending order. By default order by clause will arrange the data in ascending order.

If we want to arrange the data in descending order then we use an option called DESC stands for descending order.

We can order by clause on character columns then it will arrange the data in alphabetic order.

We can apply the order by clause on more than one column in the same table.

We can apply the order by clause only with the “select” command.

**Syntax by applying order by clause**

SEELCT \* FROM EMP ORDER BY <Column name> <ASCE><DESC>

EX: Select \* from xyz order by ename salary desc;

**Functions:**

* A function is a self contained block which is use to perform some task.
* The main advantage of function is code reusability.
* Then it automatically reduces the redundancy of the instructions so that reduces the maintenance cost and increase the system performance.
* Function always returns only one value.

Functions are dividing into two types.

1. System/predefined functions
2. User defined functions
3. **System defined functions:** A function which is defined by the system comes along with the software those functions can be called as System defined functions.
4. **User defined functions:** A function which is defined manually and programmatically those functions are called used defined functions.

In SQL predefined functions are further divided into three types.

1. Single row functions
2. Multiple row functions
3. Miscellaneous functions.
4. **Single row functions:** Single row functions are those functions will work on each and every record in the table and returs one value.
5. **Multiple row functions:** Multiple row functions are those functions will work on the entire table and returns only one value.
6. **Miscellaneous functions:** Miscellaneous functions are those functions which are used to convert the null values into not null values.

Single row functions are further divided into following types.

1. STRING/CHAR
2. MATH/NUMERIC
3. DATE
4. DATA Conversions

Multiple row functions are divided into 5 types.

1. MAX
2. MIN
3. SUM
4. AVG
5. COUNT

Count divided into three types.

1. COUNT(\*)
2. COUNT(EXP)
3. COUNT(dis.EXP)

Miscellanies functions are classified into four types.

1. NVL
2. NVL2
3. NULLIF
4. COALASCE

**Tables:** It can be defined as collection of columns and rows tables are divided into two types.

1. Database Tables ii) NON database tables.
2. **Database Tables:** A database table is a table which contains actual information.
3. **Non Database Tables:** A non database table is a table which doesn’t contain any information.

**EX:** DUMMY or DUAL.

There are there advantages of non database tables.

* Evaluate any mathematical expression.
* Evaluate any predefined function.
* Evaluate any psudo column value.

**STRING/CHAR FUNCTIONS:**

1. **ASCII():** This function returns the ASCII value from the high level language code.

**EX:** select ASCII('a') from dual;

1. **Chr():** This function return the high level language value from ASCII code.

**EX:** SQL> select chr(97) from dual;

1. **CONCAT:** This function is used to concatenate two strings

**EX:** select concat('Sathya','Technologies') from dual;

1. **INITCAP():** This function allows first letter of every word into the Capital letter.

**EX:** select initcap('siva bhargava reddy') from dual;

1. **LENGTH:** This function returns the length of the String.

**EX:** SQL> select length('Sathya technologies') from dual;

**EX2:** SQL> select ename , length(ename) from emp;

1. **LOWER():**function converts the String into the small letters.

**EX:** select lower('BHARGAV') from dual;

1. **UPPER():** This function converts the String into the capital letters.

**EX:** select upper('bhargav') from dual;

1. **LPAD():** This function is used to adding the special character from the left hand side to right hand side.

**EX:** select LPAD('Sathya', 10, '\*') from dual;

If we give negative number or zero it won’t display any thing.

1. **RPAD():** This function adds the no of special characters from the right hand side to left hand side.

EX: select RPAD('Sathya',10,'\*') from dual;

1. **LTRIM():** This function allows us to remove the blank spaces from the left hand side of the string.

**EX:** select ltrim(' Bhargav') from dual;

1. **RTRIM():** This function allows us to remove the blank spaces from the right hand side of the string.

**EX:** select rtrim('BAHRGAV ') from dual;

Select rtrim(‘BHARGAV’,’AV’) from dual;

1. **TRIM():** This function trims both sides of values.

**EX:** select trim( 'i' from 'indiani') from dual;

**We can use the trim function fallowing ways.**

EX: select trim( leading'i' from 'indiani') from dual;

EX2: select trim( trailing'i' from 'indiani') from dual;

1. **REPLACE():** This function is used to replaces more than a character.

**EX:** select replace('jack and jar','ja','bl') from dual;

1. **REVERSE():** This function allows us to reverse a word.

**EX:** select reverse('bhargav') from dual;

**EX2:** Select rtrim(‘BHARGAV’,’BH’) from dual;

1. **SOUNDEX():** This function traces out similar words which are similar to pronounce.

**EX:** select \* from emp where soundex(ename)=soundex('ALINN');

**EX:** select \* from emp where soundex(ename)=soundex('KARLING');

In the above example I used KARLING it is similar to CARLING. But letters are not similar hence forth it shows error message.

1. **SUBSTR():** This function is to be used to returns the path of the string from the main string.

**EX:** select substr('Sathya technologies','7') from dual;

**EX2:** select substr('Sathya technologies','7','5') from dual;

By using above command we can get substring through 7 and no of letters to display through 5.

**Real time scenario:** If someone would like to know acno in case forgot accno then if we know last no then we can use this function to retrieve the accno.

1. **Translate:** This function is used to replaces a character with another character with in the string.

**EX:** select translate ('jack and jue','j','b') from dual;

select translate('jack and jue','j','bl') from dual;

In the above example we got only one value translate. Because it works on one value.

1. **VSIZE():** This function is used to count the no of bytes bytes occupied by the given string.

EX: select vsize('Bhargav') from dual;

**Math functions:**

1. **ABS():** This function is used to convert –VE values into +VE values.

**EX:** select abs(-9.5) from dual;

1. **CEIL():**This function is used to round the given number to highest number.

**EX:** select ceil(-9.4) from dual;

1. **FLOOR():** This function is used to round the given number to least number.

**EX:** select floor(-9.4) from dual;

1. **Greatest():** This function is used to returns the maximum value from list of numbers.

**EX:** select greatest (5,6,7,8) from dual;

1. **Least():**This function is used to returns the minimum value from list of numbers.

**EX:** select least(5,6,7,8) from dual;

1. **MOD():** this function returns reminder value from given number.

**EX:** select mod(10,2) from dual;

1. **ROUND():** This function round converts given number to nearest number.

**EX:** Select round(-9.4) from dual;

**EX:** Select round(-9.5) from dual;

The above example shows the output -10 because round treats 9.5 as 10.

1. **SIGN():** This function returns the sign of the given value.

**EX:** select sign(-9.4) from dual;

1. **SIN():** This function give sin value of given number.

**EX:** select sin(12) from dual;

We can get cot,tan,cot,sec,cosec also.

1. **SQRT():** This function square roots the given function.

**EX**: select sqrt(16) from dual;

**Date functions:** These functions are classified into four types

1. MONTHS\_BETWEEN
2. ADD\_MONTHS
3. LAST\_DAY
4. NEXT\_DAY
5. **MONTHS BETWEEN():** This function is used to find out months between two dates.

**EX:** select MONTHS\_BETWEEN( sysdate, '12-FEB-11') from dual;

1. **ADD\_MONTHS:** This function is used to add the no of months to the given date.

**EX:** select ADD\_MONTHS(sysdate, 12) from dual;

1. **LAST\_DAY():** This is used to display last day from current day or other days.

**EX:** select last\_day(sysdate) from dual;

1. **NEXT\_DAY():** This function is used to display next day .

**EX:** select last\_day(SYSDATE) from dual;

**Group and Aggregate functions:** These functions are classified into five types those are SUM(), MAX(), MIN(), AVG(), COUNT().

1. **SUM():** This function is used to find out sum of the all values of a particular column in the table.

**EX:** select sum(sal) from emp;

1. **AVG():** This function is used to find out an average value of table’s column

**EX:** Select avg(sal) from dual;

1. **MAX():** This function is used to find maximum value of particular table’s column

EX: select MAX(SAL) from emp;

1. **MIN():** This function is used to find minimum value of particular column’s value.

EX: select MIN(SAL) from emp;

1. **COUNT():** This function is used to count the no of records in the given table. It contains the sub functions COUNT(\*), COUNT(exp), COUNT(dis exp).
2. **Count():** This function is used to count the no of records in the given table including duplicate and null values.

**EX:** select count(\*) from emp;

1. **Count(exp)**: This function is used to count the no of records in the given table including duplicate values but excluding null values.

**EX:** select count(mgr) from emp;

1. **Count(distinct):** This function is used to count no of records in the given table excluding duplicate and null values.

**EX:** select count(distinct mgr) from emp;

**Data Conversion Functions:**

* Bin\_to\_num
* Char\_to\_rowid
* Rowid\_to\_char
* To\_number
* To\_char
* To\_date

a**) BIN\_TO\_NUM**

This will convert the binary value to its numerical equivalent.

**Syn:** bin\_to\_num( *binary\_bits*)

**Ex:**

SQL> select bin\_to\_num(1,1,0) from dual;

BIN\_TO\_NUM(1,1,0)

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

6

* If all the bits are zero then it produces zero.
* If all the bits are null then it produces an error.

**b) CHARTOROWID:** This will convert a character string to act like an internal oracle row identifier or rowid.

**c) ROWIDTOCHAR:** This will convert an internal oracle row identifier or rowid to character string.

**d) TO\_NUMBER:** This will convert a char or varchar to number.

**EX:** SQL> select to\_number('10') from dual;

**TO\_NUMBER('10')**

**---------------**

**10**

**e) TO\_CHAR:** This will convert a number or date to character string.

SQL> select to\_char(round(to\_date('24-dec-06','dd-mon-yy')), 'dd mon yyyy

hh:mi:ss am') from dual;

TO\_CHAR(ROUND(TO\_DATE('

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

24 dec 2006 12:00:00 am

**EX:** select to\_char('24-dec-06') from dual;

**f) TO\_DATE:** This will convert a number, char or varchar to a date.

**EX:** SQL> select round(to\_date('24-dec-04','dd-mon-yy'),'year'), round(to\_date('11-mar-

06','dd-mon-yy'),'year') from dual;

ROUND(TO\_ ROUND(TO\_

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

01-JAN-05 01-JAN-06

SQL> select round(to\_date('11-jan-04','dd-mon-yy'),'month'), round(to\_date('18-

jan-04','dd-mon-yy'),'month') from dual;

ROUND(TO\_ ROUND(TO\_

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

01-JAN-04 01-FEB-04

SQL> select round(to\_date('26-dec-06','dd-mon-yy'),'day'), round(to\_date('29-dec-

06','dd-mon-yy'),'day') from dual;

ROUND(TO\_ ROUND(TO\_

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

24-DEC-06 31-DEC-06

**EX:**

select dob from date\_test1 where dob=to\_date('1999-05-12','yyyy-mm-dd');

Similar to insert and update.

**EX:** update date\_test1 set dob=to\_date('1999-09-24','yyyy-mm-dd') where

dob='23-SEP-19';

**Miscellaneous functions:**

1. **NVL ():** This function is used to converting the null values into not null values. These are classified into following types.
2. NVL()
3. NVL2()
4. NULLIF()
5. COALESCE()
6. **NVL():** This function is used to converting the null values into not null values.

**SYN:** NVL(a,b) the first parameter a represents column names and b represents replaced values if column contains null value then it is replaced with b value.

**EX:** select empno,ename,sal,comm,sal+nvl(comm,100) from emp;

**EX2:** select emp.\*,nvl(comm,100) as NEWSALARY from emp;

1. NVL2(): This function is used to converting the null values into not null values.

**SYN:** NVL2(a,b,c)

**EX:** select emp.\*,nvl2(comm,100,200) as NEWSALARY from emp;

Here a represents column name, b and c represents replaced values. If column contains null values then it is replaced with c if column doesn’t contain null value then it is replaced with B.

1. **NULLIF():** This function is used to converting the not null values into null values.

**SYN:** NULLIF(a,b) if a value, b values both are equal then it is replaced with null if a not equal to b it replaced with a value.

**EX:** select ename,length(ename),job,length(job), nullif(length(ename),length(job)) from emp;

**COALESCE():** This function returns first occurance of not null value.

**EX:** select coalesce(null,20,null,20) from dual;

**Case Statement:** It is also working as a switch case statement in C language. The only difference between if and switch case is in case of if condition we can check the condition or relation between more than one variable at a time but where as in switch statement we can check all the conditions on one variable.

**EX:** Select sal,

Case sal

When 500 then ‘low’

When 5000 then ‘high’

Else ‘medium’

End case

From emp;

**Decode:** It will checks value by value whatever we specify afterwards display result .

**EX:** select sal, decode(sal,1234,'LOW',2975,'MEDIUM',5000,'HIGH') from emp;

**Rowid and Rownum:** These psudo columns are used to retrieve specific record from the duplicate records.

**SYN:** select rowid, rownum from <Table name>

**EX:** select rowid,rownum from emp;

**Syntax for retrieve specific rowid:**

**SYN:** SELECT \* FROM <Table name> WHERE ROWID LIKE ‘% ROWIDVALUE’

We can’t retrieve row num through like command. We can retrieve only row id is possible.

**EX:** select \* from emp where rowid like '%C';

(sysdate, current\_date these are also psudo columns)

**Group by clause:** This clause is used to divide the values depending on it’s a true.

Group by clause always used along with group functions.

**EX:** select deptno,count(\*),sum(sal),max(sal),min(sal),avg(Sal) from emp group by deptno;

**HAVING**

This will work as where clause which can be used only with group by because of absence of where clause in group by.

**EX:** select deptno,count(\*),sum(sal),max(sal),min(Sal) from emp group by deptno having count(\*)>5;

**EX:** select deptno,count(\*),sum(sal),max(sal),min(sal),avg(sal) from emp where deptno=20 group by dept no;

Here we use where clause but we used where clause before the group by.

**Sequences:**

* Sequence is a database object which automatically generates unique numeric values on a particular column in the table. Sequence default values start with 1 and it is incremented by +1 up to hard disk capacity.
* Sequence always generates numeric values we can apply the same sequence on more than one column in sequences on different columns in the same table.
* Sequence never generates character values.
* Sequence generates positive values ascending order and descending order.
* Sequence generates negative values ascending order and descending order.
* Sequence mechanism we can’t apply an alpha numeric columns.

**Syntax for sequence**

First we should create sequence.

**SYN:** create sequence <sequence name>

**EX:** create sequence s1;

Syntax to applying the sequence on a particular column in the table

**SYN:** insert into<table name> values ( sequence.nextval,’&columnname’,’&columnname’);

**EX:** insert into emp1 values (s1.nextval,’&ename’,&sal);

In the above example nextval represents psudo column and it has two advantages.

1. It initiates the sequence number and also it gives the next value of the sequence.

**Syntax to know about the current value of the sequence**

Select sequencename.currval from dual

**EX:** select s1.currval from dual;

In the above example currval represents psudo column and it gives the current value of the sequence.

**To apply the syntax on other table:**

insert into emp2 values(s1.nextval,'&ename','&sal');

**Creating a sequence with starting ending incrimination value:**

**SYN:** create sequence <sequence name>

minvalue<integer>

maxvalue<integer>

Incremented by integer

**EX:** create sequence s2

minvalue 101

maxvalue 110

incremented by 1;

**Note:** If sequence reached MAX value then we will get error message.

**Syntax to alter the sequence:**

**SYN** Alter sequence s1

Maxvalue integer

Increment by integer;

(Here it can’t possible modify min value)

**EX:** alter sequence s1

maxvalue 120

increment by 2;

**Example for applying the sequence more than one column on same table.**

insert into emp2 values(s1.nextval,'&ename',s1.nextval);

**Example for creating sequence with descending order**

create sequence s3

minvalue 1

maxvalue 10

increment by -1;

**NOTE:** Here it takes first maximum value next it takes minimum value.

**Example for creating the sequences with the negative values**

create sequence s4

minvalue -5

maxvalue -1

increment by1

**Example for creating the sequence with the negative values with positive increments:**

create sequence s5

minvalue -5

maxvalue -1

increment by -1;

**Example for applying for the two different sequences on two different columns in the same table.**

insert into emp1 values(s4.nextval,'&ename',s5.nextval);

**How to see the list of all sequences**

**EX:** select \* from user\_sequences;

|  |
| --- |
| Data dictionary tables are user\_tab\_columns, recyclebin, user\_sequences |

**Creating the sequence with the cycle option:**

**EX:** create sequence s8

minvalue 10

maxvalue 15

cycle nocache;

**OUTPUT:**

select \* from rose;

NO NAME SAL

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

1 siva 1

2 bhargav 2

3 reddy 3

11 rohit 11

45 ravi 45

44 sonal 44

43 vinamra 43

10 vishal 10

12 narad 12

13 saradh 13

14 bharath 14

NO NAME SAL

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

15 suneel 15

10 isha 10

11 chawla 11

**NOTE:** If we give cycle no cache then we will get sequences on columns with cycle.

That means if minvalue 10 and maxvalue is 15 then after entering five records it is not possible to enter but here we need not bother about limit. It will automatically take from min value.

**Syntax to drop sequence**

**EX:** drop sequence s2;

|  |
| --- |
| Q) Why we should not drop Table by Alter?  A) Because table is database object but column is not a database object. |

**JOINS:**

1. This is a mechanism which is used to combine or add one or more than one table.
2. Joins are classified into following types those are
3. Equi join/simple join/ outer join
4. Natural join
5. Non – equi join
6. Outer join

It again divided into three types those are

1. Left outer join
2. Right outer join
3. Full outer join
4. Self join
5. Cross join/ Cross product join / Cartesian product join

Whenever we need to get complete and whole information regarding a particular point then we need to join the tables.

1. Simple or equi or Inner join:
2. In this join the matching rows will be displayed as output.
3. In this join we need to use to equal operator in the where clause of the select statement.
4. If we want to perform the join operation at least we require one common column between the tables.
5. In this join whenever the rows are not matching those records will not be displayed.

**SYN:** SELECT \* FROM <tab1>,<tab2>, -----------,<tab n> where <condition>

**EX:** select e1.\*,e2.\* from std e1,std e2 where e1.deptno=e2.deptno;

**EX2:** select e1.\*,e2.no, e2.name from std e1, std e2 where e1.deptno=e2.deptno;

**EX3:** select e1.\*,e2.\* from std e1 inner join std1 e2 on e1.deptno=e2.deptno;

1. **Natural join:**

This join was introduced from Oracle 9i onwards, equi join and natural join both are exactly same as far as output concerned. There are there difference between equi join and natural join.

1. No need to check where condition.
2. No need to mention table name before common column.
3. Common column will be displayed in front of the output.

**EX:** select \* from emp natural join ksr;

**NOTE:** for example two tables contain same structure(eno,ename,marks) then its not possible to do natural join.

1. **Non equi join:** In this join we need to use only relational operator such as >,<,>=,<=,!= except ‘=’ operator. The main advantage of non equi join is even through there is no common column. We can perform the join operation.

**EX:** select \* from manu m1, manu2 m2 where m1.no>=m2.no;

**EX:** select \* from manu m1, manu2 m2 where m1.no>=m2.no and m1.deptno!=m2.deptno;

**OUTER JOINS:** In equi join there might be a chance of losing information to recover the lossed information we need to use outer joins. Outer joins are broadly divide into three types.

1. Left outer joins
2. Right outer joins
3. Full outer joins
4. **Left outer join:** This join is the combination of equi join operation plus lossed information from the left hand side of the table.

**EX:** select \* from manu m1 left outer join manu1 m2 on m1.deptno=m2.deptno;

**Oracle 8i model Leftouter joins:**

select \* from manu m1, manu1 m2 where m1.deptno=m2.deptno(+);

**Right outer joins:** The join is the combination of equi join operation + lossed information from the right hand side table.

**EX:** select \* from manu m1 right outer join manu1 m2 on m1.deptno=m2.deptno;

**Oracle 8i model Right outer join:**

select \* from manu m1, manu1 m2 where m1.deptno(+)=m2.deptno;

**Full outer join:**  This join is the combination of equi join operation + lossed information from lefy hand side + lossed information from right hand side table.

**EX:** select \* from manu full outer join manu1 on manu.deptno=manu1.deptno;

**Oracle 8i model:**

select \* from manu m1, manu1 m2 where m1.deptno=m2.deptno(+)

union

select \* from manu m1, manu1 m2 where m1.deptno(+)=m2.deptno;

**V. Self join:** Joining a table by itself known as self join. Whenever we have relationship between two columns in the same table then we need to use self join.

**EX:** select e1.empno,e2.mgr from emp e1,emp e2 where e1.empno=e2.mgr;

**No use in real time:**

**V1. Cross join:**  Cross join is the product of two or more than two tables it means that m no of rows in one table n no of tables in another table then we will get the product of mXn.

**EX:** select \* from emp1 cross join dept1;

**TCL:** This is the third sub language in SQL which contains the commands

1. Commit
2. Rollback
3. Save point

**Transaction:**  Any operation that can perform by using DML commands is known as Transaction.

**Session:** It can e be defined as some interval of time or some span of time. It means that the moment the user log on to the database to the user logoff to the database this span of the time is known as session. We can terminate the session two ways

1. Nornal Termination
2. Abnormal Termination
3. **Normal Termination:** Terminating the session by typing exit or quit SQL prompt then this termination can be called as Normal Termination. Normal Termination always saves our Transactions.
4. **Abnormal Termination:** Terminating the session by directly close the window button or directly switch off the C.P.U or power off.

Abnormal Termination does not save our transaction.

**Commit:** This command is used for save the transactions explicitly from the moment by user logon to the database to till execute this command.

**SYN:** Commit; Then commit will complete.

**Rollback:** This command is used to discard the transactions from the moment the user logon to the database till execute this command or executing the commit command too .

**Syn:** Rollback; Rollback complete

**Save point:** save point is reference name for a particular set of transactions.

**SYN:** savepoint <savepoint name>

**EX:** savepoint s3;

**To retrieve syntax is**

**EX:** savepoint to <save point name>

**EX:** rollback to savepoint s1;

**Note:** We can’t rollback the committed transactions but we can commit the rollback transactions.

**Constraints:** Constraint is the mechanism which is used to restrict the invalid data which is enters by the end user by implementing business rules we can apply the constraints on two sitchuation on the table.

1. During the creation of the table.
2. After creation of the table.

Constraints have two names

1. System defined Constraints names
2. User defined constraints names
3. **System defined Constraint names:**  A constraint name which is defined by the system internally those constraints can be called as system defined constraint names. System defined constraint names are always in the format of SYS\_CN.Here n represents any no.
4. **User defined Constraints:** The constraint name which is defined by the user manually those means those names can be called as user defined constraint names. We can apply constraints in two levels.
5. Column level constraints
6. Table level constraints
7. **Column level Constraints:** Applying the constraints after defining the column immediately those constraints can be called as column level constraints.
8. **Table level constraints:** Applying the constraints after defining all the columns in the table or at the end of the table those constraints can be called as table level constraints.Constraints are classified into six types.
9. Unique
10. Not null
11. Check
12. Primary key
13. Foreign key
14. Defult

**Unique constraint:**

1. This constraint doesn’t allow us to enter duplicate values of a particular column in the table.
2. We can apply constraint more than one column in the table.
3. We can apply this constraint more than one column in the same table.
4. This constraint allows us to enter null values.
5. **Unique key:**

Unique key is a constraint. It never allows duplicate values. But it allows null values.

We can give constraint in two levels

1. Table level constraints.
2. Column level constraints.

**Example for Column level constraints:**

create table work1 (empno number(10) unique, ename varchar(10), sal number(10));

**Example for Table level constraints:**

create table work2(empno number(10),ename varchar(10), sal number(10), unique(empno));

**Constraint violation**

unique constraint (SCOTT.SYS\_C005251) violated.

1. **Not Null:** Not null is a constraint which never allows null values but allows duplicates. We can apply this constraint in column level only.

**Column level Ex:** create table work3 (empno number(10) not null, ename varchar(10), sal number(10));

**Table level Ex;** create table work4 (empno number(10) ,ename varchar(10),sal number(10) not null);

1. **Primary key:** It never allows duplicates and null values.

**Column level:** create table work6 (empno number(10) primary key,ename varchar(10),sal number(10));

**Table level**: create table work7 (empno number(10),ename varchar(10),sal number(10), primary key(empno));

1. **Check:** Which is used to check particular constraint.

**EX:** create table work8 (empno number(10),ename varchar(10),sal number(10) check (sal=3000);

1. **Foreign key Constraint:** Using this constraint we can maintain the relation between the tables with in the database. This constraint can also be called as referential integrity. If there is any dependencies on the parent table that is dept then we are not suppose to delete the parent record from the parent table. If there is no parent values in the parent table. We can’t enter the values within the child table that is employee table.

**Super keys and Candidate keys:**

**Definition:**A superkey is a combination of attributes that can be uniquely used to identify a database record. A table might have many superkeys. Candidate keys are a special subset of superkeys that do not have any extraneous information in them.

**Examples:**

Imagine a table with the fields <Name>, <Age>, <SSN> and <Phone Extension>. This table has many possible superkeys. Three of these are <SSN>, <Phone Extension, Name> and <SSN, Name>. Of those listed, only <SSN> is a candidate key, as the others contain information not necessary to uniquely identify records.

**EX:** create table child1(no number(10),sal number(10),constraint bhargav foreign key(sal) references

Child(no));

**NOTE:**

**1.** We should give the foreign key relationship to another table which has already consist primary ky.

2. We can give the foreign key relationship to parent table’s primary key initiated column only.

**ON DELETE CASCADE:** This option is used to delete the parent “record” from the parent table even though there are some dependencies.

**NOTE:** It is possible to give only at the time of creating foreign key constraint.

**EX:** create table baju1 (no number(10), foreign key(no) references baju(no) on delete cascade);

**Default:** This constraint is used to give default value into a particular column in the table.

**EX:** create table baju2 (no number(10) default 111);

**Example for applying the constraint after creating the table:**

alter table baju1 modify(no number(10) unique);

**Note:** Here we can modify previous table constraints only.

**Syntax to see the list of the Constraints:**

select \* from user\_cosntraints;

Desc user\_constraints;

**Syntax to see the list of constraints on a particular table**

select constraint\_name,constraint\_type,status from user\_constraints where table\_name='BAJU1';

CONSTRAINT\_NAME C STATUS

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

SYS\_C005281 R ENABLED

SYS\_C005282 U ENABLED

**Create User Defined Constraint names:**

create table iron(no number(10) constraint rohit primary key,name varchar(10));

select Constraint\_name,Constraint\_type,status from user\_constraints where table\_name='IRON';

CONSTRAINT\_NAME C STATUS

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

ROHIT P ENABLED

**NOTE:** It shows C as constraint for check and not null but first priority for not null.

**Syntax to drop/enable/disable the Constraint:**

**SYN:** ALTER TABLE<Table name> DISABLE/ENABLE/DROP CONSTRAINT CONSTRAINT\_NAME

**EX:** alter table iron disable constraint rohit;

**EX:** alter table iron enable constraint rohit;

**EX:** alter table iron drop constraint rohit;

**Composite Primary key:** Composite primary key means share primary key properties to more than one column at a time in the same table.

**EX:** create table vinod(no number(10),name varchar(10),sal number(10),constraint pse primary key(no,name,sal));

**NOTE:** Here we can insert same values into columns, different values but do not give null value. But we can give duplicate values.

**To copy a table from one table to another table:**

**SYN:** CREATE TABLE<Target table name> AS SELECT \* FROM <Source Table name>

**EX**: create table ranga as select \* from vinod;

**Syntax to copy a table to another table with specific columns only**

CREATE TABLE TABLE NAME<Target table name> as select col1,col2,-------,coln from <source table name>

**EX:** create table ranga1 as select no,name from vinod;

**Copying a table from one table to another table without data:**

CREATE TABLE<Target table name> as select \* from <Source table name> where <false condition>

**EX:** create table ranga3 as select \* from vinod where 1=2;

Name Null? Type

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

NO NUMBER(10)

NAME VARCHAR2(10)

SAL NUMBER(10)

**Copying a table from one table to another table without data with specific columns:**

**SYN:** CREATE TABLE<Target table name> as select col1,col2,-----------,coln from <Source table name> where <false condition>

**EX:** create table ranga4 as select no,name from vinod where 1>2;

**Insert the data from one table to another table:**

**SYN:** insert into elora1 select \* from elora;

**NOTE:** We can insert values from one table to another table the table contains same structure and same data types. Otherwise we can’t insert.

**DCL:** This is the fourth sub language in Sql contains two commands 1. Grant 2. Revoke.

**Syn:** CREATE USER<User name> IDENTIFIED BY <Password>

**Grant:** This command is used to giving the privileges and roles to a normal user.

**Privilege:** It can be defined as permission or a simple command such as

Create table, create any table, alter table, alter any table.

**Role:** It can be defined as collection of privileges or permissions. It can be classified into two types

1. System defined roles
2. User defined roles
3. **System defined roles:**  A role which is already constructed along with any software those roles can be called as system defined roles.

**EX:** Connect, resource, dba

**User defined roles:** A role which is defined by user manually those roles can be called as user defined roles.

**EX:** display\_data.

SQL> show user

USER is "SCOTT"

SQL> connect as sysdba

Enter user-name: sysdba

Enter password:

Connected.

SQL> show user

USER is "SYS"

SQL> create user bhargav identified by siva ;

User created.

**Grant:** grant role1, role2, privilliege1, privilliege2,--------,grant n, privilege n to <user name>

**EX:** grant connect to bhargav;

**EX:** grant resource to bhargav;

**To give permission to access other user data;**

**EX:** grant select on scott.emp to bhargav;

**EX:** grant update, delete,insert on scott.emp to bhargav;

**EX:** grant all on scott.emp to bhargav;

**Revoke:** This command is used to take back the permissions from a normal user.

**EX:** revoke all on scott.emp from Bhargav;

**Grant privileges create INDEX, VIEW, and SYNONYM:**

**EX:** Grant create any index to SCOTT;

Grant succeeded.

**EX:** Grant create any view to SCOTT;

Grant succeeded.

**EX:** grant create materialized view to scott;

**EX:** Grant Create any Synonym to scott;

Grant succeeded.

**Note:** Here any means we can create particular database object on any user account.

**Syntax for password change:**

Changing password for Bhargav

Old password: Siva

New Password: Reddy

**Syntax to see the list of privileges:**

Select \* from session\_privs;

**To see the users:**

Select \* from dba\_users

**To show user\_id:**

Select \* from dual;

**To lock account:**

Alter user Bhargav account lock/unlock;

**Syntax to drop user account:**

**EX:** Drop user username cascade;

Drop user Prakash

**Note:** If user account contains no data we need not to cascade otherwise cascade.

**Sub queries:**

A query contains another query is known as nested query or sub query. We can write query into a query upto 256 queries.

In between the outer query and inner query, inner query executed first then it will execute the outer query .

Outer query is always executed descending on inner query result.

Sub queries are classified into two types.

1. Single row sub queries.
2. Multi row sub queries.
3. **Single row sub query:**
4. If the inner query returns only one value then those queries can be called as single row sub query are equal to >,<,>=,<=,!=.
5. These operators can be called as single row sub queries.
6. Inner query returns only one value in two classes.

**Case (i):** If we are using any group or aggregate functions.

**Case (ii):** If we are using any group or aggregate functions any group or aggregate functions.

**Multiple row sub queries:**

* If the inner query returns more than one value then those queries can be called as multiple row sub querites.
* The operators which are using in multiple row sub queries are IN, ANY, ALL, SOME.

**Note:**

* Here ANY and SOME both are same.
* These two operators can be called as multiple row operators.

Q) Write a query display the employee details whose salary is less than 7788.

A) select \* from emp where sal<(select sal from emp where empno=7788);

Q) Write a query display the employee details whose salary is less than 7788 working under 30th dept.

A) select \* from emp where sal<(select sal from emp where empno=7788) and deptno=30;

Q) Write a query display the employee details whose salary is less than maximum salary.

A) select \* from emp where sal<(select max(sal) from emp);

Q) Write a query display the employee details whose salary is less than maximum salary and who are having the commission.

A) select \* from emp where sal<(select max(sal) from emp) and comm is not null;

Q) Write a query display the employee details whose salary is greater than maximum salary.

A) select \* from emp where sal > (select max(sal) from emp);

Q) Write a query display the employee details who are working under BLAKE.

A) select \* from emp where mgr=(select empno from emp where ename='BLAKE');

Q) Write a query display the employee details who are working sales dept.

A) select \* from emp where deptno=(select deptno from dept where dname='SALES');

Q) Write a query display maximum salary details who are working under 10th dept.

A) select max(sal) from emp where sal=(select max(sal) from emp where deptno=10)

Q) Display the employee details whose salary is less than maximum total salary in descending order ename wise.

A) select \* from emp where sal<(select max(sal+nvl(comm,0)) from emp) order by ename desc;

Q) write a query display maximum total salary who were join in the year 81.

A) select \* from emp where sal=(select max(sal+nvl(comm,0))from emp where hiredate like '%81');

Q) Write a query count the no of employees working in the ‘CHICAGO’

A) select count(\*) from emp where deptno=(select deptno from dept where loc='CHICAGO');

Q) Find out total salary of maximum salary of employee working in the Sales Department.

A) select sal+nvl(comm,0) from emp where sal=(select max(sal) from emp where

deptno=(select deptno from dept where dname='SALES'));

Q) Display the employee details whose experience is more than minimum experience employee.

A) select \* from emp where (sysdate-hiredate)/365 <(select min(sysdate-hiredate)/365 from emp )

Q) Display the employee details working under the newyork and their name contains the letter ‘A’

A) select \* from emp where deptno=(select deptno from dept where loc='NEW YORK') and ename like '

%A%';

Q) Write a query display the third maximum salary of employee.

A) select \* from emp where sal=(select max(sal) from emp where sal <(select max(sal)

from emp where sal<(select max(sal) from emp)));

Q) Write a query display the maximum salary of employee name whose name ends with n.

A) select ename from emp where sal=(select max(sal) from emp where ename like '%N');

Q) select emp.\*,sal+nvl(comm,0) from emp where deptno=(select deptno from dept where dname='RESEARCH’;

**SET OPERATORS:** There are 4 set operators in SQL those are UNION, UNION ALL, INTERSECT and MINUS.

**UNION:** This operator returns all the values from all the tables excluding duplicate values.

**EX:** select deptno from emp union select deptno from dept;

**UNION ALL:** This operator returns all the values from all the tables including duplicate values.

**EX:** select deptno from emp union all select deptno from dept;

**INTERSECT:**

**EX:** select deptno from emp intersect select deptno from dept;

**MINUS:**

**EX:** select deptno from emp minus select deptno from dept;

Q) Write a query display the last three records in the table.

A) select \* from emp where rownum <=(select count(\*) from emp) minus select \* from emp where rownum<=(select count(\*) from emp)-3;

Q) To Display the first three records.

A)select \* from emp where rownum<=3;

Q) To display the fifth record.

A) select \* from emp where rownum=5;

Q) to display the records dynamically.

A) select \* from emp where rownum<=&rownum minus select \* from emp where rownum<=&rownum;

**Multiple Row Sub queries:**

Q) Write a query display the employee details who are working under sales and accounting department.

A) select \* from emp where deptno in(select deptno from dept where dname in('ACCOUNTING','SALES'));

Q) Write a query display the maximum salary employee details working under sales and accounting department.

A) select \* from emp where sal=(select max(sal) from emp where deptno

in(select deptno from dept where dname in('SALES','ACCOUNTING')));

Q) Write a query count the no of employees working in BOASTON and NEWYORK.

A) select count(\*) from emp where deptno in(select deptno from dept where loc in('BOASTON','NEWYORK'));

Q) Write a query display the employee details whose salary is less than any employee salary working under 20th dept.

A) select \* from emp where sal<any(select sal from emp where deptno in 20);

(OR)

1. select \* from emp where sal<some(select sal from emp where deptno in 20)

**Note:** ANY and SOME are same.

Q) Write a query display the employee details who are working under BLAKE and BHARGAV.

A) select \* from emp where mgr in(select empno from emp where ename ='BLAKE' or ename = 'VISHAL');

Q) write a query display the maximum total salary of employees who are working under ‘BLAKE’ and ‘VINAY’.

A) select \* from emp where (sal+nvl(comm,0))=(select max(sal+nvl(comm,0)) from emp where mgr in(select empno from empwhere ename='BLAKE' or ename ='VISHAL'));

Q) Write a query display the employee details whose salary is greater than all employees’ salaries who are working under 20th dept.

A) select \* from emp where sal>all(select sal from emp where deptno=20)

Q) Write a query display the employee details who are comes under grade1 and grade2.

A) select emp.\*,grade from emp,salgrade where sal>=losal and sal<=hisal and grade in

(select grade from salgrade where grade in (1,2));

Q) Display the maximum salary employee details working under 10th and 20th dept.

A) select \* from emp where sal in(select min(sal) from emp where deptno in(20,30))

Q) Write a query display the employee details of the department table available under scott user.

A) select \* from emp where exists(select \* from tab where tname='DEPT');

Q) Write a query display the employee details if VIBHAV table not available under SCOTT user.

A) select \* from emp where not exists(select \* from tab where tname='VIBHAV');

**Synonym:** Synonym is a database object. It is a shortcut name for original base table and it contains the physical representation of data.

1. If we create physical representation of data synonym will be created on the entire table, and it is not possible to create a synonym based on the partial table and subset of a table.
2. We can’t create a synonym more than one table.
3. If we perform any DML operation on synonym it automatically shows effect on base table also. Like that if make any modification on base table it automatically modify the synonym.
4. If we drop the base table the corresponding synonyms won’t be dropped and those become invalid.
5. Synonyms will become invalid in two cases.
6. When we drop the base table.
7. When we change the table name.

On invalid synonym It is not possible to perform any type operations. We can make the invalid synonym as valid synonym we can create a synonym with a base table without a base table.

1. On invalid synonym if the user trying to perform the operations then we will get the following error message.

Synonym translation is no longer valid.

1. Synonyms are classified into two types.
2. Private synonyms.
3. Public synonyms.
4. **Private Synonyms:** Private synonyms are synonyms those are confined to only one particular user and these synonyms are access by one particular user.
5. **Public synonyms:** Public synonyms are synonyms those are accessed by many users at a time those synonyms can be called as public synonyms.

Public synonyms always created by DBA.

**SYN:** create synonym <synonym name> for < table name>

**Syntax for creating a synonym:**

CREATE SYNONYM<synonym name> FOR <table name>

**EX:** create synonym v for EMP;

**Public synonyms:**

**SYN:** create public synonym <synonym name> for <table name>

**EX:** create public synonym ps for po;

**Syntax to drop particular synonym:**

**Syn:** DROP SYNONYM <Synonym name>

**EX:** drop synonym B;

**NOTE:**  we can’t modify the structure of the synonym.

**VIEWS:**

* 1. View is a database object which contains logical representation of data.
  2. By using views we can hide some column data from the user.
  3. View does not occupy the physical memory.
  4. We can create a view based on entire table or partial table or subset of table.
  5. If we perform any type of operations view the same operations automatically affected to the corresponding base table and vice – versa.
  6. We can create a view based on another view upto maximum 32 views.
  7. If we drop the base table the corresponding views will not be dropped and those become invalid.
  8. On invalid views we can’t perform any type of operations.
  9. Views will become invalid in three cases.
     1. When we drop the base table.
     2. When we change the table name.
     3. When we modify the structure (Rename) of the base table.
  10. We can make an invalid view as a valid view.
  11. We can create a view based on one table and also based on the more than one table.
  12. Views are divided into two types.

1. Simple views
2. Complex Views.
3. **Simple Views:** Creating the view based only one table is known as simple view.
4. **Complex View:** Creating the view based on more than one table is called Complex view.

|  |  |
| --- | --- |
| **View** | **Synonym** |
| 1. It is logical representation of data. 2. We can create a view based on the partial table. 3. We can create view more than one table. 4. View will become invalid in three cases. | 1. It is physical representation of data. 2. We can’t create a synonym on partial table. 3. We can’t create a synonym based on more than one table. 4. Synonym is invalid in two cases. |

**NOTE:** We can create a view based on the synonym and also we can create a synonym based on the view.

**Syntax to create view:**

CREATE VIEW <View Name> as SELECT \* FROM <Table Name>

**EX:** create view bn as select \* from lak;

**Create a view base on Partial table:**

**SYN:** Create View <View name> a s select col1, col2 , -------- ,col n from <Table Name>

**EX:** create view see as select ename, sal from emp1;

**Read Only View:** It is specific type of the view. So we can’t perform any type of operations of DML operations.

**EX:** create view kpp as select \*from emp with read only;

**Syntax to create a view without a base table:**

**SYN:** create force view <View Name> as select \* from <Table name>

**EX:** create force view fv as select \* from aaa;

**Warning:** View created with compilation errors.

**NOTE:** If we create a synonym for one table then we can’t create view for it and also same thing applicable for reverse.

**Complex Views:** Creating the views based on more than one table is known as complex views.

**Syn:**  CREATE VIEWE<View Name> AS SELECT \* FROM <tab1>,<tab2>-----,<tab n>

**EX:** create view cv as select \* from emp,dept where emp.deptno=dept.deptno;

**NOTE:** Here we will get on error message. According to RDBMS rules no database object should not contain duplicate columns.

The above example returns the following error message.

**NOTE:** on complex views we can’t perform any type of DML operations.

**EX:** create view cv as select \* from emp natural join dept;

**Materialized views:** These are special type of views the main difference between normal view and materialized view is if we drop the base table it won’t invalid. Before creating the materialized view user need to remember two points.

1. He need to take permission from DBA.
2. On which table we need to create the materialized view that table should contain primary key Constraint.
3. Materialized view stores data.
4. We can’t perform DML Operations on Materialized view.

**EX:** create materialized view dv as select \* from emp;

**Indexes:** Index is a database object which is used to retrieve the data quickly from esisting table. Indexes are classified into two types those are

1. **Unique Index:** If we apply any primary key Constraint or unique Constraint or any particular column on the table oracle engine internally maintains one of the index mechanisms called UNIQUE Index.
2. **Non Unique index:** An index mechanism which is performed by the user manually those indexes are can be called as non unique indexes. Whenever we apply an index mechanism on any particular column in the table then Oracle engine internally maintains a separate table called index table which maintains chain index relationship between the regular table and normal table.

**Syntax To create an index:**

CREATE UNIQUE INDEX<Index name> AS TABLE NAME <Column name>

**EX:** create unique index ix on leela(no);