**Database**

**Data­­­­­­­**

* It is any kind of information.
* Example-images, students information, videos, files etc.

**Database**

* It is a Logical storage of data
* A **database** is a collection of information

**DBMS (Database Management System)**

* It is a system or a software used to manage database stored in any format.

**RDBMS (Relational Database Management System)**

* It is a system or a software use to manage data which strictly stored in table format.

**Different type of SQL's statements**

**1. DQL(Data Query Language)**

* Select : Select record from a table.

**2. DDL(Data definition Language)**

* Create : Create new table, or database.
* Alter : Alter existing table, column description.
* Drop : Table and data will be deleted permanently.
* Truncate : It will delete all the records from table but table structure will be as it is.

**3. DML(Data Manipulation Language)**

* Insert : Insert new record.
* Update : Update or modify existing records.
* Delete : Delete existing record.

**4. DCL(Data Control Language)**

* Grant : Allows user to read or write on certain database object.
* Revoke : Keeps user from read/write permission on database object.

**Operators in Database**

* Binary /Arithmetic Operator (+ , - , \* , / )
* Comparison Operator ( > , < , >= , <= , != , in )
* Logical Operator ( and , or , not )
* Pattern matching Operator ( Like )

**Query-**select salary, salary+1000 from hr.Employees

**Output**-It will print salary of employee and salary +1000

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

**In keyword**

In keyword is use to check more than one value.

**Query**-select EMPLOYEE\_ID,SALARY from hr.Employees where EMPLOYEE\_ID in(100,120,140,150)

**Ans:** It will print all the employees whose EmpID is 100,120,140,150

------------------------------------------------------------------------------------------------------------------------- **Like :**

‘A%’- Start with A

‘%a’- Ending with a

‘%a%’- a is anywhere in word

‘\_a\_b%’- 2nd letter will be a and 4th letter will be b

**Query: Select first\_name from hr.employees where first\_name like ‘A%’.**

**Ans:** It will print first name of employees starting with A .

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

**Column Alias :**

* A column alias renames a column heading.
* It requires double quotation marks if it contains spaces or special characters.
* Example-

Select last\_name as Surname from hr.employees.

Select salary,salary\*12 as “Annual Salary” from hr.employees

**Functions**

**1.character**

**a.Case Manipulation**

* upper( ) , lower( ), initcap( ) initial character capital

**b.Character Manipulation**

* length( ), concat( ), substr( ), instr( ), reverse( ), lpad( ), rpad( ), trim( ), replace( )

**2.Number**

* Round( ), trunk( ), mod( )

**3.Date**

* Sysdate, current\_timestamp, localtimestamp, Add\_months( ), Months\_between( )

-----------------------------------------------------------------------------------------------------------------------**Dual is a virtual table which allocates some temporary space during query execution.**

**This table is used whenever we want to work with general data .**

**upper( )**

**Query-**select upper('welcome')from dual

Output-WELCOME

**Query-**select upper(FIRST\_NAME)from hr.Employees

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

**lower( )**

**Query-**select lower('WELCOME')from dual

Output-welcome

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

**initcap( )**

**Query-**select initcap('welcome to pune')from dual

Output-Welcome To Pune.

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

**Length( )**

**Query-**select length('welcome to pune')from dual

**concat( )**

**1.Query**-select concat('welcome' , 'pune')from dual (for 2 words only)

**Output**-welcomepune

**2.Query-**select ‘welcome’||’to’||’pune’ (for more than 2 words)

**3.Query**-select ‘Full name of employee’||employee\_id||’is’||first\_name||last\_name|| from hr.employees where employee\_id between 101 and 120

**Substr( )**

**Query-**select substr('welcome to pune',1,5)from dual

**Output-**welco

**Query-**select 'Full name of employee'||EMPLOYEE\_ID||' '||'is'||' '|| substr(FIRST\_NAME,1,1)||' . '||LAST\_NAME from hr.Employees

**Output-**Full Name of Employee 100 is S.king

-----------------------------------------------------------------------------------------------------------------**instr( )**

**Query:** select instr('welcome to pune','e')from dual

**Output:**2

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

**Reverse( )**

**Query:** select reverse('welcome to pune')from dual

**Output:** enup ot emoclew

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

**Lpad( )**

**Query:** select lpad('1234',6,'0')from dual

**Output:**001234

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

**Rpad( )**

select rpad('1234',6,'\*')from dual

**Output:**1234\*\*

**trim( )**

**Query:** select trim('e' from 'welcome to pune') from dual

**Output:** welcome to pun

-----------------------------------------------------------------------------------------------------------------**replace( )**

**Query:** select replace('Dack and Due','D','Bl')from dual

**Output**: Black and Blue

**Round( )**

**Query**: select round(12345.678,2)from dual

**Output**:12345.68

**Query:** select round(12345.678,-1)from dual

**Output**:12350

-----------------------------------------------------------------------------------------------------------------**Trunc( )**

**Query:** select trunc(12345.678,2)from dual

**Output:**12345.67

**Query:** select trunc(12345.678,-1)from dual

**Output:**12340

----------------------------------------------------------------------------------------------------------------- **mod( )**

**Query:** select mod(100,3)from dual

**Output:**1

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

**Sysdate( )**

**Query:** select sysdate from dual

**Query:** select sysdate , current\_timestamp , localtimestamp from dual

**Output:**29-DEC-17 29-DEC-17 08.21.13.184000 PM +00:00 29-DEC-17 08.21.13.184000 PM

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

**Months\_between ( )**

**Query:** select months\_between('12-Dec-14','12-Oct-12')from dual

**Query:** select round(months\_between(sysdate,'21-Nov-90')/12)from dual

----------------------------------------------------------------------------------------------------------------**Add\_months( )**

**Query:** select Add\_months (sysdate,4)from dual

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

**To\_Char( )**

**Query:** select To\_char (sysdate,'month')from dual

**Output:**January

**Query:** select To\_char (sysdate,'month,year')from dual

**Output:** January twenty eighteen

**Query:** select To\_char (sysdate,'month,year,day')from dual

**Output:** January twenty eighteen,Tuesday

**Query:** select To\_char (sysdate,'yyyy')from dual

**Output**:2018

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

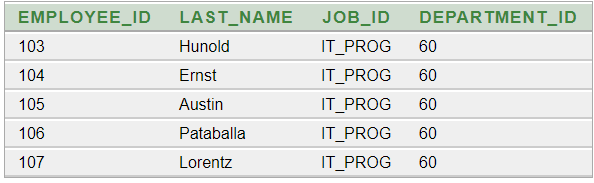
**WHERE Clause:**

* Where clause follows FROM clause.

**Example-**

Select employee\_id, last\_name, job\_id, department\_id from hr.employees **where** department\_id=60;

**Output :**

****

**Comparision Condition :**

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| = | Equal To |
| > | Greater Than |
| >= | Greater Than or equal to |
| < | Less Than |
| <= | Less Than or equal to |
| != | Not equal to |
| Between \_\_and\_\_ | Between two values |
| IN(set) | Match anyone of a list of values |
| LIKE | Match a character pattern |
| IS NULL | Is a null value |

**Example:**

1. Select last\_name ,salary from hr.employees where salary<=3000
2. Select last\_name ,salary from hr.employees where salary>=20000
3. Select last\_name ,salary from hr.employees where salary=7000
4. Select salary from hr.employees where salary between 2500 and 3000
5. R Select last\_name from hr.employees where manager\_id IN(100,101,103)
6. Select last\_name ,salary from hr.employees where first\_name LIKE ‘S%’
7. Select manager\_id from hr.employees where manager\_id is NULL.
8. Select last\_name,job\_id from hr.employees where job\_id NOT IN (‘IT\_PROG’,’ST\_CLERK’)
9. Select employee\_id,last\_name,job\_id,salary from r.employees

Where salary>=10000 or job\_id like ‘%MAN%’;

**Order By Clause :**

* The order by clause can be used to sort the rows.
* The order by clause comes last int the select statement
* **Example -**
* select last\_name, job\_id, department\_id, hire\_date from hr.employees order by hire\_date
* select last\_name, job\_id, department\_id, hire\_date from hr.employees order by hire\_date desc

**To\_Char Function :**

**Syntax:** to\_char(date,’format\_model’)

Where format\_model :

* It must be enclosed by single quotation mark.
* It is case sensitive.
* It Can include any valid date format element.
* It Is separated from the date value by a comma.

**Elements of Date Format Model :**

|  |  |  |
| --- | --- | --- |
| **Elements** | **Result** | **Example** |
| YYYY | Full year in numbers | 2019 |
| YEAR | Year spelled out | Twenty Nineteen |
| MM | Two digit value for month | 12 |
| MONTH | Full name of the month | December |
| MON | Three letter abbreviation of the month | Dec |
| DY | Three letter abbreviation of the day of the week | Fri |
| DAY | Full name of the week | Friday |
| DD | Numeric day of the month | 07 |
| DDth | Th in the last of Numeric day of the month | 7th |
| DDsp | Spelling of Numeric day of the month | seven |
| DDspth | th in last to the spelling of Numeric day of the month | seventh |
| DD “of” MONTH | Word written in “ ” will print as it is | 07 of December |
| HH:MI:SS | Hours:Minutes:Seconds | 11:20:20 |

* **Example:**
  + - Select last\_name , employee\_id ,to\_char(hire\_date,’MM/YY’ from hr.employees Where last\_name=’Higgins’
    - Select to\_char(sysdate,’DD-MM-YYYY’ )from dual

**Nesting Function :**

* + - Single row functions can be nested to any level.
    - Nested functions are evaluated from the innermost level to the outermost level.
    - **Syntax:**

**F3 ( F2 ( F1 ( col , arg1 ) , arg2 ) , arg3 )**

* + - * **Example :**
* Select first\_name , lower(concat(substr(first\_name,1,3),’.’||last\_name)) from hr.employees where first\_name like ‘A%’ and last\_name like ‘%a’

**Group Function :**

* Group function operates on sets of rows to give one result per group.

**Types Of Group Function:**

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

**Syntax :**

Select [ column ] , group\_function( column )

From table

[where condition]

[group by column]

[order by column]

**Example :**

* Select min(salary) , sum(salary) , max(salary) , avg(salary) from hr.employees
* Select min(hire\_date),max(hire\_date) from hr.employees

**COUNT Function**

Count function has 3 formats :

**1.Count(\*) :** It returns the number of rows in a table that satisfy the criteria of the SELECT statement including duplicate rows and rows containing null values in any of the columns.

**Example :**

* Select count(\*) from hr.employees where department\_id=50; (It will display number of employees in the department 50)

**2.Count(expression) :** It returns the number of non-null values that are in the column identified by expression.

**Example :**

* Select count(commission\_pct)from hr.employees where department\_id=80; (It will display the number of employees in the department 80 who earn commission).

**3.Count( DISTINCT expression) :** It returns the number of unique , non-null values that are in the column identified by expression.

**Example :**

* Select count(distinct department\_id)from hr.employees

**Group By Clause :**

* All The columns in the SELECT list that are not in the group functions(for ex-max(salary):except salary column) must be in the group by clause.
* Example:
* Select department\_id, avg(salary)from hr.employees group by department\_id

**Having Clause :**

* **Syntax :**

SELECT column, group-function from table

[where condition]

[group by group\_by\_ expression]

[HAVING group-condition]

[order by column];

* **Example :**
* Select job\_id,sum(salary) Payroll from hr.employees where job\_id like ‘%REP%’ group by job\_id having sum(salary)>13000 order by sum(salary)
* Select job\_id,max(salary) from hr.employees where job\_id like ‘%REP%’ group by job\_id having max(salary)>=5000 order by max(salary) desc

**Joins**

**Types of Joins**

**1.Natural Join**

* The natural join clause is based on all columns in the two tables that have same name.
* It selects rows from two rows that have equal values in all matched columns.
* If the columns having the same name have different data types, an error is returned.
* **Syntax** **:**
* Select table1.column,table2.column from table1 natural join table2
* **Example :**
* Select hr. employees . department\_id , hr. locations . city from hr. employees natural join hr.locations

**2.Inner Join**

* **Example :**
* Select \* from hr.employees inner join (or join)hr.departments on hr.employees.department\_id=hr.departments.department\_id

**3.Outer Join**

* 1. **Left Outer Join**
* **Example :**
* Select \* from hr.employees left join hr.departments on hr.employees.department\_id=hr.departments.department\_id
  1. **Right Outer Join**
* **Example :**
* Select \* from hr.employees right join hr.departments on hr.employees.department\_id=hr.departments.department\_id
  1. **Full Outer Join**
* **Example :**
* Select \* from hr.employees full join hr.departments on hr.employees.department\_id=hr.departments.department\_id

**Table Alias**

* Table alias are used to simplify queries.
* The table alias is valid for only the current select statement.
* **Example :**
* Select e.employee\_id,e.last\_name,d.location\_id from hr.employees e join hr.departments d **using**(department\_id)
* Select e.employee\_id,e.last\_name,d.location\_id ,e.department\_id,d.department\_id from hr.employees e join hr.departments d **on** e.department\_id=d.department\_id

**Subqueries**

* **Syntax:**

Select . . . . .

From table

Where condition(select . . . . from table)

* The subquery is executed once before the main query, then the result returned by the subquery is submitted to the main query.
* The subquery must be enclosed by round brackets.
* Place subqueries on the right side of comparison condition.
* A subquery can not be placed in the group by clause.

**Example-1:**

Select first\_name , last\_name , salary , employee\_id

from hr.employees

where employee\_id= (select employee\_id from hr.employees where employee\_id=110)

**Example-2:**

Select employee\_id,salary

From hr.employees

Where salary IN (select salary from hr.employees Where employee\_id=100)

**To Create Table**

* **Syntax :**
* Create table table\_name ( column1 datatype, column2 datatype ,……. ,……….);
* **Example :**
* Create table student(roll\_no int, name varchar(100),address varchar(100));

**Data Types**

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Varchar | It can store minimum 1 and maximum 2000 bytes |
| Varchar2 | It can store minimum 1 and maximum 4000 bytes |
| Char(10) | Fixed length character data of length 10 bytes (minimum :1 and maximum :2000) |
| Date | A date |
| Datatime | A date time |
| Float | Floating point number |

**Conditions To create Table**

* Table name can be up to 30 char long.
* Table names must begin with alphabet.
* Table names can’t contain single/double quotes.
* Names can contain a-z,0-9,\_,$,#
* Names cannot be reserved words (date, varchar , string ,int )
* **Example:**
* Create table student(id number, name varchar(100),address varchar(100))
* Insert into student values(1,’ABC’,’Pune’)
* Alter table student add state varchar(100)
* Update student set state=’Maharashtra’ where id=1
* Delete from student where id=1;
* Alter table student drop column state
* Update student set column\_name=’’
* Alter table table\_name modify column column\_datatype
* Truncate table student
* Drop table student

**Constraints :**

* Constraint is a rule to which data must confirm.
* Constraint names are optional.
* **Syntax :**

Create table table\_name

(column1 datatype constraint,

column2 datatype constraint,

-----------------)

* **Types of Constraints:**

1.NOT NULL Constraint

2.UNIQUE Constraint

3.PRIMARY KEY Constraint

4.FOREIGN KEY Constraint

5.CHECK Constraint

**1.NOT NULL constraint**

* Not null constraint ensures that null values are not permitted for the column.
* **Example** :create table student (id int not null , name varchar(100)not null).

**2.UNIQUE constraint**

* A unique key constraint requires that every value in a column or set of columns be unique.
* It means no two rows of a table can have duplicate values in a specified column or set of columns.
* **Example** :create table student (id int not null , email varchar(100) unique).

**3.PRIMARY KEY constraint**

* Primary key constraint creates a primary key for the table.
* Only one primary key can be created for each table.
* Primary key constraint is a column or set of columns that uniquely identifies each row in a table.
* **Example** :create table student (id int primary key, name varchar(100))

**4.FOREIGN KEY constraint**

* Foreign key is a column or group of columns within a table that references a key in some other table.
* If the foreign key consists of multiple columns, the foreign key value is considered null if any of its column contains null.
* A foreign key is a key used to link two tables together.
* **Example:**

Create table employee(emp\_id int primary key,name varchar(100),salary int)

Create table department( dept\_id int primary key ,dept\_name varchar(100),emp\_id int references employee(emp\_id))

**5.CHECK constraint**

* Check constraint defines a condition that each row must satisfy.
* A single column can have multiple CHECK constraints.
* **Example:**

Create table student(id int not null,name varchar(100),age int,CHECK(age>=18))

Create table student(id int not null,name varchar(100),age int, address varchar(100),CHECK(age>=18 and address=’pune’))

**Database Objects**

* Table
* View
* Index
* Sequence
* Synonyms

**1.Table:**

Create table empnew as select \* from hr.employees

Select \* from empnew (it will show all records of hr.employees)

Desc empnew ( It will show table structure)

**2.View :**

* View is a database object.
* View is mainly used to reduce query complexity.
* View is called as shadow of a table or virtual table.
* View does not contain its own data, it takes data from one or more base tables.
* If base table is deleted then there will be no use of a view.
* We cannot alter view.
* **Example 1:**

create or replace view myview as select \* from hr.employees

Select \* from viewemp

Drop view viewemp

**3.Index:**

* Index maintained ordered data .
* Index is used to retrieve data fastly.
* Because of index query execution becomes faster.
* **Rules of creating Index :**
  + Create index on a table which contain large amount of data.
  + Don’t create index on a column which is frequently updated.
  + Create index on a column which is frequently used.
* **Syntax**

Create index index\_name on table\_name(column\_name)

* **E­­­xample**

Create index ixname on emp(name)

**4.Sequence :**

* Sequence automatically generates unique number
* It can be ascending or descending.
* **Start with :** This clause specifies from where the sequence starts, this once set cannot be modified. Starting value should be less than maxvalue.
* **Increment by :** This clause specifies the interval between sequence numbers, it can be a positive or negative numbers but can not be zero.

**Example-1**

Create sequence seq1

Start with 1

Increment by 1

Maxvalue 100

* Create table employee (id int, name varchar(100))
* Insert into employee values(seq1.nextval,’ABC’)
* Insert into employee values(seq1.nextval,’PQR’)
* Select seq1.nextval from dual
* Select seq1.nextval from dual

**Example-2**

Create sequence seq2

Start with 10

Increment by -1

Maxvalue 10

Minvalue 1

**5.Synonym :**

* It is a permanent nickname to a table.
* **Example :**
* Create synonym e for hr.employees (select \* from e)
* Create synonym d from hr.departments. (select \* from d)
* Drop synonym e

**Data Control Language (DCL) :**

Grant, revoke

**How to create users**

* Create user user1 identified by pass123
* Grant connect to user1.
* Grant resource to user1.
* Create user user2 identified by pass456
* Grant connect to user2
* Grant resource to user2.
* **Now login to user1**
* Create table emp( eid int, name varchar(100))
* Insert into emp1 values(1,’abc’)
* Insert into emp1 values(2,’pqr’)
* Insert into emp1 values(3,’xyz’)
* Select \* from user2.emp2(table or view does not exists)
* Grant select on emp1 to user2(now check in user2 (select \* from user1.emp1)-it will show the records of the table)
* Revoke select on emp from user2
* **Now login to user2**
* Create table emp2( eid int, address varchar(100))
* Insert into emp2 values(1,’Pune’)
* Insert into emp2 values(2,’Mumbai’)
* Insert into emp2 values(3,’Nashik’)
* Select \* from user1.emp1(table or view does not exists)
* Grant all on emp2 to user1(now check in user1 (select \* from user1.emp1)-it will show records of the table)
* Revoke select on emp2 from user1

**Transaction Control Language (TCL) :**

* Statements used to manage the changes made by DML statements.
* It allows statement to be grouped together into logical instruction.
  + Commit
  + Rollback
  + Savepoint
* **Example**
* Create table t1(id number, name varchar(100))
* Insert into t1 values(1,’ravi’)
* Insert into t1 values(2,’deep’)
* Insert into t1 values(3,’rock’)
* Commit;
* Update t1 set name=’abc’ where id=3; (1 row updated)
* Rollback;
* Select \* from t1;
* Update t1 set name=’rani’ where id=3; (1 row updated)
* Select \* from t1;
* Commit;
* Savepoint s1;
* Insert into t1 values(4,’pqr’);
* Savepoint s2;
* Insert into t1 values(5,’xyz’);
* Savepoint s3;
* Update t1 set name=’Minal’ where id=2;
* Select \* from t1;
* Rollback to s2;

**PL/SQL(Procedural Language/Structured Query Language)**

* PL/SQL is a block of code which contain procedural language+ SQL language.
* PL/SQL is used to perform some extra operations on the fetched data of the table.
* When we execute PL/SQL block it get separated on server side, firstly SQL statement get executed with database then the response of that SQL statement will placed with procedural language and perform some operations and also generate response. That response will display on a client side in normal text format.
* **Syntax:**
  + Declare  *(optional)*
  + <declaration Section>
  + Begin  *(manadatory)*
  + <executable Section>
  + End; *(manadatory)*
* **Example-1 ( Set serveroutput on; )**
  + begin
  + dbms\_output.put\_line (‘Welcome To PL/SQL’);
  + end;

**Loops**

1. Simple Loop
2. While Loop
3. For loop

**1.Simple Loop:**

* + Declare
  + i number:=1;
  + Begin
  + Loop
  + dbms\_output.put\_line(i);
  + i:=i+1;
  + Exit when i>10;
  + End loop;
  + End;

**2.While Loop:**

* + Declare
  + i number:=1;
  + Begin
  + While(i<=10)
  + Loop
  + Dbms\_output.put\_line(i);
  + i:=i+1;
  + End loop;
  + End;

**3.For Loop:**

* + Declare
  + i number:=1;
  + Begin
  + For i in 1 . . 10
  + Loop
  + dbms\_output.put\_line(i\*i);
  + End loop;
  + End;

**Stored Procedure**

* Procedure is a block of code which is stored permanently after compilation.
* When we execute any SQL statements on database, that statement every time compiled and then execute, because of that database performance becomes slow.
* So by using stored procedure we can store such statements permanently into database after compilation.
* In procedure, we can pass input and output type parameters to pass values externally.
* Procedures are created to perform one or more DML operations over database.
* It is used for repeated execution.
* **Example**
* create table sptable(id int,name varchar(100),salary int)
* create or replace procedure spdemo(type varchar,id int,name varchar,salary int)

as

**begin**

if(type='Insert')then

begin

insert into sptable values(id,name,salary);

end;

end if;

if(type='Update')then

begin

update sptable set name=name,salary=salary where id=id;

end;

end if;

if(type='Delete')then

begin

delete from sptable where id=id;

end;

end if;

**end;**

* exec spdemo('Insert',1,'Minal',20000);
* exec spdemo('Insert',2,'Sonal',30000);
* exec spdemo('Insert',3,'Tejal',40000);
* exec spdemo('Insert',4,'Sanju',50000);
* exec spdemo('Insert',5,'Manju',70000);
* exec spdemo('Delete',1,'Minal',20000);
* select \* from sptable;

**CURSOR**

* cursors are like "tables" they may hold multiple rows (records if you will).

1. Create cursor.
2. Open cursor.
3. Fetch data from cursor.
4. Close cursor.

**Example**

* **Declare**

cursor empcursor is

select employee\_id, first\_Name from hr.employees;

eid number;

name varchar(100);

**begin**

open empcursor;

loop

fetch empcursor into eid,name;

dbms\_output.put\_line(eid||' '||name);

exit when empcursor%notfound;

end loop;

close empcursor;

**end**;

**(Note : The %ROWTYPE attribute lets you declare a record that represents a row in a table or view.)**

**Trigger**

* A trigger is a PL/SQL block structure which is fired when a DML statements like Insert, Delete, Update is executed on a database table.
* A trigger is triggered automatically when an associated DML statement is executed.
  + create table tbldemo

( id number,

name varchar(100),

salary number )

* + create table tbldemo\_backup

( id number,

name varchar(100),

salary number,

activity\_time timestamp,

action varchar(100)

)

* create or replace trigger trgtbldemo

after insert on tbldemo

for each row

begin

insert into tbldemo\_backup values(:new.id,:new.name,:new.salary,sysdate,'Insert');

end;

* create or replace trigger trgupdatetbldemo

after update on tbldemo

for each row

begin

insert into tbldemo\_backup values(:old.id,:old.name,:old.salary,sysdate,'Update');

end;

* create or replace trigger trgdeletetbldemo

after delete on tbldemo

for each row

begin

insert into tbldemo\_backup values(:old.id,:old.name,:old.salary,sysdate,'Delete');

end;

* insert into tbldemo values(1,'Ajay',20000)
* update tbldemo set salary =30000 where id=1
* delete from tbldemo where id=1
* select \* from tbldemo;
* select \* from tbldemo\_backup

**END**