Contents

[Introduction of DBMS 2](#_Toc88608300)

[DATA 2](#_Toc88608301)

[DATA BASE 2](#_Toc88608302)

[DATA-BASE MANAGEMENT SYSTEM (DBMS) 2](#_Toc88608303)

[Types of DBMS 2](#_Toc88608304)

[Arithmatic Operator – 9](#_Toc88608305)

[Use of “-”Arithmetic Operator 9](#_Toc88608306)

[Use of “\*”Arithmetic Operator:- 10](#_Toc88608307)

[Use of “/”Arithmetic Operator:- 10](#_Toc88608308)

[Literals 11](#_Toc88608309)

[ Dual 12](#_Toc88608310)

[Concatenation 12](#_Toc88608311)

[Orderby 15](#_Toc88608312)

[RELATIONAL OPERATOR 17](#_Toc88608313)

[LOGICAL OPERATOR 17](#_Toc88608314)

[SPECIAL OPERATOR 19](#_Toc88608315)

[BETWEEN 20](#_Toc88608316)

[LIKE(Pattern matching) 21](#_Toc88608317)

[NULL 23](#_Toc88608318)

[FUNCTION 25](#_Toc88608319)

[Replace 27](#_Toc88608320)

[Multirow Function 32](#_Toc88608321)

[HAVING 37](#_Toc88608322)

[SUB-QURIES 38](#_Toc88608323)

[JOINS 40](#_Toc88608324)

[Note: 41](#_Toc88608325)

[DDL 42](#_Toc88608326)

[ALTER 44](#_Toc88608327)

***SQL CONTENTS: -***

* **Introduction of DBMS**
* **Types of DBMS**
* **DQL (Data Query Language)**
* Joint, Sub- Queries, Co- Related Sub-Queries
* **DDL (Data Definition Language)**
* Create, Alter, Rename, Drop, Truncate
* Data Type, Constraints
* **DML (Data Manipulation Language)**
* Insert, Update, Delete
* **TCL (Transaction Control Language)**
* Commit, Same point, Roll Back, Flashback, Purge
* **DCL (Data Control Language)**
* Grant, Revoke
* **Functions**
* Single Row Functions
* Multi Row Functions
* Data Functions
* **Normalization (1 NE, 2NE, 3NE)**

# Introduction of DBMS

## DATA

It may be defined as the useful information & all the data are stored in a centralized location is known as data.

DATA BASE

It is defined as a useful information which is organized by certain set of rules is called database.

DATA-BASE MANAGEMENT SYSTEM (DBMS)**-**

It is defined as a **collection of programs** which is written in order to **manage data base effectively** is known as DBMS.

# Types of DBMS

1. Flat file DBMS
2. Heretical DBMS
3. Network DBMS
4. **RDBMS (Relational data-base management system)**

* **RDBMS (Relational data-base management system)-**

It is defined as **collection of data** which is **organized** by following **some relational rule** is called RDBMS.

In this type **data** will be **store in the form of tables** and its is also easy to **relate multiple tables** to fetch an output.

**Store Data with Rules and Table**

* **Data Base Vendor-**
* Vendor are those who manufactures the database & sells it is called Data Base Vendor.

|  |  |
| --- | --- |
| **Name of Vendor** | **Product** |
| ORACLE | 10G,11G,12G |
| SAP | Sybase |
| IBM | DB2 |
| Open Source | My SQL |

* **ENTITY- Each and every table** is called entity.

Example--->

**Entity**

Table name- **Student**

**Column Name/ Table Structure**

**Record /Data**

|  |  |  |
| --- | --- | --- |
| **S\_ID** | **S\_Name** | **Course** |
| 01 | Ram | SQL |
| 02 | Sayam | JAVA |

**Show All table Name in SQL Code: -**

* To display tables names in the any SQL database then we have to use below query

Syntax- **Select \* from tab;**

O/p- TNAME TABTYPE CLUSTERID

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

DEPT TABLE

EMP TABLE

BONUS TABLE

SALGRADE TABLE

**Display column name in SQL Code: -**

To display column names from any table in a SQL database then we have to use below query

Syntax- **desc tablename;**

Example- ***desc emp;***

O/p-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Column** | **Data Type** | **Length** | **Precision** | **Scale** | **Primary Key** | **Nullable** | **Default** | **Comment** |
| [EMP](javascript:ret_Column('WKSP_KURESH02.EMP');) | [EMPNO](javascript:ret_Column('EMPNO');) | NUMBER | - | 4 | 0 | 1 | - | - | - |
|  | [ENAME](javascript:ret_Column('ENAME');) | VARCHAR2 | 50 | - | - | - | nullable | - | - |
|  | [JOB](javascript:ret_Column('JOB');) | VARCHAR2 | 50 | - | - | - | nullable | - | - |
|  | [MGR](javascript:ret_Column('MGR');) | NUMBER | - | 4 | 0 | - | nullable | - | - |
|  | [HIREDATE](javascript:ret_Column('HIREDATE');) | DATE | 7 | - | - | - | nullable | - | - |
|  | [SAL](javascript:ret_Column('SAL');) | NUMBER | - | 7 | 2 | - | nullable | - | - |
|  | [COMM](javascript:ret_Column('COMM');) | NUMBER | - | 7 | 2 | - | nullable | - | - |
|  | [DEPTNO](javascript:ret_Column('DEPTNO');) | NUMBER | - | 2 | 0 | - | nullable | - | - |
|  |  |  |  |  |  |  |  |  |  |

**Display any particular table information in SQL Code: -**

* To display any particular table information in a SQL database then we have to use below query

Syntax- **Select \* from tablename;**

Example- ***Select \* from emp;***

O/p-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
| 7839 | KING | PRESIDENT |  | 11/17/1981 | 5000 |  | 10 |
| 7698 | BLAKE | MANAGER | 7839 | 05-01-1981 | 2850 |  | 30 |
| 7782 | CLARK | MANAGER | 7839 | 06-09-1981 | 2450 |  | 10 |
| 7566 | JONES | MANAGER | 7839 | 04-02-1981 | 2975 |  | 20 |
| 7788 | SCOTT | ANALYST | 7566 | 12-09-1982 | 3000 |  | 20 |
| 7902 | FORD | ANALYST | 7566 | 12-03-1981 | 3000 |  | 20 |
| 7369 | SMITH | CLERK | 7902 | 12/17/1980 | 800 |  | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 02/20/1981 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 02/22/1981 | 1250 | 500 | 30 |
| 7654 | MARTIN | SALESMAN | 7698 | 09/28/1981 | 1250 | 1400 | 30 |
| 7844 | TURNER | SALESMAN | 7698 | 09-08-1981 | 1500 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 01-12-1983 | 1100 |  | 20 |
| 7900 | JAMES | CLERK | 7698 | 12-03-1981 | 950 |  | 30 |
| 7934 | MILLER | CLERK | 7782 | 01/23/1982 | 1300 |  | 10 |

**Display multiple column names from a table information in SQL Code: -**

* To display multiple column names from any table in a SQL database then we have to use below query

Syntax- **Select column1name, column2name from tablename;**

Example- ***Select ename, job from emp;***

O/p-

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

**\*Note:-**

1. We can’t use other column name which in not present in the table from which we are trying to retrieve data .

(Ex- If we searching for dept. name in emp table then we can’t find it because dept. name information is present in the dept. table only)

I/p- *Select dept from emp;*

O/p- Error ….Dept column not present in the emp table

1. SQL in not a case-sensitive language.
2. Knowledge Table name and Column name is mandatory.
3. Table names and Column name need to be use as it is present in the table
4. SQL end with ;

* **Aliasing -**

**Display Aliasing column in SQL Code: -**

* Giving another name for an existing column name for a table is called aliasing.

Syntax :- **select columnname as aliasing from tablename;**

**Example: - *select* *sal as salary from emp;***

O/p-

|  |
| --- |
| **SALARY** |
| 5000 |
| 2850 |
| 2450 |
| 2975 |
| 3000 |
| 3000 |
| 800 |
| 1600 |
| 1250 |
| 1250 |
| 1500 |
| 1100 |
| 950 |
| 1300 |

**Example: - *select* *salary from emp;***

O/p- Error at Line 1

“Salary”: invalid Identification.

**Note-**

|  |
| --- |
| **SALARY** |
| 800 |
| 900 |
| 1200 |

1.  Aliasing is not a permanent change. It is used for just to display purpose for output. -
2. By default, aliasing will take “as” keyword.

**Example: - *select* *sal salary from emp;***

|  |
| --- |
| ***SaLaRy*** |
| 800 |
| 900 |
| 1200 |

 **Note-** To Display the aliasing name as it is given in the query then we have to use “aliasingname”.

**Example: - *select* *sal “SaLaRy” from emp;***

**Display Aliasing a table in SQL Code: -**

* Giving another name for an existing table name to a old table is called aliasing.

Syntax :- **select \* from table\_name aliasing\_name;**

**Example: - *select* *from emp e;***

O/p-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Emp No** | **Ename** | **Job** | **Mcr** | **HireDate** | **Sal** | **Comm** |
| 7867 | Tuna | Cleark | 7402 | 20-8-2020 | 800 | 300 |
| 8765 | Muna | Manager | 7562 | 23-8-2020 | 1600 | 500 |

# Red Foam HandArithmatic Operator –

1. Red Foam Hand+ Addition
2. Red Foam Hand- Substruction
3. Red Foam Hand\* Multiplication
4. Red Foam Hand/ Division
5. Modl Modulus

On select statement we can able to perform Arithmetic Operation.

**Example- Write a query to display employee names and increase the salary of 100 rupees?**

O/p- ***Select Ename, sal+100 from emp;***

|  |  |
| --- | --- |
| **Ename** | **sal** |
| Smith | 800 |
| Allen | 1600 |

|  |  |
| --- | --- |
| **Ename** | **sal+100** |
| Smith | 900 |
| Allen | 1700 |

Original Table

Using Arithmatic Operator “+” Table

## Use of “ - ”Arithmetic Operator

**Example- Write a query to display employee names and decrease the salary of 100 rupees?**

O/p- ***Select Ename, sal-100 from emp;***

|  |  |
| --- | --- |
| **Ename** | **Sal-100** |
| Smith | 700 |
| Allen | 1500 |

|  |  |
| --- | --- |
| **Ename** | **sal** |
| Smith | 800 |
| Allen | 1600 |

Using Arithmatic Operator “**-**” Table

Original Table

# Use of “\*”Arithmetic Operator:-

**Example- Write a query to display employee names and their 12 month salary in rupees?**

O/p- ***Select Ename, sal\*12 from emp;***

|  |  |
| --- | --- |
| **Ename** | **Sal\*12** |
| Smith | 9600 |
| Allen | 19200 |

|  |  |
| --- | --- |
| **Ename** | **sal** |
| Smith | 800 |
| Allen | 1600 |

Using Arithmatic Operator “**\***” Table

Original Table

Using Arithmatic Operator”**-**” Table

# Use of “/”Arithmetic Operator:-

**Example- Write a query to display employee names and their 1/10th salary in rupees?**

|  |  |
| --- | --- |
| **Ename** | **sal** |
| Smith | 800 |
| Allen | 1600 |

|  |  |
| --- | --- |
| **Ename** | **Sal/10** |
| Smith | 80 |
| Allen | 160 |

O/p- ***Select Ename, sal/10 from emp;***

Using Arithmatic Operator”**\***” Table

Original Table

**Use of “Modulus“Arithmetic Operator:-**

**Example- Write a query to display salary and show the reminder after divided by 3?**

O/p- ***Select salary, mod(sal,3) from emp;***

If we divide salary with mod operator it will divide salary with the mod number and show the reminder ex- 800/3=2

|  |  |
| --- | --- |
| **Sal** | **Mod(sal,3)** |
| 800 | 2 |
| 1600 | 1 |

|  |  |
| --- | --- |
| **Ename** | **sal** |
| Smith | 800 |
| Allen | 1600 |

Using Arithmatic Operator “ **mod**” Table

Original Table

# Literals

* **Usages of data directly** is called literals.
* Data will be classified into 3 things.

***Numbers***- sal, age, marks ….Example **07, 08, 09**.

***String***- ename, jobs, location….Example **‘Satya’, ‘Ram’, ‘Sayam’**.

***Date***- AD, BC, DOB ….Example **‘07-Dec-2021’**.

* **Note -**
* **Number data** are use **directly**.
* Using **String and Date data** should mention in **single cote** ‘ ’.
* Dual **–**
* Dual is a dummy table which works as a worksheet to do analysis.

**Example** – select 1 from dual;

Dummy Column Name

Attribute

|  |
| --- |
| 1 |
| 1 |

**O/p**-

**Example** – select uma from dual;

Dummy Column Name

Attribute

|  |
| --- |
| uma |
| uma |

**O/p**-

**Example-** select‘ename’ from emp;

|  |
| --- |
| **Ename** |
| Ename |
| Ename |
| ……….. |
| Ename |
|  |

**O/p-**

14 times

Concatenation **–**

* **Joining or merging two or more column values or literals** is called **concatenation**.
* To **achieve concatenation**, we have to **use** (**||**) **double vertical pipes**.

Example- Select ename || job from emp;

O/p-

|  |
| --- |
| **ENAME||JOB** |
| KINGPRESIDENT |
| BLAKEMANAGER |
| CLARKMANAGER |
| JONESMANAGER |
| SCOTTANALYST |
| FORDANALYST |
| SMITHCLERK |
| ALLENSALESMAN |
| WARDSALESMAN |
| MARTINSALESMAN |
| TURNERSALESMAN |
| ADAMSCLERK |
| JAMESCLERK |
| MILLERCLERK |

Write a query to display **my name is ename** from emp table?

**Answer** – *select ‘My name is’ || ename from emp;*

O/p-

|  |
| --- |
| **'MYNAMEIS'||ENAME** |
| My name is KING |
| My name is BLAKE |
| My name is CLARK |
| My name is JONES |
| My name is SCOTT |
| My name is FORD |
| My name is SMITH |
| My name is ALLEN |
| My name is WARD |
| My name is MARTIN |
| My name is TURNER |
| My name is ADAMS |
| My name is JAMES |
| My name is MILLER |

Write a query to display **ename is earning salary** from emp table?

**Answer** – *select ename || ' is earning ' || sal from emp;*

O/p-

|  |
| --- |
| **ENAME||'ISEARNING'||SAL** |
| KING is earning 5000 |
| BLAKE is earning 2850 |
| CLARK is earning 2450 |
| JONES is earning 2975 |
| SCOTT is earning 3000 |
| FORD is earning 3000 |
| SMITH is earning 800 |
| ALLEN is earning 1600 |
| WARD is earning 1250 |
| MARTIN is earning 1250 |
| TURNER is earning 1500 |
| ADAMS is earning 1100 |
| JAMES is earning 950 |
| MILLER is earning 1300 |

Write a query to display **ename is a [job]** from emp table?

**Answer** – ***select ename || ‘is a [‘ || job||’]’ from emp****;*

O/p-

|  |
| --- |
| **ENAME||'ISA['||JOB||']'** |
| KING is a [ PRESIDENT ] |
| BLAKE is a [ MANAGER ] |
| CLARK is a [ MANAGER ] |
| JONES is a [ MANAGER ] |
| SCOTT is a [ ANALYST ] |
| FORD is a [ ANALYST ] |
| SMITH is a [ CLERK ] |
| ALLEN is a [ SALESMAN ] |
| WARD is a [ SALESMAN ] |
| MARTIN is a [ SALESMAN ] |
| TURNER is a [ SALESMAN ] |
| ADAMS is a [ CLERK ] |
| JAMES is a [ CLERK ] |
| MILLER is a [ CLERK ] |

## Orderby

By using ‘orderby’ clause we will be able to rearrange the output in ascending or descending order.

Ex:

Select sal from emp

Order by sal desc;

|  |
| --- |
| **SAL** |
| 5000 |
| 3000 |
| 3000 |
| 2975 |
| 2850 |
| 2450 |
| 1600 |
| 1500 |
| 1300 |
| 1250 |

o/p:

Note:

By default order by is in ascending order.

Ex:

Select ename from emp order by ename;

o/p:

|  |
| --- |
| **ENAME** |
| ADAMS |
| ALLEN |
| BLAKE |
| CLARK |
| FORD |
| JAMES |
| JONES |
| KING |
| MARTIN |
| MILLER |

Select: it chooses which column needs to displayed in the output

From: it chooses the table to fetch the data

Where: As per the condition provided it will execute each and every row and decides which row to pick which row to reject.

STD

|  |  |  |
| --- | --- | --- |
| SID | SNAME | COURSE |
| 1 | RAJA | JAVA |
| 2 | RANI | MAVA |
| 3 | CHAMPA | BAVA |
| 4 | CHAMELI | RAVA |

Select \* from std where sname = ‘RAJA’ ;

o/p:

|  |  |  |
| --- | --- | --- |
| SID | SNAME | COURSE |
| 1 | RAJA | JAVA |

Write a query to display BLAKE record?

Select \* from emp where ENAME = 'BLAKE';

O/p:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **JOB** | **MGR** | **HIREDATE** | **SAL** | **COMM** | **DEPTNO** |
| 7698 | BLAKE | MANAGER | 7839 | 05-01-1981 | 2850 | - | 30 |

Write a query to display employee name , designation and joining date, for all the CLERKS?

Select ename, job, hiredate from emp where job = 'CLERK';

|  |  |  |
| --- | --- | --- |
| **ENAME** | **JOB** | **HIREDATE** |
| SMITH | CLERK | 12/17/1980 |
| ADAMS | CLERK | 01-12-1983 |
| JAMES | CLERK | 12-03-1981 |
| MILLER | CLERK | 01/23/1982 |

O/P:

Write a query to display all the employee details who joined on 23-may-87?

Select \* from emp where hiredate = ‘23-MAY-87’ ;

O/P:

# RELATIONAL OPERATOR

It can be perform only in where statement but not in select statement

* > Greater than
* < smaller than
* >= grater than equal to
* <= smaller than equal to
* = equal to
* != not equal to

Write a query to display all the employee details who is earning salary more than 1500?

Select \* from emp where sal>1500;

O/P:

Write a query to display all employee details apart from SALESMAN?

Select \* from emp where job!= ‘SALESMAN’;

O/P:

# LOGICAL OPERATOR

“AND” when ever we use and operator rows get selected if both the conditions are satisfied.

“OR” when ever we use or operator rows get selected if anyone condition is satisfied.

“NOT” it is an opposite operator

Write a query to display all the details of CLERK who belongs to 20th dept?

Select \* from emp where job=’CLERK’ and deptno = 20;

O/P:

Write a query to display all the details of salesman who is earning salary more than 1500?

Select \* from emp where job=’SALESMAN’ and sal>1500;

O/P:

Write a query to display all employee details who belongs to 10& 30 dept?

Select \* from emp where deptno = 10 or deptno = 30;

O/P:

Write a query to display all employee details of salesman & manager?

Select \* from emp where JOB=’SALESMAN’ or JOB=’MANAGER’;

O/P:

# SPECIAL OPERATOR

IN:

* The record should present in same list or same column
* When we use multiple ‘or’ operator then we are going to replace a special operator called ‘in’.

EX:

Select \* from emp where mgr in(7698,7839,7566);

O/P:

Write a query to display all employee details of CLERK & MANAGER who belongs to 30th dept?

Select \* from emp where job in(‘CLERK’,’MANAGER’) and deptno= 30;

O/p:

Write a query to display all employee details who joined in the year 81?

Select \* from emp where hiredate>=’01-JAN-81’ and hiredate<=’31-DEC-81’;

O/P:

# BETWEEN

Whenever we want to display the values in range then we are going to use a special operator called between.

Ex:

Select \* from emp where hiredate between ’01-JAN-81’ and ’31-DEC-81’;

O/P:

Write a query to display all employee details who is earning salary 2000 to 3000?

Select \* from emp where sal between ‘2000’ and ‘3000’;

O/P:

# LIKE(Pattern matching)

“%” It derives ‘0’ to ‘n’ characters

“\_” it derives a single character

|  |  |  |
| --- | --- | --- |
| SID | SNAME | COURSE |
| 1 | RAJA | BIOLOGY |
| 2 | RANI | ZOOLOGY |
| 3 | CHAMPA | BOTANY |
| 4 | CHAMELI | SOCIAL |

STD

Select \* from std where sname like ‘R%’;

O/P: RAJA

RANI

Select \* from std where course like ‘%y’;

O/P: BIOLOGY

ZOOLOGY

BOTANY

Select \* from std where course like ‘B%Y’;

O/P: BIOLOGY

BOTANY

Select \* from std where sname like ‘%A%’;

O/P: RAJA

RANI

CHAMPA

CHAMELI

Select \* from std where sname like ‘\_\_\_\_’;

O/P: RAJA

RANI

Select \* from std where sname like ‘\_A%’;

O/P: RAJA

RANI

Select \* from std where sname like ‘R\_\_I’;

O/P: RANI

Write a query to display all names who’s names starts with S?

Select \* from emp where ename like ‘S%’;

O/P:

Write a query to display all employee details whose designation ends with ‘MAN’?

Select \* from emp where job like ‘%MAN’;

O/P:

Write a query to display all employee details who joined in year of 81?(by using like operator)

Select \* from emp where hiredate like ‘%81’;

O/P:

Write a query to display all employee details who joined in the month of December?

Select \* from emp where hiredate like ‘%DEC’;

O/P:

Write a query to display all employee details whose salary contains 2nd letter 5 in it?

Select \* from emp where SAL like ‘%5\_’;

O/P:

# NULL

* It is an empty space or blank space. Null means nothing.
* It didn’t occupy space in the memory.
* Null!=0
* To achieve null we have to use special operator called “is”.

Write a query to display all employee details who does not have commission?

Select \* from emp where comm is null;

Write a query to display all employee details who don’t have manager number?

Select \* from emp where mgr is null;

Special Operator

Logical Operator

IN

BETWEEN

LIKE

IS

AND

OR

NOT

Write a query to display all employee details apart from the names start with S?

Select \* from emp where ename not like ‘S%’;

O/P:

Write a query to display all the employee details apart from 10 & 30 dept?

Select \* from emp where deptno not in(10,30);

O/P:

Write a query to display all employee details who is earning salary less than 1000 and more than 2000?

Select \* from emp where sal not between 1000 and 2000;\

O/P:

Write a query to display all employee details who have some commission?

Select \* from emp where comm is not null;

O/P:

# FUNCTION

* SINGLE ROW FUNCTION
* MULTI ROW FUNCTION
* DATE FUNCTION

Single row function: It will take multiple inputs and gives you correspondent output for each input.

Upper()

Lower()

Initcap()

Concat()

Replace()

Length()

Substr()

Write a query to display all the employee name in lower case and designation initial capital letter?

Select lower(ename), initcap(job) from emp;

O/P:

Concat()

Syntax:- select concat(arg1, arg2) from tablename;

Arg1,arg2: column names/ literals.

Concat function is similar to concatenation. Where we are going to join or merge two or more column values or literals. Concat function will take maximum of two arguments.

Select concat (ename, job) from emp;

O/P:

Write a query to display in below format?

My name is SMITH

My name is ALLEN

Select concat (‘my name is’ , ename) from emp;

O/P:

Write a query to display in below format?

Smith is earning 800

Select concat(caoncat(ename, ‘ is earning ‘ ), sal) from emp;

O/P:

Write a query to display below format?

Smith is earning 800 and joined on 7th-DEC-80

Select concat (concat(concat(ename, ‘ is earning ’), sal, ‘ and joined on ’),hiredate) from emp;

O/P:

## Replace

Replace function will replace or will change old text data to new text data.

Syntax:

Select replace(arg1, arg2, arg3) from tablename;

Arg1: column name

Arg2: which char / char to pick

Arg3: replacing char / char

Ex: select replace (‘java’, ‘j’, ‘M’) from dual;

O/P: MAVA

Select replace (‘test’, ‘t’, ‘j’) from dual;

O/P: jest

Select replace (‘Test engineer’, ‘test’, ‘web’) from dual;

O/P: web engineer

Ex: select replace (‘test’, ‘e’, ‘’) from dual;

O/P: t st

Note:

Replace function will take maximum of 3argument minimum of two arguments. If 3rd argument is not given then it will remove the characters which we have given in the 2nd argument.

Ex: select replace (‘test’, ‘e’) from dual;

O/P: tst

Write a query to display all the employee details whose designation contains man? (without using like operator)

Select \* from emp where job!= replace(job, ‘MAN’);

O/P:

Write a query to display all employee details whose salary contain digits ‘5’ without containing like operator?

Select \* from emp where sal!= replace(sal, ‘5’);

Length():

It returns the number of characters present in a given string. String can be column name/ literal.

Syntax: select length (column/literal) from tablename;

Ex: select length(‘uma shankar’) from dual;

Dual

O/P: 11

WAQ to display number of characters present in each employee name?

Syntax – select length (ename) from emp;

Example- select ename, length(ename) from emp;

O/p-

WAQ to display all the employee details who have exactly 5 character without using like operator?

Syntax- select \* where 5 = length (ename) from emp;

Example- select \* from emp where length (ename)=5 ;

O/p-

WAQ a query to display all employee details whose name starts with ‘A’ and ends with ‘Z ’ and have exactly 26 charaters?

Example- select \* from emp

where ename like ‘A%Z’

and length(ename)= 26;

**Substr()**

It returns a part of characters which is present in the given string. Your string can be column name/literal.

Syntax- select substr(agr1,agr2,agr3)

from tablename;

arg1-> column/literal

arg2-> starting position of char

arg3-> No of characters to pick

Example- select substr(‘ UMA SHANKAR ’ 5,5) from dual;

O/p- Substr

SHABKAR

Example- select substr(‘ Start’ 1,1) from dual;

O/p-S

T

Example- select substr(‘ test engineer’ 6) from dual;

O/p- Substr

Engineer

**Note-**

Substr take maximum of arguments and minimum of 2 arguments. If 3rd argument is not given it will go till the end of the string.

Example – select substr(‘test engineer’ -5,2) from dual;

O/p- Su

In

Example – select substr(‘test engineer’ -5,-2) from dual;

O/p- Null

WAQ to display all emp details who joined in years 81, by using substring ? 01-dec-81

Example- Select \* where 81= substr(hiredate,8,2)

From emp;

WAQ to display 1st character and last character of emp name?

Example- Select substr(ename,1,1)

Substr(ename,-1,-1)

from emp;

O/p- S S ENAME

S H SMITH

A N ALLEN

WAQ select \* from emp

Where substr(hiredate,-2)=81;

O/p-

WAQ to display last 3 char of employee name?

Example- select substr(ename,-3,3) ename from emp;

O/p-

WAQ to display all employee details whose designation ends with ger? By using substr

Example- select \* from emp where substr(job,-3)=’GER’;

O/p-

WAQ to display all employee details whose and last letter of employee name is similar to middle letter of month

Example- select \* from emp substr(ename,-2,1)

Substr(hiredate, 5, 1);

O/p-

# Multirow Function

It will take multiple inputs and gives you single output.

Example- Min (), Max (), Sum (), Avg (), Count () etc.

Example- select min(sal) from emp;

O/p-

**Note**

We can have a combination of single row function and a column name in select statement but we can’t have a combination of multirow function and column name in select statement. Otherwise we will going to get an error.

Example- select max(sal), ename from emp;

Select max(sal),ename;

O/p- error

Example – select min(ename) from emp;

O/p-

Example – select min(hiredate) from emp;

O/p-

Example – select count(ename) from emp;

O/p-

**Note-**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Number** | **String** | **Date** |
| **min()** | ✓ | ✓ | ✓ |
| **max()** | ✓ | ✓ | ✓ |
| **sum()** | ✓ | X | X |
| **avg()** | ✓ | X | X |
| **count()** | ✓ | ✓ | ✓ |

WAQ to display total salary for all the managers?

Example- select sum(sal) from emp where job=’MANAGER’;

O/p-

|  |  |
| --- | --- |
| **Format of SQL Query** | **Execution flow of SQL Query** |
| select | select .........5 |
| from | from.........1 |
| where | where .........2 |
| group by | group by .........3 |
| having | having .........4 |
| order by | order by .........6 |

Example- select ename sal as salary from emp

Where salary > 1500;

O/p-

Example- select ename, sal as salary from emp order by salary;

O/p-

WAQ to display highest salary of the employee who joined in the month of DEC?

Ans- select max(sal) from emp

where hiredate like ‘%DEC%’;

WAQTD number of employees who earns commission

Ans- select count(\*)

from emp where comm is not null;

Or,

Select count(comm) from emp;

**DISTINCT()**

Distinct keyword or distinct function will give unique values which is present in the column.

Distinct keyword is used in select statements.

**Note-**

* We can have a combination of distinct column name and column name in select statement.
* We can’t have more than one distinct function in select statement.

Example- select ename, distinct(deptno) from emp;

O/p- Error (Because 1st distinct function will come and column name must be on it .other wise it will show error.

WAQTD number of employees present in each dept?

Example- select count(\*) from emp group by dept no;

O/p-

**GROUP BY**

It divides a table into multiple groups. Group by class will produce each output for each group.

In select statement we can have a combination of multirow function and a column name in select statement only when if we make that column in group by clause.

Example- select count(\*),

deptno from emp

group by dept no;

O/p-

Example- select ename from emp group by dept no;

O/p- error

Example- select \* from emp group by dept no;

O/p-error

WAQTO highest salary of each job?

Example- select max(sal), job from emp group by job;

O/p-

WAQTO Total salary for each job exert ‘CLERK’?

Example- select sum(sal)

from emp

where job!= ‘CLERK’

group by job;

write a query to display of each dept excluding clerk?

Select avg(sal) deptno from emp where job!= ‘clerk’ group by deptno;

O/p:

Write a query to display number of employee present in each dept no and display only those deptno who have minimum four employees in it?

Select count(\*), deptno from emp groupby deptno having count(\*)>=4;

O/P:

Note:

1. Single row functions can be used in select statement and also in where clause.
2. Multirow function can be used only in select statement.
3. In where clause if we use multirow function we are going to get an error.

# HAVING

1. When ever we are providing condition for a group then we are going to use having clause.
2. The purpose of having is to select a group if having condition is satisfied or rejected a group if having condition is not satisfied.
3. In having clause we will use multirow functions.
4. Having clause is work by groupby.

Write a query to display total salary for each job except clerk and that total salary should exceed 5000 and that salary should come in descending order?

Select sum(sal) from emp where job!=’clerk’

Group by job having sum(sal)>5000

Order by sum(sal) desc;

O/P:

# SUB-QURIES

When ever we have an indirect condition generally we go for subquery. When ever we write a subquery at least one common column data should exist between the table.

There are two types of sub-queries.

1. Single row sub queries
2. Multi row sub queries

Single row sub queries:

Single row sub queries will return a single output hence we use relational operator.

Multirow sub queries:

Multi row subqueries will return multiple output hence we use in.

Select \* from faculty where sal = (select sal from faculty where fname=’Gaurav’);

FACULTY

|  |  |  |  |
| --- | --- | --- | --- |
| SID | FNAME | COURSE | SAL |
| 1 | PRAKASH | MANUAL | 100 |
| 2 | ANUPAMA | SQL | 100 |
| 3 | RAGHUL | SQL | 200 |
| 4 | GAURAV | MANUAL | 200 |

Write a query to display all employee details whose deptno is similar to ALLEN?

Select \* from emp where deptno = (select deptno from emp where ename= ‘ALLEN’);

O/P:

Write a query to display all employee details whose designation is similar to the employee whose employee number is 7900?

Select \* from emp where job = (select job from emp where empno = 7900);

O/P:

Write a query to display details whose salary is more than avg salary of 10th dept?

Select \* from emp where sal>(select avg (sal) from emp where deptno = 10);

O/P:

Write a query to display all the employee details whose salary is more than MARTIN salary and designation is similar to BLAKE?

Select \* from emp where sal> ( select sal from emp where ename = ‘MARTIN’) and job=(select job, sal from emp where ename= ‘BLAKE’);

O/P:

Write a query to display all employee details who is working under JAMES?

Select \* from emp where mgr=(select empno from emp where ename=’JONES’);

O/P:

Write a query to display all the employee details who have minimum 3 employees working under them?

Select count(\*) empno>3 from emp (select count(\*) empno from emp groupby mgr);

O/P:

# JOINS

1. Joining or merging one or two table is called joint.
2. Joins are used to fetch the data from multiple table.

Types of joins

1. Cross join / catesian join:

* In this type of joins we will be able to join one or two table
* In this join each and every record of one table is going to match with each and every record of other table.

Ex: If table ‘A’ contains 10(ten) records

If table ‘B’ contains 4(four) records

Then output will be 10\* 4 = 40(forty) records

## Note:

When ever we use cross join we always gets current data as well as incorrect data. That is the reason in real time scenarios we won’t use cross joints.

Select \* from emp, dept;

O/P:

1. Equi joins/ inner joins:

* When ever we use equi joins we always gets matched records.
* For equi joins there should be a common column exist between the tables.
* In equi joins proper condition should be provided and we use equal operator in the condition

Select \* from emp, dept where emp.deptno dept .deptno;

O/P:

Write a query to display all the employee names and their work location?

Select ename, loc from emp, dept where emp.deptno = dept.deptno;

O/P:

Write a query to display employee info and dept info of analyst?

Select \* from emp, dept where emp.deptno = dept.deptno and job = ‘ANALYST’);

O/P:

Write a query to display only dept details who is earning sal more than 1500?

Select \* from emp dept where deptno in (select deptno from emp where sal> 1500);

O/P:

# DDL

Primary key:

1. It is an unique representation or unique identifier column of a table
2. In a table there should be one primary key.
3. Primary key is a combination of unique key and not null key.

What are difference between primary key and unique key?

Primary key:

1. In a table there should be only one primary key column.
2. Primary key will block duplicate & null values

Unique key:

1. In a table we have multiple unique keys.
2. Unique will allow null values and block duplicate values.

Foreign key:

1. Foreign key constrain is also known as referential integrity constrain
2. The keyword which we use for foreign key is references.
3. In a table we can have multiple foreign keys.
4. Foreign key builds a relationship between two tables.
5. Foreign key in a current table is called child table and the referic table is called as parent table or master table.

Candidate key:

The column which is eligible to become primary key is called candidate key.

Alternate key:

After choosing a column as primary key the remaining candidate key is called it as alternate key.

Syntax for create a table:

Create table tablename

(columnname datatype(size) constraint,

columnname datatype(size) constraint,

columnname datatype(size) constraint);

Ex:

Create table uma

(uno number(5) unique,

Uname char(10) notnull);

O/P: table created

Check: select \* from tab;

|  |  |
| --- | --- |
| Tname | tabtype |
| Dept | Table |
| emp | Table |
| Bonus | Table |
| Uma | Table |

# ALTER

After command is used to do modification for an existing table. It can be adding a column for existing table. Remaining old column name to new column name and deleting column from existing table.

Syntax for adding a column:

Alter table tablename;

Add(columnname datatype(size) constraint);

Ex:

Alter table uma

Add (mobno number(10) unique);

O/P: table created

Check: Desc uma;

|  |
| --- |
| Name |
| Uno |
| Uname |
| Mob no |

O/P:

Syntax for rename column name:

Alter table tablename

Rename column oldcolumnname to newcolumnname;

Ex: Alter table uma

Rename column mobno to mno;

O/P: Table altered

Check: desc uma;

O/P: Uma

|  |
| --- |
| Uno |
| Uname |
| mno |

Syntax for drop a column:

Alter table tablename

Drop column columnname;

Ex: Alter table uma drop column mno;

O/P: Table altered

Check: Desc uma;

|  |
| --- |
| Uno |
| uname |

O/P: Name

Syntax for renaming a table:

Rename oldtablename to newtablename;

Ex: rename uma to shankar;

O/P: table renamed

Check: select \* from tab;

O/P: Tname tab type

|  |  |
| --- | --- |
| Dept | Table |
| Emp | Table |
| Bonus | Table |
| Salgrade | Table |
| Shankar | Table |

Syntax for dropping a table:

Drop table tablename;

Ex: drop table shankar;

O/P: table dropped

Difference between drop and truncate:

Drop: drop command will delete the complete table.

Syntax: Drop table tablename;

Truncate: truncate will delete all the records from the table but table structure remains same.

Syntax: truncate table tablename;

**DML**

* By using DML commands we will be to insert values for existing table modify existing value to new value and deleting existing value from a table.
* All ‘DML’ commands are temporary change.

**Syntax for inserting a record-**

Insert into Tablename

(column1, column2, column3)

values(value1, value2, value3);

Example- inset into shankar

(uno,uname)

Values(1,’UMA’)

O/p- row created

Check- select \* from shankar;

O/p-

**Syntax for update record-**

update tablename

set column name = newvalue

where condition;

Example- update emp

set job= ‘SUPERMAN’

O/p- 4 rows updated

where job = ‘SALESMAN

check- select \* from emp;

O/p-

**Syntax for deleting a record-**

delete tablename

where condition;

O/p- 1 row deleted

Example- delete shankar -------->

where uno = 1

O/p-No rows selected

check- select \* from emp; ------>

**TCL (Transaction control language)**

**Commit-**

* Commit will be used to make that particular transaction as permanent change in date base.
* All ‘DML’ commands are temporary change.
* To making it permanent with have to use a ‘TCL’ command called commit.

**Roll back-**

* It acts as an undo function. When ever we perform rollback command it always it always goes to previous commit.

**Save Point-**

|  |  |  |
| --- | --- | --- |
| **-** | **-** | **-** |
| 1 | A | B |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Commit; | | |
|  |
| **-** | **-** | **-** |  |
| 1 | A | B |  |
| 2 | C | D |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Save point A; | | |  |
|  |
| **-** | **-** | **-** |  |
| 1 | A | B |  |
| 2 | C | D |  |
| 3 | E | F |  |
|  |  |  |  |
|  |  |  |  |
| Save point B; | | |  |
|  |
| **-** | **-** | **-** |  |
| 1 | A | B |  |
| 2 | C | D |  |
| 3 | E | F |  |
| 4 | G | H |  |
|  |  |  |  |
| RollBack; | | |  |
| RollBack **A**; | | |  |
| RollBack **B**; | | |  |

Save point acts as a breakpoint for commit command and rollback command.

**Note-**

* Truncate values can’t be rule back where as deleted values can be role back.

**Flashaback**

After dropping a table we can restore if from the recycle bin with the help of flash back command.

Syntax- Flashback Table table name to beforedrop;

**Purge**

If we want to delete table from recycle bin then we have to use a command called purge.

Syntax- Purge table tablename;

**Outer Join**

When ever we perform inner joins we always get some loss of data i.e. unmatched records. To get matched records and unmatched records will go for outer joints.

There are 3 types of outer joins.

1. Left outer join
2. Right outer join
3. Full outer join

**Left outer join**

In the type we will be able to get all the rewards of left side table and matched records of right side table.

Example- select \* from emp, dept

Where emp.dept no= dept.deptno(+);

O/p-

**Right outer join**

In the type we will be able to get all the rewards of right side table and matched records of left side table.

Example- select \* from emp, dept

Where emp.dept no(+)= dept.deptno;

O/p- 15 rows selected

**Full outer join**

In the type we will be able to get all the rewards from both table.

Example- select \* from emp, dept

Where emp.dept no(+)= dept.deptno(+);

O/p- Error (A predicate may reference only one outer table.

**Note-**

Full outer joins can’t be achieved as per oracle standards. Hence

We use “ANSI Rule”.

**ANSI Rule**

**Cross joins-**

Select \* from emp cross join dept;

**Inner joins-**

Select \*

from emp join dept

on emp.deptno = dept.deptno;

**Left outer join-**

Select \*

from emp left outer join dept

on emp.deptno = dept.deptno;

**Right outer join-**

Select \*

from emp right outer join dept

on emp.deptno = dept.deptno;

**Self Join**

* Without aliasing we can’t achieves self joints.

Example- select \*

From emp e emp i

Wher e.mgr=i.empno;

O/p-

WAQTD employee name and their manger details of JAMES?

Syntax- Select e.ename,e.mgr

from emp e,emp i

where e.mgr=i.emp.no;

O/p-

WAQTD employee name and their manager details of JAMES?

Syntax- Select e.ename,e.mgr

from emp e,emp i

where e.mgr=i.emp.no

and e.ename- ‘James’;

O/p-

WAQTD who earn salarys?

Syntax- Select \*

from emp e,emp i

where e.sal=i.sal

**Corelated Subquries**

* It is similar to subqueries
* It works for both joins & subqueries.

**Rownum**

* It is a psedo column.
* Row num values are numeric in nature & unique in nature.
* By default, rownum value is 1/
* Rownum work for <, >, = operator.

**Example-** select \* from emp where rownum=1;

O/p-

**Example-** select \* from emp where rownum=3;

O/p-

**Example-** select \* from emp where rownum<=3;

O/p-

WAQTD 3rd record of emp table?

select \* from (select emp.\*, rownum as a from emp)

where a=3;

O/p-

WAQTD 5th record of emp table?

select \* from (select emp.\*, rownum as a from emp)

where a=5;

O/p-

WAQTD 4th and 6th record of emp table?

select \* from (select emp.\*, rownum as a from emp)

where a in (4,6);

O/p-

**Nth least salary**

Example- Select min(sal)

From (select distinct(sal) from emp order by sal desc) where rownum <=3;

Nth higest 🡪 desc🡪min ()

**Nth least salary**

Example- Select max (sal)

From (select distinct(sal) from emp order by sal desc ) where rownum <=3;

Nth higest 🡪 asc🡪man ()