**SQL >>**

Stands For Structured Query Language With SQL We Can Store, Retrieve, Select, Extract, Read, Update, And Delete The Data In The Database.

**WHAT IS DATA?**

The World Is Divided Into Two Things One Is Non-Living ThingsAnother One Living Things But in Programing Language i.e. C, C++, JAVA, PYTHON, PHP Everything Will Be Consider as Object.

Data Describes the Properties of an Object

(Or)

Data Describes the Attributes of an Entity

EXAMPLE -1

****OBJECT

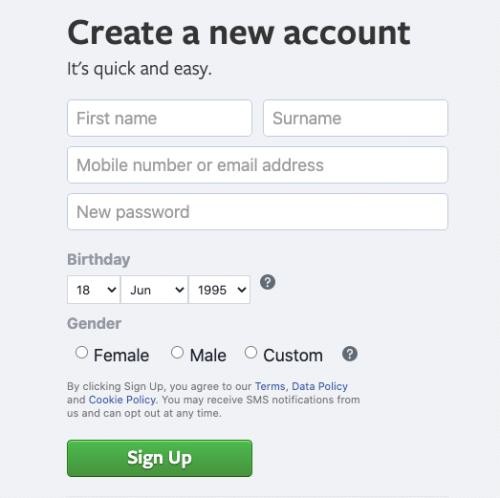
**PROPERTIES DATA**

|  |
| --- |
| **NAME NIBBA** |
| **AGE 24** |
| **GENDER MALE** |
| **DATEOFBIRTH 14-FEB-2000** |
| **HEIGHT 5’1’’ FEET** |
| **WEIGHT 56KGS** |
| **QUALIFICATION B.TECH** |
| **INSTA\_ID call\_me\_nibba**  **GIRL\_FRIEND\_NAME NIBBI** |

**HUMAN**

EXAMPLE:-2

**Properties or Attributes**



ENTITY (OBJECT)

**WHAT IS DATABASE?**

Database Is Place (or) A Medium Which We Store the Data

In A Systematic and Organized Manner



The Basic Operations That Can Be Performed On Database Are

* CREATE / INSERT
* READ / RETRIEVE / SELECT / EXTRACT
* UPDATE / MODIFY
* DELETE / DROP

These Operations Are Referred As **CRUD** Operations.

**DATABASE MANAGEMENT SYSTEM (DBMS)**:

"It is a software which is used to maintain and manage the database”

**“Security** and **authorization”** are the two important features that DBMS provides.



|  |  |  |
| --- | --- | --- |
|  | | Software  Data      Database |
| QUERY  LANGUAGE |  |
|  | |
| FILE FORMAT | |  |

DBMS

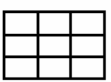
We use query language to communicate or interact with DBMS

DBMS stores the data in the form of **files***.*

**RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS):**

It is a type of DBMS software in which we store the data in the form of

Tables (rows & columns) ".





STRUCTURED QUERY

LANGUAGE

Database

Data

*Software*

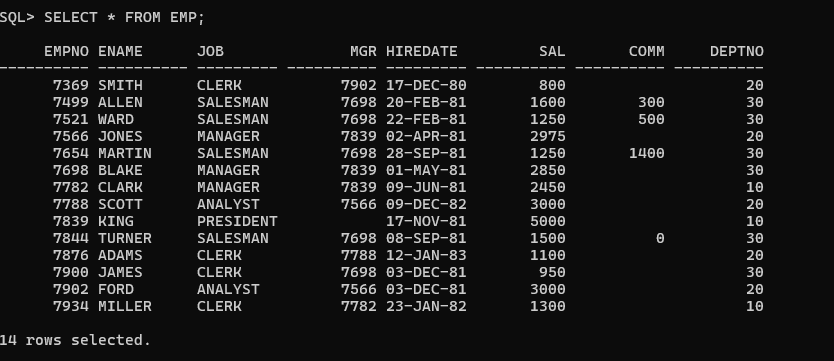
TABLE (ROW&COLUMNS)

We use SQL to communicate or interact with RDBMS

RDBMS stores the data in the form of **Tables.**

EXAMPLE

**EMPLOYEE TABLE WITH DATA**

****

**RELATIONAL MODEL**

Relational Model was designed by “EDGER **FRANK CODD” (“TED”)**

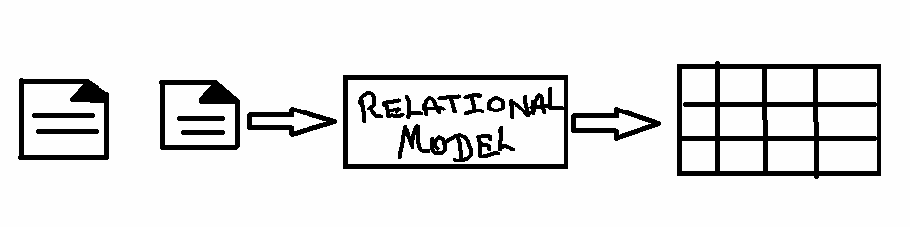
In Relational Model we can store the data in the form of *tables*

Any DBMS which follows Relational Model becomes RDBMS.

DBMS.

Relational Model

RDBMS



**TABLE:**

"It is a logical organization of data which consists of Columns &Rows”

(Or)

“The Combination of Horizontal Lines and Vertical lines Is Knows as Table.”

Rows

/

Records

/

Tuples

Columns / Attribute / Fields

Cell

Employee:

|  |  |  |
| --- | --- | --- |
| **EID** | **ENAME** | **SALARY** |
| 1 | SMITH | 1000 |
| 2 | ALLEN | 1500 |
| 3 | CLARK | 2000 |

RULES OF E.F CODD

Rule No:1

1. The data entered into a cell must always be a “**single valued data.”**

Example:

|  |  |  |
| --- | --- | --- |
| **EID** | **ENAME** | **PHONE\_NO** |
| 1 | SMITH | 101 |
| 2 | ALLEN | 102 , 202 |
| 3 | CLARK | 103 |

|  |  |  |  |
| --- | --- | --- | --- |
| **EID** | **ENAME** | **PHONE\_NO** | **ALTERNATE NO** |
| 1 | SMITH | 101 |  |
| 2 | ALLEN | 102 | 202 |
| 3 | CLARK | 103 |  |

Rule No 2

In RDBMS we store everything in the form of tables including metadata

Example: *Metadata:* The details about a data is knows as Metadata.

### MetaTable

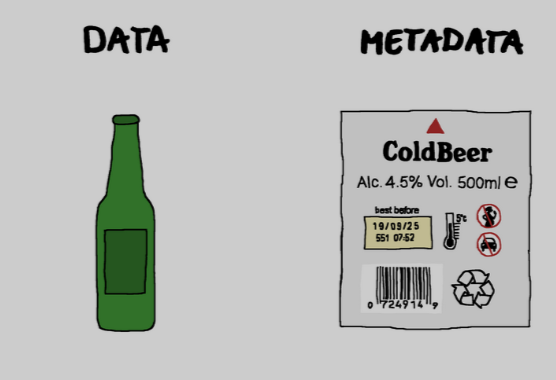
PHOTO

DATA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | | |  |
| **EID** | **ENAME** | **PHOTO** | | |
|  | 1 | SMITH |  |  |  |  |
|  | | |
| 2 | ALLEN |  | | |
| 3 | CLARK |  | | |

Image Name : Mypic size : 127kb resolution : 400 x 600 format : jpeg

|  |  |  |  |
| --- | --- | --- | --- |
| **Image name** | **size** | **Format** | **Resolution** |
| Mypic | 127 | jpeg | 400 x 600 |
|  |  |  |  |

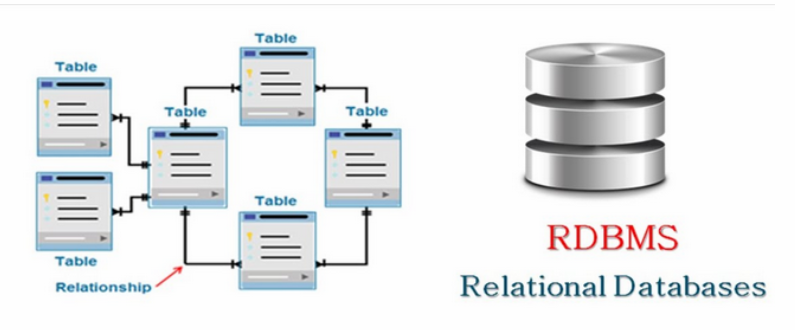


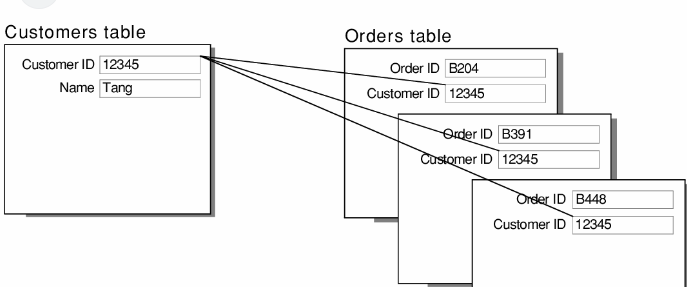
**Metadata is automatically generated (Auto generated)**

**Rule No: 3**

1. According to EDGER FRANK we can store the data in Multiple Tables If needed we can establish a connection between the tables with the Help of “**Key Attribute.”**

**Using (primary key, foreign key)**

****

****

**Rule No 4**

The data entered into the table can be validated in 2 steps.

* 1. By assigning Datatypes To column(s)
  2. By assigning Constraints To column(s)

Note point:-

**Datatypes are mandatory, Constraints are Optional.**

**DATATYPES**

**It is used to specify or determine the type of data**

**That will be stored in a particular memory location.**

**Datatypes in SQL**

1. CHAR(SIZE) DATA TYPE
2. VARCHAR (size)/ VARCHAR2(size) DATA TYPE
3. DATE DATA TYPE
4. NUMBER(PRECISION ,[SCALE])
5. LARGE OBJECTS
   1. Character Large Object. (CLOB)
   2. Binary Large Object (BLOB)

**KEEP IN MIND!**

**NOTE: SQL is not a Case Sensitive Language.**

**CHAR (SIZE) DATA TYPE;-**

1. In character datatype we can store 'A-Z' , 'a-z' , '0-9' And Special Characters( $ , & , @ , ! … ) .
   * Characters must always be enclosed within single quotes ‘ASHOKIT '.(‘ ‘)
   * Whenever we use char datatype we must mention size
   * **Size:** it is used to specify number of characters it can store.
     + The maximum number of characters it can store is

**2000ch**.

Char follows **FIXED LENGTH MEMORY ALLOCATION**

Syntax: CHAR (SIZE)

Example: CHAR (8)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R** | **A** | **M** |  |  |  |  |  |

Used Memory Unused Memory (memory wastage)

In character datatype we can store ALPHANUMARIC

(COMBINATION OF ALPHA BITES AND NUMARIC TYPE OF DATE)

EXAMPLE ‘ABC123’

**VARCHAR (SIZE) DATA TYPE:-**

1. In varchar datatype we can store 'A-Z' , 'a-z' , '0-9' And Special Characters( $ , & , @ , ! … ) .

Characters must always be enclosed within single quotes ' RAMAKRISHNA SIR'.

Whenever we use char datatype we must mention size

**Size**: it is used to specify number of characters it can store.

The maximum number of characters it can store **2000 CHARACTERS**

**Varchar (size) follows “variable length memory allocation** “.

Syntax: VARCHAR (SIZE)

Example: VARCHAR (8)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R** | **A** | **M** |  |  |  |  |  |

Used Memory Unused Memory (free memory)

* In character datatype we can store ALPHANUMARIC

(COMBINATION OF ALPHA BITES AND NUMARIC TYPE OF DATE)

EXAMPLE: Insta\_Id ‘link2ram’

**VARCHAR2 (SIZE) DATA TYPE:-**

In varchar2 datatype we can store 'A-Z', 'a-z' , '0-9' And Special

Characters( $ , & , @ , ! … ) .

Characters must always be enclosed within single quotes ' RAMAKRISHNA SIR'.

Whenever we use char datatype we must mention size

**Size**: it is used to specify number of characters it can store.

The maximum number of characters it can store **4000 CHARACTERS**

**Varchar2 (size) follows “variable length memory allocation** “.

Example: VARCHAR (8)

Syntax: VARCHAR2 (SIZE)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R** | **A** | **M** |  |  |  |  |  |

Used Memory Unused Memory (free memory)

* In character datatype we can store ALPHANUMARIC

(COMBINATION OF ALPHA BITES AND NUMARIC TYPE OF DATE)

EXAMPLE: Insta\_Id ‘xyz123’

**Example:**

By using char (size) Data Type Varchar2 (size**)**

## STUDENT

|  |  |  |  |
| --- | --- | --- | --- |
| **USN** | **SNAME** | **ADDRESS** | **PAN\_NO** |
| CHAR(4) | VARCHAR(10) | VARCHAR2(10) | CHAR(10) |
| RAM | NIBBA | AMEERPET | ABC123XYZ1 |
| RAVI | NIBBI | KPHB | ABC123XYZ2 |

**ASSIGNMENT:**

DIFFERENTIATE BETWEEN CHAR & VARCHA

**ASCII: [American Standard Code for Information Interchange**]

**SELECT ASCII (‘A’)**

**FROM DUAL;**

|  |  |
| --- | --- |
| 'A' | 65 |
| 'Z' | 90 |
| 'a' | 97 |
| 'z' | 122 |

**NUMBER (PRECISION,[SCALE])**

SYNTAX: **NUMBER** (Precision, **[Scale**])

**NUMBER:** It is used to store numeric values.

[ ] - Not Mandatory.

**Precision**: it is used to determine the number of digits used to store integer value

**Scale**: it is used to determine the number of digits used to store decimal (floating) value within the precision

By default scale value is zero (**0**)

Scale is not mandatory

EXAMPLE WE HAVE TAKEN DIGIT 7

|  |  |  |
| --- | --- | --- |
| Example : | Number ( 3 ) | +/- 777 |
| Example : | Number ( 5 , 0 ) | +/- 77777 |
| Example : | Number ( 5 , 2 ) | +/- 777.77 |
| Example : | Number ( 7 , 3 ) | +/- 7777.777 |
| Example : | Number ( 4 , 4 ) | +/- .7777 |
| Example : | Number ( 5 , 4 ) | +/- 7.7777 |
| Example : | Number ( 3 , 6 ) | +/- .000777 |
| Example : | Number ( 5 , 8 ) | +/- .00077777 |
| Example : | Number ( 2 , 7 ) | +/- .0000077 |

|  |  |  |
| --- | --- | --- |
| **EID** | **PHONE\_NO** | **SALARY** |
| Number( 3 ) | Number ( 10 ) | Number ( 7 , 2 ) |
| 101 | 9985396677 | 25000.65 |

**DATE:** it is used to store dates in a particular format.

It used Oracle specified Format.

|  |  |  |
| --- | --- | --- |
| 'DD-MON-YY' | OR | 'DD-MON-YYYY' |
| '12-AUG-24' |  | '12-AUG-2024' |

SYNTAX: **DATE**

**EMPLOYEE\_DETAILS**

|  |  |  |
| --- | --- | --- |
| **EMPLOYE\_DOB** | **EMPLOYEE\_HIREDATE** | **EMPLOYEE\_ANNIVERSARY** |
| DATE | DATE | DATE |
| ’22-JUN-2000’ | ’22-MAY-2018’ | ’24-MAY-2024’ |

**LARGE OBJECTS**

**CHARACTER LARGE OBJECT (CLOB)**

Datatype we can store 'A-Z' , 'a-z' , '0-9' And Special Characters( $ , & , @ , ! … ) .

It is used to store characters up to 4 GB of size.

In character datatype we can store ALPHANUMARIC.

(COMBINATION OF ALPHA BITES AND NUMARIC TYPE OF DATE)

CHARACTER LARGE OBJECT VALUES WILL BE STORED IN CHARACTER

FORMAT

|  |
| --- |
| SYNTAX : **CLOB** |

EXAMPLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| @ | # | $ | ^ | 4 | 5 |
| E | 4 | 7 | 8 | % | \* |
| 5 | ^ | ( | \_\* | # | $ |

**Binary large object (BLOB):**

Datatype we can store 'A-Z, 'a-z', '0-9' And Special Characters ($, &, @, #…).

It is used to store characters up to 4 GB of size.

In Binary Large Object Datatype we can store the Values/data in Binary Format

|  |
| --- |
| SYNTAX: **BLOB** |

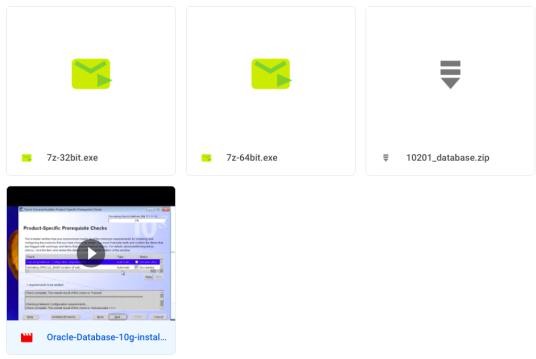
EXAMPLE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1` | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 |

FOR WINDOWS

To download **bit.ly/roSoftWIN**

(Or)

 TO download **tinyurl.com/jspsqlsw**

**CONSTRAINTS**

It Is Rule and Restriction /Terms and Condition Are Given To the Column(s).

**Types of Constraints:**

1. UNIQUE
2. NOT NULL
3. CHECK
4. PRIMARY KEY
5. FOREIGN KEY.
6. **UNIQUE:** "It is used to avoid duplicate values into the column ".

Duplicate means repeated more than one time.

1. **NOT NULL:** "It is used to avoid Null ".

Null means blank space/ empty

Not Null With Out Blank Spac

1. **CHECK** : "It is an extra validation with a condition

If the condition is satisfied then the value is accepted else Rejected ".

|  |
| --- |
| **Check (Length(column\_name) =size**) |

CHECK IS USED TO CHECK THE SIZE OF VALUES

|  |  |  |
| --- | --- | --- |
| 0 TO 9 | SIZE IS | 1 |
| 10 TO 99 | SIZE IS | 2 |
| 100 TO 999 | SIZE IS | 3 |
| 1000 TO 9999 | SIZE IS | 4 |

1. **PRIMARY KEY:** "It is a constraint which is used to identify a record uniquely from the table”.

**Characteristics of Primary key**:

* + We can have only 1 PK in a table
  + PK cannot accept duplicate / repeated values.
  + PK cannot accept Null

PK is always a combination of Unique and Not Null Constraint

1. . **FOREIGN KEY** : "It is used to establish a connection between the Tables "

Characteristics of Foreign key:

* + We can have only Multiple FK in a table
  + FK can accept duplicate / repeated values.
  + FK can accept Null
  + FK is not a combination of Unique and Not Null Constraint.
  + For an Attribute ( column ) to become a FK ,it is mandatory

That it must be a PK in its own table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EID** | **NAME** | **SALARY** | **DNO** | FK | **CID** FK |
| 1 | A | 10000 | 20 | | 2 |
| 2 | B | 20000 | 10 | | 3 |
| 3 | C | 35000 | 20 | | 1 |
| 4 | D | 50000 | 10 | | 2 |

|  |  |  |
| --- | --- | --- |
| **CID** | **CNAME** | **CNO** |
| 1 | X | 1001 |
| 2 | Y | 2002 |
| 3 | Z | 3003 |

|  |  |  |
| --- | --- | --- |
| **DNO** | **DNAME** | **LOC** |
| 10 | D1 | L1 |
| 20 | D2 | L2 |

Parent Table

**Example for Foreign Key :**

**Emp**

Child Table

**Customer**

**Dept**

Parent Table

1. Differentiate between Primary key and Foreign key .

|  |  |
| --- | --- |
| **PRIMARY KEY** | **FOREIGN KEY** |
| It is used to identify a records Uniquely from the table. | It is used to establish a connection Between the tables |
| It cannot accept Null | It can accept Null |
| It cannot accept duplicate values | It can accept duplicate values |
| It is always a combination of Not Null and Unique constraint | It is not a combination of  Not Null and Unique constraint |
| We can have only 1 PK in a table | We can have Multiple FK in a table |

NOTE NULL

Null is a keywordwhich is used to represent Nothing / Empty Cell.

**Characteristics of Null:**

* + Null doesn’t represent 0 or Blank Space.
  + Any operations performed on a Null will result in Null itself.
  + Null doesn’t occupy any Memory.
  + We cannot Equate Null.

STATEMENTNS ARE CLASSIFIED INTO 5 DIFFERENT TYPES

* DATA DEFINITION LANGUAGE ( DDL )
* DATA MANIPULATION LANGUAGE ( DML )
* TRANSACTION CONTROL LANGUAGE ( TCL )
* DATA CONTROL LANGUAGE ( DCL )
* DATA QUERY LANGUAGE ( DQL )

DATA DEFINITION LANGUAGE (DDL)

“DDL is used to construct an object in the database and deals with the Structure of the Object/ENTITY”

Statements in DDL (Dr.CAT)

* + 1. CREATE
    2. RENAME
    3. ALTER
    4. TRUNCATE
    5. DROP

CREATE

“It Is Used To Build / Construct an Object”

Create Is Used To Create an Entity/Object (Or) View (Virtual Table)

**How to Create a Table**

* Name of the Table(s) cannot have same names.
* Number of Columns.
* Names of the column
* Assign Constraints

[Not mandatory]

Syntax to create a table:

CREATE TABLE Table\_Name(

Column\_Name1 datatype constraint\_type, Column\_Name2 datatype constraint\_type, Column\_Name3 datatype constraint\_type,

.

.

Column\_NameN datatype constraint\_type

);

Example

Create table customer

(

Customer\_id int primary key,

Customer\_name varchar (10),

Customer\_number number (10) not null check (length (Customer\_number) =10),

Address varchar (14)

);

NOTE

**TO SEE THE TABLE\_NAME SQL PLUS APPLICATION**

SQL>SELECT \*

2 FROM TAB;

**TO** **DESCRIBE THE COLUMN\_NAMES WITH DATA TYPE IN SQL PLUS APPLN**

**SYNTAX:-DESCRIBE TABLE\_NAME**;

EXAMPLE:-DESCRIBE CUSTOMER;

CUSTOMER PRODUCTS

CUSTOMER\_ID CUSTOMER\_ID

(PK) (FK)

**SYNTAX TO CREATE TABLE TO ESTABLIISH CONNECTIONS BETWEEN THE MULTIPLE TABLE**

CREATE TABLE Table\_Name

(

Column\_Name1 datatype constraint\_type,

Column\_Name2 datatype constraint\_type,

Column\_Nmae3 datatype constraint\_type,

.

Column\_NameN datatype,

Constraint Foreign key references Parent\_Table\_Name (Column\_Name)

);

CREATE TABLE PRODUCT

(

Product\_ID Number (2) primary key,

Product\_NAME Varchar (10),

PRICE Number (7, 2) check (Price > 0),

Customer\_id int,

Constraint CID\_FK Foreign Key (customer\_id) References Customer (customer\_id)

);

**RENAME:** "IT IS USED TO CHANGE THE NAME OF THE OBJECT”

**Syntax: RENAME Table\_Name TO New\_Table \_Name;**

**RENAME Customer TO Cust;**

**SQL>DESCRIBE CUST**

**ALTER:**” IS USED TO MODIFY THE STRUCTURE OF THE TABLE AFTER CREATATION”

**TO ADD AN COLUMN\_AFTER CREATATION OF THE TABLE**

**Syntax: ALTER TABLE Table\_Name**

**ADD Column\_Name Datatype Constraint\_type;**

**Example: ALTER TABLE Cust**

**ADD MAIL\_ID Varchar (15);**

**SQL> DESCRIBE CUST;**

**TO DROP A COLUMN AFTER CREATATION OF THE TABLE:**

**Syntax**: **ALTER TABLE Table\_Name DROP COLUMN Column\_Name;**

**Example:** **ALTER TABLE Cust**

**ADD MAIL\_ID Varchar (15);**

**SQL> DESCRIBE CUST**

**TO RENAME AN COLUMN\_AFTER CREATATION OF THE TABLE**

**Syntax: ALTER TABLE Table\_Name**

**RENAME COLUMN Column\_Name TO new\_Column\_Name;**

**Example: ALTER TABLE Cust**

**RENAME COLUMN Customer\_number TO PHONE\_NO;**

**SQL> DESCRIBE CUST;**

**TO MODIFY THE DATATYPE** **AN COLUMN\_AFTER CREATATION OF THE TABLE**

**Syntax: ALTER TABLE Table\_Name**

**MODIFY COLUMN\_NAME New\_Datatype;**

**Example: ALTER TABLE Cust**

**MODIFY CNAME CHAR (10)**

**SQL>DESCRIBE CUST;**

**TO MODIFY NOT NULL CONSTRAINTS AFTER CREATATION OF THE TABLE**:

**Syntax: ALTER TABLE Table\_Name**

**MODIFY COLUMN\_NAME Existing\_datatype [NULL]/NOT NULL;**

**Example:** **ALTER TABLE Cust**

**MODIFY ADDRESS Varchar (15) Not Null;**

**SQL>DESCRIBE CUST;**

**TO ADD FOREIGN KEY ESTABLISH CONNECTION BETWEEN THE MULTIPLE\_TABLE**

**AFTER CREATATION OF THE TABLES**

**Syntax: ALTER TABLE Child\_Table\_Name**

**Add Froeign\_key (column\_name) references parent\_table\_name (column\_name)**

**Example:**

**Alter table product**

**Add foreign key (customer\_id) references customer (customer\_id)**

**TRUNCATE:** "**IT IS USED TO REMOVE ALL THE RECORDS FROM THE TABLE PREMANENTLY WITHOUT DISTURBING THE STRUCTURE OF THE TABLE”**

**(Or)**

**“We want to delete the data inside table permanently than we use Truncate”**

**Syntax:** **TRUNCATE TABLE Table\_Name;**

**KEEP IT IN MIND!**

**PLEASE DON’T PERFORM THE TRUCATE OPERATION**

**ON ANY DEVELOPMENT TABLES**

**STUDENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **SID** | **SNAME** | **SAGE** | **GENDER** |
| 1 | A | 21 | MALE |
| 2 | B | 23 | MALE |
| 3 | C | 21 | FEMALE |

**Example: TRUNCATE TABLE STUDENT;**

|  |  |  |  |
| --- | --- | --- | --- |
| **SID** | **SNAME** | **SAGE** | **GENDER** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**NOTE POINT:-Delete the data inside table permanently**

**DROP:** "**IT IS USED TO REMOVE THE TABLE FROM THE DATABASE”**

**(Or)**

**“IT REMOVES BOTH DATA AND THE STRUCTURE OF THE TABLE PERMANENTLY**

**FROM THE DATABASE.”**

**Syntax: DROP TABLE Table\_Name;**

**ORDERS BIN TABLE**

Bin$1234abc3456

|  |  |  |  |
| --- | --- | --- | --- |
| **OID** | **Oname** | **ORPHNO** | **ORAddress** |
| 1 | A | 1234567890 | HYD |
| 2 | B | 1234567899 | PUNE |
| 3 | C | 1234567880 | DELHI |

**ADDRESS**

**IN SQL PLUS 10g Application**

**SQL>SELECT \***

**2 FROM TAB;**

**T\_NAME T\_TYPE**

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

**DEPT TABLE**

**EMP TABLE**

**SALGRADE TABLE**

**BONUS TABLE**

**CUSTOMER TABLE**

**PRODUCT TABLE**

**STUDENT TABLE**

**BIN$1234abc3456 TABLE**

**SQL>SELECT \***

**2 FROM RECYCLEBIN;**

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

**BIN$1234abc3456 TABLE DROP ORDERS**

**THE TABLE IS PRESENT IN RECYCLE BIN WE HAVE OPTIONS**

**ONE IS TO RECOVER THE TABLE BY (FLASHBACK STATEMENT)**

**ANOTHER ONE DELETE FORVER FROM DATABASE BY (PURGE)**

**TO RECOVER THE TABLE FROM RECYCLEBIN**

**Syntax: FLASHBACK TABLE Table\_Name TO BEFORE DROP;**

FLASHBACK

Address: BIN$123abc3456

**Orders**

**FLASHBACK TABLE Orders**

**TO BEFORE DROP**

**TO DELETE THE TABLE FROM BIN (GONE FOREVER) PURGE**

**Syntax: PURGE TABLE Table\_Name;**

ORDERS

**DROP**

Gone Forever

BIN$123abc3456

**FLASHBACK TABLE**

**TABLE\_NAME TO BEFORE DROP;**

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

**NOTE: DDL STATEMENTS ARE AUTO-COMMIT STATEMENTS**

**DATA MANIPULATION LANGUAGE (DML)**

**“It is used to manipulate the Object by performing insertion, updating and deletion”.**

1. INSERT
2. UPDATE (UID)
3. DELETE

**INSERT: “It is used to insert / create records in the table”**

**Syntax: INSERT INTO Table\_Name VALUES (v1, v2 , v3 …… ) ;**

**INSERTING THE VALUES IN CUSTOMER**

DESCRIBE CUSTOMER;

|  |  |  |  |
| --- | --- | --- | --- |
| CID | CNAME | CNO | ADDDRESS |
| NUMBER(2) | VARCHAR(10) | NUMBER(10) | VARCHAR(20) |
|  |  |  |  |

**INSERT INTO CUSTOMER VALUES (1, ‘ASHOK’, 9985396677,’AMEERPET’)**

ONE ROW CREATED

|  |  |  |  |
| --- | --- | --- | --- |
| CID | CNAME | CNO | ADDDRESS |
| NUMBER(2) | VARCHAR(10) | NUMBER(10) | VARCHAR(20) |
| 1 | ASHOK | 9985396677 | AMEERPET |

**INSERT INTO CUSTOMER VALUES (2, ‘RAM’, 9398314309,’KPHB’);**

ONE ROW CREATED

SELECT \*

FROM CUSTOMER;

|  |  |  |  |
| --- | --- | --- | --- |
| CID | CNAME | CNO | ADDDRESS |
| ;NUMBER(2) | VARCHAR(10) | NUMBER(10) | VARCHAR(20) |
| 1 | ASHOK | 9985396677 | AMEERPET |
| 2 | RAM | 9398314309 | KPHB |

**Another way of inserting data into the table is shown below;**

**SYNTAX:-**

**INSERT INTO TABLE\_NAME VALUES (&COLUMN\_NAME1, &COLUMN\_NAME2,,,,);**

**ENTER VALUE FOR COLUMN\_NAME1;**

**ENTER VALUE FOR COLUMN\_NAME2;**

**ENTER VALUE FOR COLUMN\_NAME3;**

**CREATE TABLE BOOKS**

**(**

**BOOK\_ID INT PRIMARY KEY,**

**BOOK\_NAME VARCHAR2 (16) NOT NULL**

**);**

**DESCRIBE BOOKS;**

**COLUMN\_NAMES DATA\_TYPES**

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

**BOOK\_ID NUMBER (38)**

**BOOK\_NAME VARCHAR2 (16)**

**INSERT INTO BOOKS VALUES (&BOOK\_ID, &BOOK\_NAME);**

**ENTER VALUE FOR BOOK\_ID:11**

**ENTER VALUE FOR BOOK\_NAME:’JAVA’**

**1 ROW CREATED**

**SQL> SELECT \***

**2 FROM BOOKS;**

|  |  |
| --- | --- |
| **BOOK\_ID** | **BOOK\_NAME** |
| **11** | **JAVA** |

**UPDATE: It is used to modify an existing value**

**Syntax: UPDATE Table\_Name**

**SET Column\_Name = Value, Column\_Name = Value,,,,,,,,,,**

**[WHERE COLUMN\_NAME OPERATOR ‘DATA/VALUE**’];

**SQL>UPDATE BOOKS**

**2 SET BOOK\_NAME =’ORACLE SQL’**

**3 WHERE BOOK\_ID =11;**

**1 ROW UPDATED**

|  |  |
| --- | --- |
| **BOOK\_ID** | **BOOK\_NAME** |
| **11** | **ORACLE SQL** |

**DELETE: It is used to remove a particular record from the table.**

**Syntax: DELETE FROM Table Name**

**[WHERE COLUMN\_NAME OPERATOR ‘DATA/VALUE’];**

**SQL>DELETE FROM BOOKS**

**WHERE BOOK\_ID =11;**

|  |  |
| --- | --- |
| **BOOK\_ID** | **BOOK\_NAME** |
|  |  |

**ASSIGNMENT ON DML STATEMENTS**

1. WAQT UPDATE THE SALARY OF EMPLOYEE TO DOUBLE THEIR SALARY IF

HE IS WORKING AS A MANAGER.

1. WAQT CHANGE THE NAME OF SMITH TO SMIITH.
2. WAQT MODIFY THE JOB OF KING TO 'PRESIDENT'.
3. WAQT TO CHANGE NAME OF ALLEN TO ALLEN MORGAN.
4. WAQT HIKE THE SALARY OF THE EMPLOYEE TO 10%. IF EMPLOYEES EARN LESS THAN 2000 AS A SALESMAN.
5. WAQ TO DELETE THE EMPLOYEES WHO DON’T EARN COMMISSION.
6. WAQ TO REMOVE ALL THE EMPLOYEES HIRED BEFORE 1987 IN DEPT 20
7. DIFFERENTIATE BETWEEN TRUNCATE AND DELETE STATEMENTS.

|  |  |
| --- | --- |
| **TRUNCATE** | **DELETE** |
| Belongs to DDL | Belongs to DML |
| Removes all the records from the  Table permanently. | Removes a particular record from the  Table. |
| Auto COMMIT | Not auto COMMIT. |

**TRANSACTION CONTROL LANGUAGE (TCL)**

Any DML change on a table is not a permanent one. We need to save the DML changes in order to make it permanent we can also undo (ignore) the same DML changes on a table. The DDL changes cannot be undone as they are implicitly saved.

(Or)

“It is used to control the transactions done on the database ".The DML Operations performed on the Database are known as Transactions such as Insertion, Updating and Deletion.

We have 3 Statements:

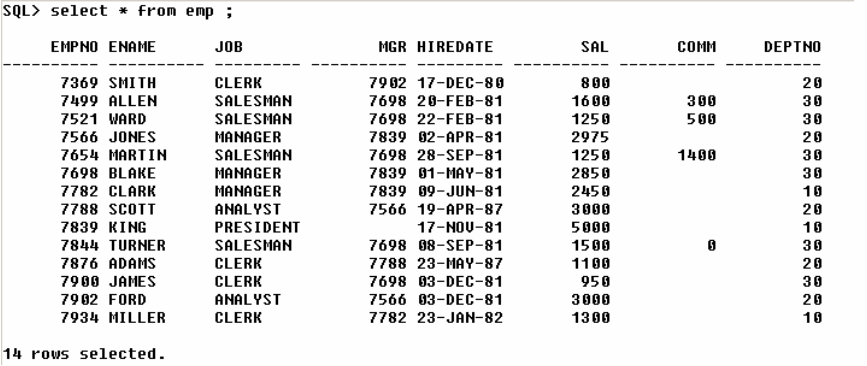
COMMIT

ROLLBACK

SAVEPOINT

**ROLLBACK**

It undoes (ONE STEP BACK WORD) the DML changes performed on a table.

**** Let us see in the below example how rollback works,

**SQL>DELETE FROM EMP;**

**14 row deleted.**

**SQL>SELECT \***

**2 FROM EMP;**

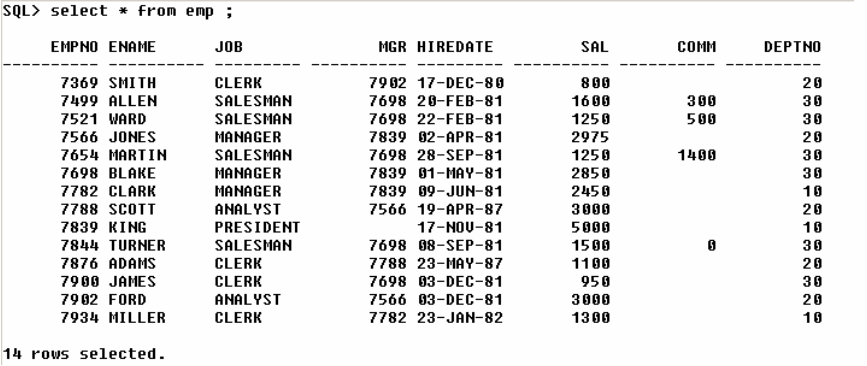
**No row selected**

**SQL>ROLLBACK;**

**Rollback completed**

**SQL>SELECT \***

**2 FROM EMP;**

****

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

**HOW TO CREATE DUPLICATE TABLE WITH SAME DATA (Or) SAME VALUES**

**SYNTAX:**

**SQL>CREATE TABLE New\_Table\_Name AS SELECT \* FROM orig\_Table\_Name;**

**EXAMPLE**

**SQL>CREATE TABLE TEST AS SELECT \* FROM DEPT;**

TABLE CREATED

**SELECT \* FROM TAB**;

T\_NAME T\_TYPE

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

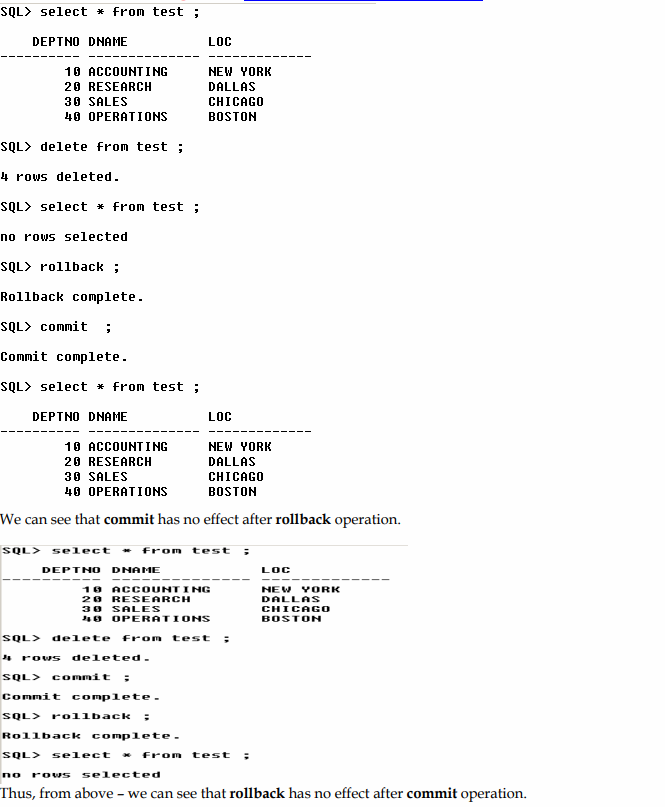
TEST TABLE

**KEEP IT IN MIND FIRST CREATE DUPLICATE TABLE**

**COMMIT**

“It saves the DML changes permanently to the database“

Committing after rollback & vice versa will not have any effect

**** Let us explain the above statement with an example,

ASSIGNMENT**: SQL 10g Application**

**CREATE TABLE STUEDENT**

**STUDENT\_ID INT (PRIMARY KEY),**

**STUDENT\_NAME VARCHAR2 (15),**

**STUDENT\_AGE NUMBER (3),**

**STUDENT\_DATE\_OF\_BIRTH DATE,**

**STUDENT\_BRANCH VARCHER2 (12)**

**CEATE TABLE COLLEGE**

**COLLEGE\_ID INT (PRIMARY KEY),**

**COLLEGE\_NAME VARCHAR2 (13),**

**COLLAGE\_ADDRESS VARCHAR2 (16),**

**COLLEGE\_INAGURATION\_DATE DATE,**

**STUDENT\_ID INT (FOREIGN KEY)**

**AFTER CREATATION OF BOTH TABLES ESTABLISH CONNECTION**

**BETWEEN MULTIPLE TABLES**

**INSERT THE 10 RECORD IN EACH (STUDENT, COLLEGE) TABLE**

**PERFORM COMMIT OPERATION**

**PERFORM THE DELETE OPERATON**

**PERFORM THE ROLLBACK OPERATION**

**PERFORM THE UPDATE OPERATION**

**PERFORM THE DROP OPERATION**

**PERFORM THE FLASHBACK OPERATION**

**SAVE POINT:**

This statement is used to mark the positions or restoration points.

(Nothing related to DB) .

SYNTAX: **SAVEPOINT** Savepoint\_Name;

"This statement is used to control the flow of data between the users ".

We have 2 statements:

1. GRANT
2. REVOKE

**GRANT:** THIS STATEMENT IS USED TO GIVE PERMISSION TO A USER.

**SYNTAX: GRANT SQL\_STATEMENT**

**ON TABLE\_NAME TO USER\_NAME;**

**REVOKE:** THIS STATEMENT IS USED TO TAKE BACK THE PERMISSION FROM THE USER.

**SYNTAX: REVOKE SQL\_STATEMENT**

**ON TABLE\_NAME FROM USER\_NAME;**

**KEET IT MIND TRY THIS**

SQL> SHOW USER;

SQL> SHOW USER;

USER is "SCOTT"

SQL> CONNECT

Enter user-name: HR Enter password: \*\*\*\*\* Connected.

USER is "HR"

SQL> SELECT \*

2 FROM SCOTT.EMP; FROM SCOTT.EMP

\*

ERROR at line 2:

ORA-00942: table or view does not exist

SQL> CONNECT

Enter user-name: SCOTT Enter password: \*\*\*\*\* Connected.

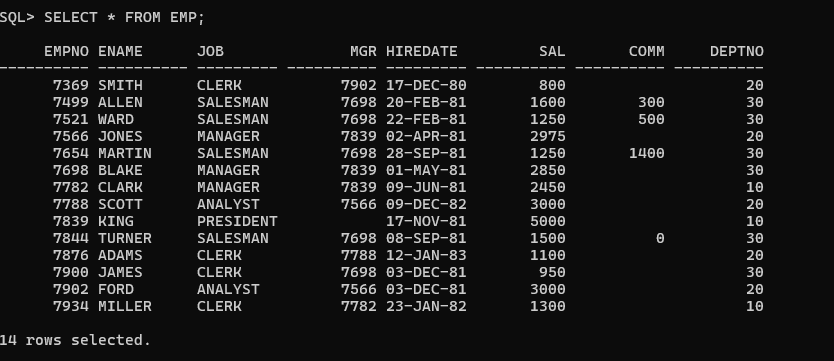
SQL> GRANT SELECT ON EMP TO HR;

Grant succeeded. SQL> CONNECT

Enter user-name: HR

Enter password: \*\*\*\*\*

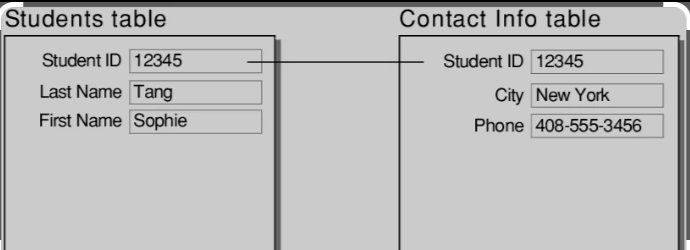
Connected.

****

TYPES OF TABLE RELATATIONS

* ONE TO ONE
* ONE TO MANY
* MANY TO ONE
* MANY TO MANY

**ONE TO ONE**



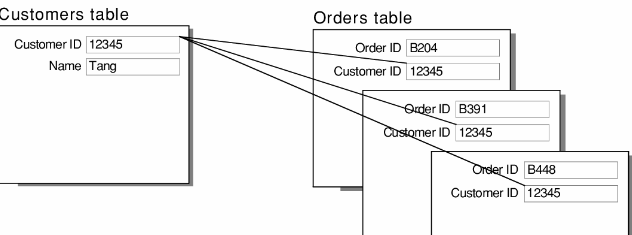
(PARENT\_TABLE) (CHILD\_TABLE)

CREATE THE TABLE ESTABLISH CONNECTION BETWEEN

THE TABLE

INSERT THE VALUES IN BOTH TABLES

**ONE TO MANY**

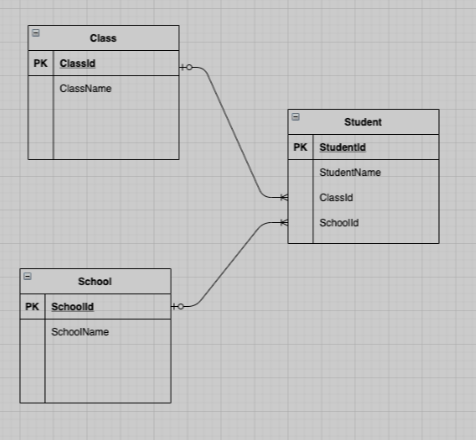


PARENT\_TABLE CHILD\_TABLES

CREATE THE TABLES ESTABLISH CONNECTION BETWEEN

THE TABLE

INSERT THE VALUES IN MULTIPLE TABLES

**MANY TO ONE**

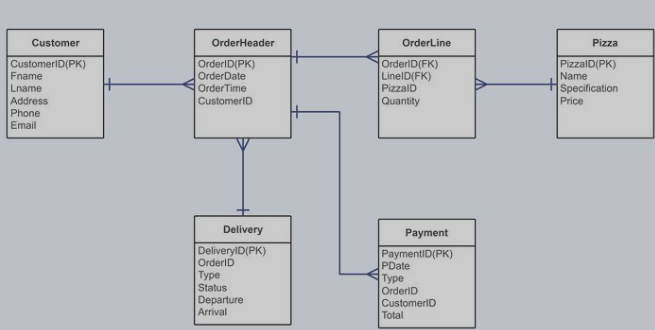
**PARENT\_TABLE CHILD\_TABLE**

CREATE THE TABLES ESTABLISH CONNECTION BETWEEN

THE TABLE

INSERT THE VALUES IN MULTIPLE TABLES

ASSIGNMENT:-



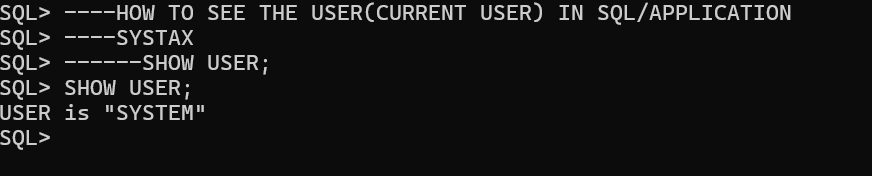
CREATE THE TABLES ESTABLISH CONNECTION BETWEEN

THE TABLE INSERT THE VALUES IN MULTIPLE TABLES

**SQL APPLICATION 19C download link**

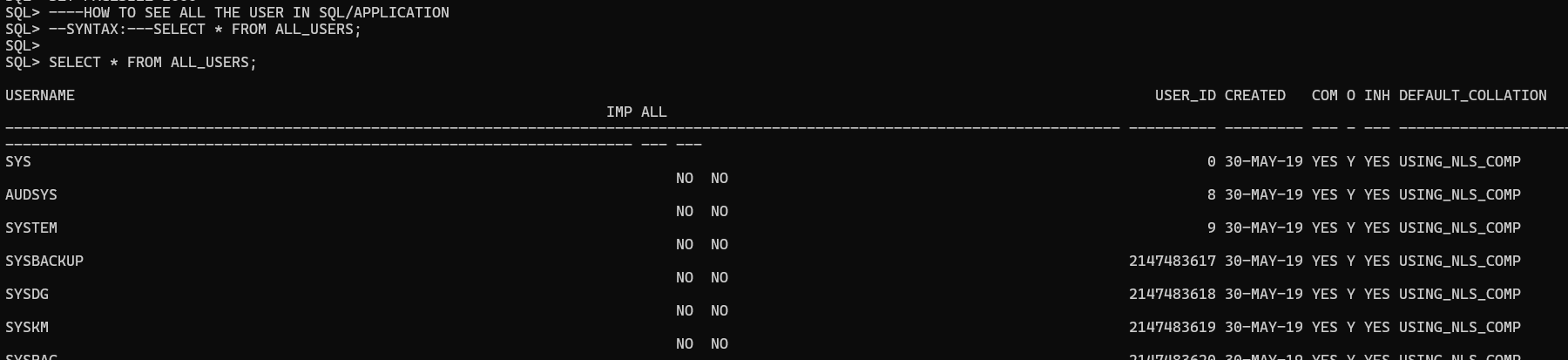
<https://drive.google.com/file/d/1Rbw9_k-XzaJ2roPmio3ifvfq104kqU3C/view>

**SCOTT DATABASE DEPT AND EMP TABLE**



HOW TO SEE ALL THE USER IN SQL/APPLICATION

**SQL>SELECT \* FROM ALL\_USERS;**



----HOW CREATE A NEW USER

----HOW TO CREATE TABLE IN NEW USER

----HOW INSERT DATE IN NEW USER TABLE

----HOW TO UPDATE DATA IN NEW USER TABLE

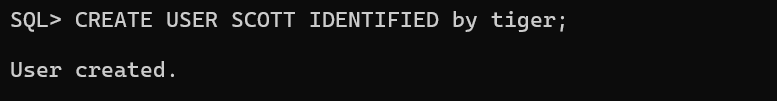
----HOW TO DELETE THE DATA IN NEW USER TABLE

STEP -1

**Create user username identified by password;**

CREATE USER SCOTT IDENTIFIED by tiger;

User created

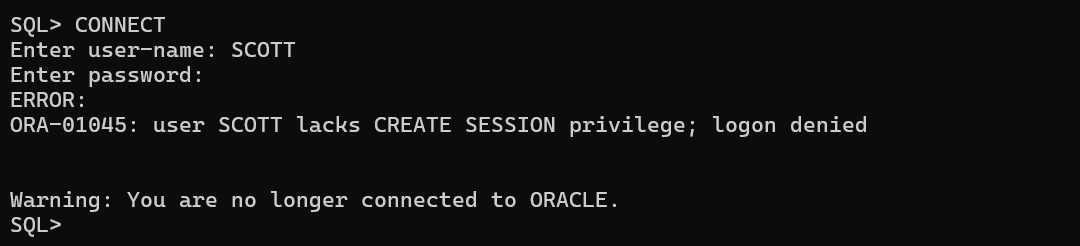


**READ**

NOW YOU HAVE CREATED THE USER

USER NAME IS SCOTT.

WE CAN NOT CONNECT WITH NEW USER SCOTT



WE CAN NOT CREATE A TABLE IN NEW USER SCOTT

WE CAN NOT INSERT THE VALUES/DATE IN NEW USER SCOTT

WE CAN NOT UPDATE/MODIFY THE DATA/VALUES NEW IN USER SCOTT

WE CAN NOT DELETE/DROP THE DATA IN NEW USER SCOTT

**CONCLUSION-** WITH OUT CONNECT WITH NEW USER WE CAN NOT CREATE/INSERT/UPDATE/DELETE THE DATE IN NEW USER SCOTT

------TO OVER COME THIS USER (SCOTT) NEED TO TAKE PERMISSION (PRIVILEGES)

----FROM DATABASE ADMINSTRATION (DATABASE ADMIN)

----WE HAVE DATABASE ADMIN i.e(system/tiger)

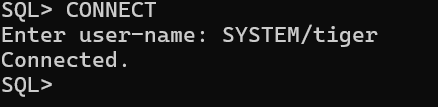
STEP-2

SQL> connect

Enter user-name: system

Enter password: tiger

Connected



WE HAVE DATABASE ADMIN i.e(system/tiger)

TO CREAT/INSERT/UPDATE/DELETE DATA OF USER.

DATABASE ADMIN i.e (system/tiger)

SHOULD GRANT THE PRIVILEGES (PERMISSIONS) TO USER SCOTT

STEP-3

**Grant all privileges to user\_Name;**

**PRIVILEGES MEANS PERMISSION**

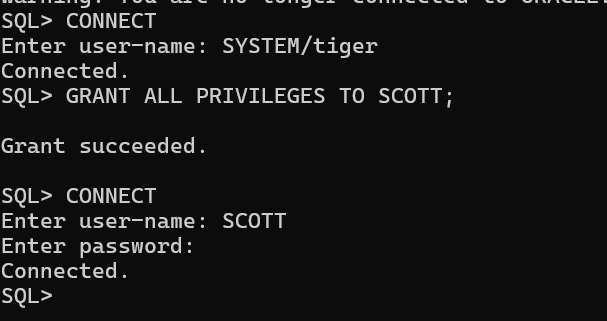
SQL>GRANT ALL PRIVILEGES TO SCOTT; -----copy/past in SQL Appn.

GRANT SUCCEEDED

(Or)

SQL> GRANT DBA TO SCOTT; -----copy/past in SQL Appn

GRANT SUCCEEDED



STEP-4

SQL>CONNECT

Enter user-name: SCOTT

Enter password: tiger

Create table DEPT

(

DEPTNO number (2) PRIMARY KEY,

DNAME varchar2 (14),

LOC varchar2 (13)

);

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');

INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');

INSERT INTO DEPT VALUES (40,'OPERATION','BOSTON');

SELECT \* FROM DEPT;

COMMIT;

CREATE TABLE EMP

(

EMPNO NUMBER (4) PRIMARY KEY,

ENAME VARCHAR (10),

JOB VARCHAR (9),

MGR NUMBER (4),

HIREDATE DATE,

SAL NUMBER (7, 2),

COMM NUMBER (72),

DEPTNO NUMBER (2),

CONSTRAINT F\_EMP foreign key (DEPTNO) references DEPT (DEPTNO)

);

INSERT INTO EMP VALUES

(7369, 'SMITH', 'CLERK', 7902, '17-DEC-1980', 800, NULL, 20);

INSERT INTO EMP VALUES

(7499, 'ALLEN', 'SALESMAN', 7698, '20-FEB-1981', 1600, 300, 30);

INSERT INTO EMP VALUES

(7521, 'WARD', 'SALESMAN', 7698, '22-FEB-1981', 1250, 500, 30);

INSERT INTO EMP VALUES

(7566, 'JONES', 'MANAGER', 7839, '2-APR-1981', 2975, NULL, 20);

INSERT INTO EMP VALUES

(7654, 'MARTIN', 'SALESMAN', 7698, '28-SEP-1981', 1250, 1400, 30);

INSERT INTO EMP VALUES

(7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-1981', 2850, NULL, 30);

INSERT INTO EMP VALUES

(7782, 'CLARK', 'MANAGER', 7839, '9-JUN-1981', 2450, NULL, 10);

INSERT INTO EMP VALUES

(7788, 'SCOTT', 'ANALYST', 7566, '09-DEC-1982', 3000, NULL, 20);

INSERT INTO EMP VALUES

(7839, 'KING', 'PRESIDENT', NULL, '17-NOV-1981', 5000, NULL, 10);

INSERT INTO EMP VALUES

(7844, 'TURNER', 'SALESMAN', 7698, '8-SEP-1981', 1500, 0, 30);

INSERT INTO EMP VALUES

(7876, 'ADAMS', 'CLERK', 7788, '12-JAN-1983', 1100, NULL, 20);

INSERT INTO EMP VALUES

(7900, 'JAMES', 'CLERK', 7698, '3-DEC-1981', 950, NULL, 30);

INSERT INTO EMP VALUES

(7902, 'FORD', 'ANALYST', 7566, '3-DEC-1981', 3000, NULL, 20);

INSERT INTO EMP VALUES

(7934, 'MILLER', 'CLERK', 7782, '23-JAN-1982', 1300, NULL, 10);

SQL>COMMIT;

CREATE TABLE SALGRADE

(

GRADE NUMERIC,

LOSAL NUMERIC,

HISAL NUMERIC

);

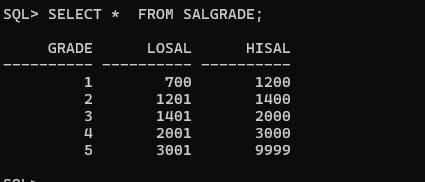
INSERT INTO SALGRADE VALUES (1, 700, 1200);

INSERT INTO SALGRADE VALUES (2, 1201 1400);

INSERT INTO SALGRADE VALUES (3, 1401, 2000);

INSERT INTO SALGRADE VALUES (4, 2001, 3000);

INSERT INTO SALGRADE VALUES (5, 3001, 9999);



CREATE TABLE BONUS

(

ENAME VARCHAR (10),

JOB VARCHAR (9),

SAL NUMERIC,

COMM NUMERIC

);

COMMIT;

**DATA QUERY LANGUAGE (DQL)**

**“DQL is used to retrieve/Select the data from the database”**

It had 4 statements

* 1. SELECT
  2. PROJECTION
  3. SELECTION
  4. JOIN

**SELECT:** "It is used to retrieve the *data* from the table and display it”

**Projection**

It is the process of retrieving the data by selecting only columns is known as projection

In projection all the records / values present in a particular column are by default selected.

**SELECTION:** "It is a process of retrieving the data by selecting both the columns and rows is known as Selection”

**JOIN:** "It is a process of retrieving the data from **multiple tables**

Simultaneously (AT A TIME) is known as Join”.

**PROJECTION**

"It is a process of retrieving the data by selecting only the columnsis known as Projection”.

In projection all the records / values present in a particular column are by default

SYNTAX :

**SELECT \* / [DISTINCT] Column\_Name / Expression [ALIAS]**

**FROM Table\_Name;**

**STUDENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SID | SNAME | SGENDER | SAGE | SBRANCH | PERCENTAGE |
| 1 | A | MALE | 21 | CSE | 76 |
| 2 | B | FEMALE | 22 | MECH | 86 |
| 3 | C | FEMALE | 23 | EEE | 87 |
| 4 | D | MALE | 23 | ECE | 76 |
| 5 | E | MALE | 24 | CIVIL | 84 |
| 6 | F | MALE | 25 | CSE | 76 |

SQL>-----WAQTD SNAME FOR ALL THE STUDENTS

SQL>SELECT SNAME

2 FROM STUDENTS;

SQL>---WAQTD SNAME SGENDER FOR ALL THE STUDENTS

SQL> SELECT SNAME, SGENDER

2 FROM STUDENTS;

SQL>-----WAQTD SNAME AGE SGENDER FOR ALL THE STUDENTS

SQL> SELECT SNAME, SAGE, SGENDER

2 FROM STUDENTS;

SQL>---WAQTD student id and student names for all the students

SQL>--- WAQTD name and branch of all the students

SQL>--- WAQTD NAME, BRANCH AND PERCENTAGE FOR ALL THE STUDENTS.

SQL>---- WAQTD details of all the students from students table

**STUDENT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SID | SNAME | SGENDER | SAGE | SBRANCH | PERC |
| 1 | A | MALE | 21 | CSE | 76 |
| 2 | B | FEMALE | 22 | MECH | 86 |
| 3 | C | FEMALE | 23 | EEE | 87 |
| 4 | D | MALE | 23 | ECE | 76 |
| 5 | E | MALE | 24 | CIVIL | 84 |
| 6 | F | MALE | 25 | CSE | 76 |

SQL>-----WAQTD SNAME FOR ALL THE STUDENTS

SQL>SELECT SNAME

2 FROM STUDENTS;

**RESULTE OF FROM CLAUSE**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SID | SNAME | | SGENDER | | SAGE | | SBRANCH | | PERC | |
| 1 | A | MALE | | 21 | | CSE | | 76 | |
| 2 | B | FEMALE | | 22 | | MECH | | 86 | |
| 3 | C | FEMALE | | 23 | | EEE | | 87 | |
| 4 | D | MALE | | 23 | | ECE | | 76 | |
| 5 | E | MALE | | 24 | | CIVIL | | 84 | |
| 6 | F | MALE | | 25 | | CSE | | 76 | |

**RESULT OF SELECT CLAUSE**

STUDENTT



|  |
| --- |
| SNAME |
| A |
| B |
| C |
| D |
| E |
| F |

ORDER OF EXECUTIO

FROM Clause

SELECT Clause

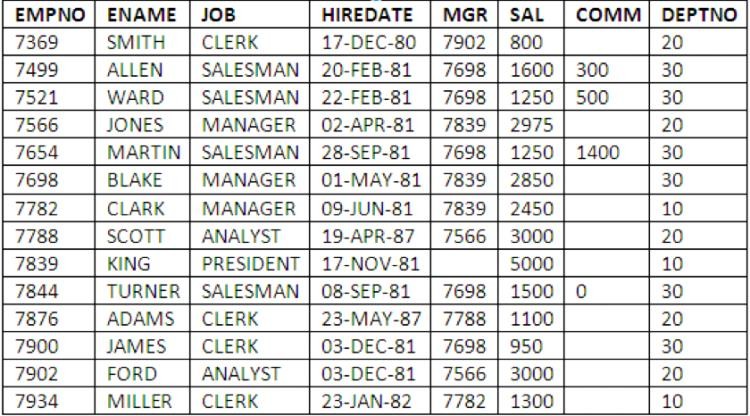
* + FROM Clause starts the execution.
  + For FROM Clause we can pass Table\_Name as an argument.(column\_name)
  + The job of FROM Clause is to go to the Database and search for the table and put the table under execution.
  + SELECT Clause will execute after the execution of FROM Clause
  + For SELECT Clause we pass 3 arguments

⬥ \*

⬥ Column\_Name

⬥ Expression

* + The job of SELECT Clause is to go the table under execution and select the columns mentioned.
  + SELECT Clause is responsible for preparing the result table.
  + Asterisk (\*): it means to select all the columns from the table .
  + Semicolon: it means end of the query.

EMP

**WAQTD name salary and commission given to all the employees.**

**WAQTD name of the employee along with their date of joining**

**DEPT**

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

SQL>---**WAQTD dname and location for all the depts.**

QUESTIONS ON EMP AND DEPT TABLE:

* WRITE A QUERY TO DISPLAY ALL THE DETAILS FROM THE EMPLOYEE TABLE.
* WAQTD NAMES OF ALL THE EMPLOYEES.
* WAQTD NAME AND SALARY GIVEN TO ALL THE EMPLOYEES.
* WAQTD NAME AND COMMISSION GIVEN TO ALL THE EMPLOYEES.
* WAQTD EMPLOYEE ID AND DEPARTMENT NUMBER OF ALL THE EMPLOYEES IN EMP TABLE.
* WAQTD ENAME AND HIREDATE OF ALL THE EMPLOYEES.
* WAQTD NAME AND DESIGNATION OF ALL THE EMPLPOYEES.
* WAQTD NAME, JOB AND SALARY GIVEN ALL THE EMPLOYEES.
* WAQTD DNAMES PRESENT IN DEPARTMENT TABLE.
* WAQTD DNAME AND LOCATION PRESENT IN DEPT TABLE.

**DISTINCT CLAUSE**

“It is used to remove the duplicate or repeated values from the Result table”

MOVIES

|  |  |  |  |
| --- | --- | --- | --- |
| **MOVIEID** | **MOVIENAME** | **MOVIEYEAR** | **MOVIESDAYS** |
| 1 | OG | 2025 | 60 |
| 2 | DEVARA | 2025 | 60 |
| 3 | KUSHI | 2023 | 75 |
| 4 | THOLI PREMA | 2018 | 80 |
| 5 | KUSHI | 2004 | 75 |
| 6 | THOLI PREMA | 1998 | 80 |

Distinct clause has to be used as the first argument to select clause

We can use multiple column(s) as an Argument to distinct clause.

It will remove the combination of columns in which the record are duplicated.

SQL>----WAQTD MOVIENAMES FOR ALL MOVIES

SQL>SELECT MOVIENAME

2 FROM MOVIES;

OUTPUT

|  |
| --- |
| **MOVIENAME** |
| OG |
| DEVARA |
| KUSHI |
| THOLI PREMA |
| KUSHI |
| THOLI PREMA |

SQL >----WAQTD REMOVE THE DUPLICATE MOVIENAMES FOR ALL MOVIES

SQL>SELECT DISTINCT MOVIENAM

2 FROM MOVIES;

|  |
| --- |
| **MOVIENAME** |
| OG |
| DEVARA |
| KUSHI |
| THOLI PREMA |
| KUSHI |
| THOLI PREMA |

|  |
| --- |
| **MOVIENAME** |
| OG |
| DEVARA |
| KUSHI |
| THOLI PREMA |

SQL>--WAQTD REMOVIE THE DUPLICATE DEPTNO FOR ALL THE

EMPLOYEEYS

SQL>--WAQTD AVOID REPATED JOB FOR ALL THE EMPLOYEES.

SQL>---WAQTD REMOVE THE REPATED SALARY FOR ALL THE EMPLOYEES

SQL>--WAQTD REMOVE THE REPATED SALARY, JOB, DEPTNO FOR ALL THE EMPLOYEES

SQL>--WAQTD REMOVE THE REPATED ENAME, SALARY, JOB, DEPTNO FOR ALL THE EMPLOYEES

**EXPRESSION**

A Statement which Gives result is known as Expression.

Expression is combination Operand and Operator.

**OPERAND**: These Are the Value That We Pass

6

Result

2 \* 3

Statement

**OPERATOR**: These are the symbols Which Perform

= some Operation on the Operand.

EXPRESSION

EMP

|  |  |  |
| --- | --- | --- |
| EMPNO | ENAME | SAL |
| 1 | SMITH | 800 |
| 2 | KING | 5000 |
| 3 | SCOTT | 3000 |

SQL>---WAQTD NAME AND SALARY GIVEN TO THE EMPLOYEE

SQL>SELECT ENAME, SAL

2 FROM EMP;

SQL>---WAQTD NAME AND ANNUAL SALARY OF THE EMPLOYEES.

SQL>SELECT ENAME, SAL, SAL\*12

2 FROM EMP;

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **SAL\*12** |
| 1 | SMITH | 800 | 9600 |
| 2 | KING | 5000 | 60000 |
| 3 | SCOTT | 3000 | 36000 |

SQL>---WAQTD ALL THE DETAILS OF THE EMPLOYEES ALONG WITH ANNUAL

---SALARY.

SQL>SELECT EMP.\*, SAL\*12

2 FROM EMP;

(Or)

SQL>SELECT EMPNO,ENAME,SAL,SAL\*12

2 FROM EMP;

SQL>----WAQTD NAME AND SALARY WITH A HIKE OF 20%

SQL>SELECT ENAME, SAL+SAL\*20/100

2 FROM EMP;

**FORMULA TO CALCULATE PERCENTAGE**

|  |  |
| --- | --- |
| **SAL+SAL\*a/100** | **SAL\*1.a** |

SQL>---WAQTD NAMES AND SALARY OF AN EMPLOYEE WITH A DEDUCTION

OF 10%

SQL>SELECT ENAME, SAL-SAL\*10/100

2 FROM EMP;

**ALIAS**

“It is an alternate name given to a Column or an Expression In the result table ".

We can assign alias name with or without using 'As' keyword.

Alias names have to be a single string which is separated by an underscore or enclosed within double quotes.

|  |  |
| --- | --- |
| Example : | ANNUAL\_SALARY |
|  | "ANNUAL SALARY" |

* WAQTD annual salary for all the employees.

SQL>SELECT SAL\*12

2 FROM EMP;

**SELECTION**

It is a process of retrieving the data by **selecting both the columns and rows**is known as Selection ".

**SYNTAX:**

**SELECT \* / [DISTINCT] Column\_Name / Expression [ALIAS]**

**FROM Table\_Name**

**WHERE COLUMN\_NAME OPERATOR ‘DATA/VALUE’ <*Filter\_Condition*>;**

**WHERE CLAUSE**

"Where clause is used to filter the records ".

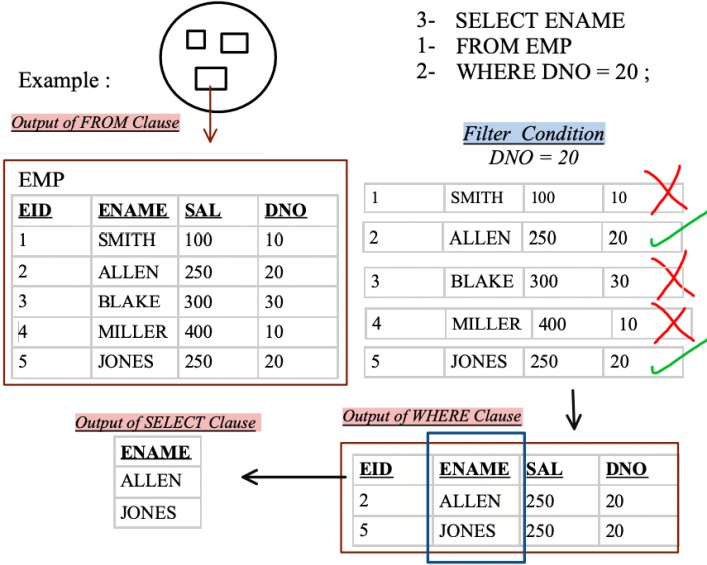
ORDER OF EXECUTION

FROM

WHERE

SELECT

SQL>--WAQTD ENAMES OF THE EMPLOYEES WORKING IN DEPTNO 20



FROM CLAUSE STARTS THE EXECUTION

FOR FROM CLAUSE WE CAN PASS THE TABLE\_NAME AS ARGUMENT.

THE JOB OF THE FROM CLAUSE IS GO TO THE DATADABE AND SEARCH FOR THE TABLE AND PUT THE TABLE UNDER EXECUTION

AFTER THE EXECUTION OF FROM CLAUSE WHERE CLAUSE

WILL EXECUTES

WHERE CLUAUSE JOB IS FILTER THE RECORDS CONDITION(S)

WHERE CLAUSE IS TO GO THE TABLE UNDER THE EXECUTION THE COLUMN MENTIONED. WHERE CLUAUSE WILL FILTER THE RECORDS CONDITION(S) ROW BY ROW

AFTER THE EXECUTION OF WHERE CLAUSE SELECT CLAUS WILL

EXECUTES

FOR SELECT CLAUSE WE PASS 3 ARGUMENTS

⬥ \*

⬥ COLUMN\_NAME

⬥ EXPRESSION

THE JOB OF SELECT CLAUSE IS TO GO THE TABLE UNDER THE EXECUTION AND SELECT THE COLUMN MENTIONED

SELECT CLAUSE IS RESPONSIBLE FOR PREPARING THE RESULT TABLE.

ASTERISK (\*): IT MEANS TO SELECT ALL THE COLUMNS FROM THE TABLE.

SEMICOLON: IT MEANS END OF THE QUERY.

SQL>---WAQTD names of the employees getting salary More than 300

SQL>SELECT ENAME,SAL

2 FROM EMP

3 WHERE SAL >300;

SQL>-- WAQTD names and salary of the employees working in dept. 10.

SQL>SELECT ENAME,SAL ,DEPTNO

2 FROM EMP

3 WHERE DEPTNO = 10;

SQL>---WAQTD All the details of the employees whose salary is less than 1000

SQL>SELECT \*

2 FROM EMP

3 WHERE SAL <1000;

==============================================================================

FROM EMP;



SQL>WAQTD NAME AND HIREDATE OF AN EMPLOYEE HIRED ON ’09-JUN-1981’

SQL>SELECT ENAME, HIREDATE

2 FROME EMP

3 WHERE HIREDATE =’09-JUN-81’;

SQL> WAQTD details of the employee whose name is 'Miller'

SQL>SELECT \*

2FROM EMP

3 WHERE ENAME = ‘MILLER’;

SQL> WAQTD details of the employee hired after '01-JAN-1982'

SQL>SELECT \*

2 FROME EMP

3 WHERE HIREDATE >’01-JAN-1982’;

SQL> WAQTD name sal and hiredate of the employees who were Hired before 1985.

SQL>SELECT ENAME, SAL, HIREDATE

2 FROM EMP

3 WHERE HIREDATE <’01-JAN-85’;

SQL>WAQTD name sal and hiredate of the employees who were Hired after 1985

SQL>SELECT ENAME, SAL, HIRDATE

2 FROM EMP

1. WHERE HIREDATE >’31-DEC-85’;

SQL>WAQTD NAMES SAL HIREDATE OF THE EMPLOYEES WORKING AS MANAGER

SQL >SELECT ENAME, SAL, HIREDATE

2 FROM EMP

3 WHERE JOB=’MANAGER’;

* WAQTD name of the employees who was hired on Valentine's On 2024
* WAQTD THE ANNUAL SALARY OF THE EMPLOYEE WHOS NAME IS SMITH
* WAQTD NAME OF THE EMPLOYEES WORKING AS CLERK
* WAQTD SALARY OF THE EMPLOYEES WHO ARE WORKING AS SALESMAN
* WAQTD DETAILS OF THE EMP WHO EARNS MORE THAN 2000
* WAQTD DETAILS OF THE EMP WHOS NAME IS JONES
* WAQTD DETAILS OF THE EMP WHO WAS HIRED AFTER 01-JAN-81
* WAQTD NAME AND SAL ALONG WITH HIS ANNUAL SALARY IF THE ANNUAL SALARY IS MORE THAN 12000
* WAQTD EMPNO OF THE EMPLOYEES WHO ARE WORKING IN DEPT 30
* WAQTD ENAME AND HIREDATE IF THEY ARE HIRED BEFORE 1981
* WAQTD DETAILS OF THE EMPLOYEES WORKING AS MANAGER
* WAQTD NAME AND SALARY GIVEN TO AN EMPLOYEE IF EMPLOYEE EARNS A COMMISSION OF RUPEES 1400
* WAQTD DETAILS OF EMPLOYEES HAVING COMMISSION MORE THAN SALARY
* WAQTD EMPNO OF EMPLOYEES HIRED BEFORE THE YEAR 87
* WAQTD DETAILS OF EMPLOYEES WORKING AS AN N ANALYST
* WAQTD DETAILS OF EMPS EARNING MORE THAN 2000 RUPEES PER MONTH
* WAQTD NAME AND DEPTNO OF THE EMPLOYEES HIRED AFTER

'01-JAN-87

* WAQTD NAME AND HIREDATE OF THE EMPLOYEES HIRED BEFORE

31-JUL-88

**OPERATORS:-**

**THE SYMBOLES WHICH ARE USED TO PERFORM THE OPERATION IS KNOW AS OPERATORS**

**(+ plus symbol Addition operation)**

**(- minus symbol Substation operation)**

**(\*into symbol multiplication operation)**

**OPERATORS IN SQL**

1. ARITHEMATIC OPERATORS **:- ( + , - , \* , / )**
2. CONCATENATION OPERATOR :- ( **||** )
3. COMPARISION OPERATORS :- ( = , != or <> )
4. RELATIONAL OPERATOR :- ( > , < , >= , <= )
5. LOGICAL OP : ( AND , OR , NOT )
6. SPECIAL OPERATOR :-
   1. **IN**
   2. **NOT IN**
   3. **BETWEEN**
   4. **NOT BETWEEN**
   5. **IS**
   6. **IS NOT**
   7. **LIKE**
   8. **NOT LIKE**
7. SUBQUERY OPERATORS:-
   1. **ALL**
   2. **ANY**
   3. **EXISTS**
   4. **NOT EXISTS**

**CONCATENATION OPERATOR** IS USED JOIN THE String.

**Symbol: ||**

SQL>---WAQTD WISH GOOD AFTER NOON TO ALL THE EMPLOYEESS

SQL>SELECT ‘GOOD AFTER NOON ‘||ENAME

2 FROM EMP;

**LOGICAL OPERATOR (AND OR NOT):**

**AND OPERATOR** Is Binary Operator. Will Accept Minimum Two Input/Conditions Process It Give One Result

|  |  |  |
| --- | --- | --- |
| **CONDITION1** | **CONDITION2** | **RESULT** |
| **T** | **T** | **T** |
| **F** | **T** | **F** |
| **T** | **F** | **F** |
| **F** | **F** | **F** |

AND OPERATOR Should Satisfy All The Conditions Then Only Result Will Be True.

SQL>--WAQTD name and deptno along with job for the employee working in dept 10..

SQL>SELECT ENAME, DEPTNO, JOB

2 FROM EMP

3 WHERE DEPTNO =10;

SQL>-- WAQTD name and deptno along with job for the employee working as manager in dept 10.

SQL>SELECT ENAME, DEPTNO, JOB

2 FROM EMP

3 WHERE JOB =’MANAGER’ AND DEPTNO =10;

SQL>WAQTD name, deptno, salary of the employee working in dept 20 and earning less than 3000.

SQL>SELECT ENAME, DEPTNO, SAL

2 FROM EMP

3 WHERE DEPTNO =20 AND SAL <3000;

**OR OPERATOR** Is a Binary Operator. Will Accept Two Condition/Inputs process it Give one Result.

|  |  |  |
| --- | --- | --- |
| **CONDITION1** | **CONDITION2** | **RESULT** |
| T | T | T |
| F | T | T |
| T | F | T |
| F | F | F |

SQL>WAQTD name and deptno of the employees if the works in dept 10 or 20

SQL>SELECT ENAME, DEPTNO

2 FROM EMP

3 WHERE DEPTNO =10 OR DEPTNO =20;

SQL>WAQTD name and deptno of the employee if the employee working as

CLERK OR MANAGER

SQL>SELECT ENAME, DEPTNO, JOB

2 FROM EMP

1. WHERE JOB =’MANAGER’ OR JOB =’CLERK’;

SQL>WAQTD Names of the employees working AS SALESMAN OR DEPTNO 30

SQL>SELECT ENAME, JOB, DEPTNO

2 FROM EMP

3 WHERE JOB = ‘SALESMAN’ OR DEPTNO =30;

**ASSIGNMENT ON LOGICAL OPREATORS**

* SQL>WAQTD NAME, DEPTNO, SALARY OF THE EMPLOYEE WORKING IN DEPT 20 AND EARNING LESS THAN 3000
* SQL>WAQTD NAME AND SALARY OF THE EMPLOYEE IF EMP EARNS MORE THAN 1250 BUT LESS THAN 3000
* SQL>WAQTD NAME AND SAL AND DEPTNO OF THE EMPLOYEES IF EMP GETS MORE THAN 1250 BUT LESS THAN 4000 AND WORKS IN DEPT 20.
* SQL>WAQTD NAME, JOB, DEPTNO OF THE EMPLOYEES WORKING AS A MANAGER IN DEPT 10 OR 30.
* SQL>WAQTD NAME, DEPTNO, JOB OF THE EMPLOYEES WORKING IN DEPT 10 OR 20 OR 30 AS A CLERK.
* SQL>WAQTD NAME, JOB AND DEPTNO OF THE EMPLOYEES WORKING AS CLERK OR MANAGER IN DEPT 10.
* SQL>WAQTD NAME , JOB , DEPTNO , SAL OF THE EMPLOYEES WORKING AS CLERK OR SALESMAN IN DEPT 10 OR 30 AND EARNING MORE THAN 1800
* SQL>WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK AND EARNING LESS THAN 1500
* SQL>WAQTD NAME AND HIREDATE OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 30
* SQL>WAQTD DETAILS OF THE EMP ALONG WITH ANNUAL SALARY IF THEY ARE WORKING IN DEPT 30 AS SALESMAN AND THEIR ANNUAL SALARY HAS TO BE GREATER THAN 14000.
* SQL>WAQTD ALL THE DETAILS OF THE EMP WORKING IN DEPT 30 OR AS ANALYST
* SQL>WAQTD NAMES OF THE EMPMLOYEES WHOS SALARY IS LESS THAN 1100 AND THEIR DESIGNATION IS CLERKSQL>WAQTD NAME AND SAL, ANNUAL SAL AND DEPTNO IF DEPTNO IS 20 EARNING MORE THAN 1100 AND ANNUAL SALARY EXCEEDS 12000
* SQL>WAQTD EMPNO AND NAMES OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 20
* SQL>WAQTD DETAILS OF EMPLOYEES WORKING IN DEPT 20 OR 30
* SQL>WAQTD DETAILS OF EMPLOYEES WORKING AS ANALYST IN DEPT 10.
* SQL>WAQTD DETAILS OF EMPLOYEE WORKING AS PRESIDENT WITH SALARY OF RUPEES 4000
* SQL>WAQTD NAMES AND DEPTNO, JOB OF EMPS WORKING AS CLERK IN DEPT 10 OR 20
* SQL>WAQTD DETAILS OF EMPLOYEES WORKING AS CLERK OR MANAGER IN DEPT 10
* SQL>WAQTD NAMES OF EMPLOYEES WORKING IN DEPT 10 OR, OR 20 OR 30, OR 40.
* SQL> WAQTD NAME, JOB, DEPTNO, SAL OF THE EMPLOYEES WORKING AS CLERK OR SALESMAN IN DEPT 10 OR 30 AND EARNING MORE THAN 1800.
* SQL>WAQTD NAME, JOB AND DEPTNO OF THE EMPLOYEES WORKING AS CLERK OR MANAGER IN DEPT 10.
* SQL> WAQTD NAME, DEPTNO, JOB OF THE EMPLOYEES WORKING IN DEPT 10 OR 20 OR 30 AS A MANAGER.
* SQL>WAQTD NAME AND SAL AND DEPTNO OF THE EMPLOYEES IF EMP GETS MORE THAN 1300 BUT LESS THAN 4000 AND WORKS IN DEPT 20.
* SQL>WAQTD NAME AND DEPTNO OF THE EMPLOYEES IF THE WORKS IN DEPT 10 OR 20
* SQL>WAQTD DETAILS OF THE EMPLOYEE WITH EMPNO 7902, 7782
* SQL>WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK AND EARNING MORE THAN 1500
* SQL>WAQTD NAME AND HIREDATE OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 10.
* SQL>.WAQTD DETAILS OF THE EMP ALONG WITH ANNUAL SALARY IF THEY ARE WORKING IN DEPT 30 AS SALESMAN AND THEIR ANNUAL SALARY HAS TO BE GREATER THAN 14000.
* SQL>WAQTD ALL THE DETAILS OF THE EMP WORKING IN DEPT 30 OR AS ANALYST
* SQL>WAQTD NAMES OF THE EMPMLOYEES WHOS SALARY IS LESS THAN 1100 AND THEIR DESIGNATION IS CLERK.
* SQL>.WAQTD NAME AND SAL, ANNUAL SAL AND DEPTNO IF DEPTNO IS 20 EARNING MORE THAN 1100 AND ANNUAL SALARY EXCEEDS 12000.

**SPECIAL OPERATOR**

* **IN**
* **IS**
* **IS NOT**
* **LIKE**
* **NOT LIKE**
* **NOT IN**
* **BETWEEN**
* **NOT BETWEEN**

IN OPERATOR Is Multi Valued Operator Which Will Can Accept Multiple Values At The RHS.

EAXMPLE:

1=1 (T) 1 IN 1(T)

1=1, 2, 3, 4, 5 (F) 1 IN (1, 2, 3, 4, 5, 6) (T)

1=100,200,300,400 (F) 1 IN (100, 2000, 300,400) (F)

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM TABLE\_NAME**

**WHERE COLUMN\_NAME IN ‘(V1, V2, V3,);**

SQL>WAQTD NAMES AND DEPTNO OF THE EMPLOYEE WORKING IN DEPTNO 10 OR 30

SQL>SELECT ENAME, DEPTNO

2 FROM EMP

3 WHERE DEPTNO = 10 OR DEPTNO =30;

SQL>SELECT ENAME, DEPTNO

2 FROM EMP

3 WHERE DEPTNO IN (10 30);

SQL>WAQTD NAMES AND JOB OF THE EMPLOYEE WORKING AS A CLERK OR MANAGER OR SALAESMAN

SQL>SELECT ENAME, JOB

1. FROM EMP

3 WHERE JOB IN (‘CLERK’,’MANAGER’,’SALESMAN’);

SQL> WAQTD EMPNO, ENAME AND SALARY OF THE EMPLOYEES WHOSE EMPNO IS 7902 OR 7839 AND GETTING SALARY MORE THAN 2725.

SQL>SELECT EMPNO, ENAME, SAL

2 FROM EMP

3 WHERE EMPNO IN (7902, 7839) AND SAL>2725;

**NOT IN**

NOT IN OPERATOR Is Multi Valued Operator Which Will Can Accept Multiple Values At The RHS. Instead Of Selecting It Rejects The Values.

Example:

1 !=1 (F) 1 NOT IN 1(F)

1 !=1, 2, 3, 4, 5 (T) 1 NOT IN (1, 2, 3, 4, 5, 6) (F)

1 !=100,200,300,400 (T) 1 NOT IN (100, 2000, 300,400) (T)

**SYNTAX:**

**SELECT \*/[DISTINCT]COLUMN\_NAME(S) / EXPRESSION/’[ALIAS]**

**FROM TABLE\_NAME**

**WHERE COLUMN\_NAME NOT IN (V1, V2, V3……);**

`

WAQTD name and deptno of all the employees except the employees working in dept. 10 or 40.

SELECT ENAME, DEPTNO

FROM EMP

WHERE DEPTNO NOT IN 10 , 40 ) ;

WAQTD name, deptno and job of the employee working in dept 20 but not as a clerk or Manager.

SELECT ENAME, DEPTNO

FROM EMP

WHERE DEPTNO = 20 AND JOB NOT IN (‘CLERK’,’MANAGER’);

**BETWEEN:** "It is used whenever we have range of values”

[Start value and Stop Value.

Between operator works including the Range

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM TABLE\_NAME**

**WHERE COLUMN\_NAME BETWEEN ‘(V1, V2, V3,);**

WAQTD name and salary of the employees if the EMP is earning Salary in the range 1000 to 3000.

SELECT ENAME, SAL

FROM EMP

WHERE SAL **BETWEEN** 1000 AND 3000;

WAQTD name and deptno of the employees working in dept 10 and hired during 2019 (the entire year of 2019).

SELECT ENAME,, DEPTNO

FROM EMP

WHERE DEPTNO = 10 AND HIREDATE **BETWEEN** '01- JAN-2019' AND '31-DEC-2019’;

WAQTD name, sal and hiredate of the employees hired during 2017 into dept 20 with a salary greater than 2000.

SELECT ENAME, SAL, HIREDATE

FROM EMP

WHERE DEPTNO = 20 AND SAL> 2000 AND HIREDATE**BETWEEN** '01-JAN2017' AND 31-DEC-2017’;

**NOT BETWEEN:** It is Opposite of Between

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM TABLE\_NAME**

**WHERE COLUMN\_NAME NOT BETWEEN ‘(V1, V2, V3,);**

WAQTD name and salary of the employees if the EMP is not earning Salary in the range 1000 to 3000.

SELECT ENAME, SAL

FROM EMP

WHERE SAL NOT BETWEEN 1000 AND 3000;

WAQTD name and deptno of the employees working in dept 10 and not hired during 2019

SELECT ENAME, DEPTNO

FROM EMP

WHERE DEPTNO = 10 AND HIREDATE **NOT BETWEEN** '01- JAN-2019' AND '31-DEC-2019’;

WAQTD name, sal and hiredate of the employees who were not hired during 2017 into dept 20 with a salary greater than 2000S

ELECT ENAME, SAL, HIREDATE

FROM EMP

WHERE DEPTNO IN 20 AND SAL> 2000 AND HIREDATE **NOT BETWEEN** '01-JAN2017'

AND 31-DEC-2017’;

**IS NUL: - "**It is used to compare only NULL**"**

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM EMP**

**WHERE COLUMN\_NAME IS NULL;**

|  |  |  |  |
| --- | --- | --- | --- |
| **EID** | **ENAME** | **SAL** | **COMM** |
| 1 | A | 1000 | 100 |
| 2 | B | null | null |
| 3 | C | null | 200 |
| 4 | D | 2000 | null |

WAQTD name of the employee who is not getting salary.

SELECT ENAME

FROM EMP

WHERE SAL **IS** NULL;

WAQTD name of the Employees who doesn’t get commission.

SELECT ENAME

FROM EMP

WHERE COMM **IS** NULL;

WAQTD name, sal and comm of the EMP if the EMP doesn’t earn both.

SELECT ENAME

FROM EMP

WHERE SAL IS NULL AND COMM IS NULL;

**IS NOT NULL:** "It is used to compare the values with NOT NULL ".

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM EMP**

**WHERE COLUMN\_NAME IS NOT NULL ;**

WAQTD name of the employee who is getting salary.

SELECT ENAME, SAL

FROM EMP

WHERE SAL IS NOT NULL;

WAQTD name of the employees who gets commission.

SELECT ENAME, COMM

FROM EMP

WHERE COMM IS NOT NULL;

WAQTD name, sal and comm of the employees if the emp doesn’t earn commission but gets salary

SELECT ENAME, SAL, COMM

FROM EMP

WHERE COMM IS NULL AND SAL IS NOT NULL;

**LIKE:** "It is used for Pattern Matching ".

To achieve pattern matching we use special characters

* PERCENTILE (%)
* UNDERSCORE (\_)

**SYNTAX**

**SELECT \*/ [DISTINCT] COLUMN\_NAME(s) / EXPRESSION / [ALIAS]**

**FROM EMP**

**WHERE COLUMN\_NAME LIKE ‘pattern’;**