CREATING AND MANAGING TABLES

Q.1. Write a query to create a table employee with empno, ename, designation, and salary.

```
CREATE TABLE EMPLOYEE
(
        EMPNO NUMBER (4),
        ENAME VARCHAR2 (10),
        DESIGNATIN VARCHAR2 (10),
        SALARY NUMBER (8,2)
);
```

Q.2. Write a query to display the column name and datatype of the table employee.

```
DESC EMP;
```

Q.3 Write a query to Insert Values in employee table

```
INSERT ALL
  INTO emp  VALUES (1,'akshara','HR',10000)
  INTO emp  VALUES (2,'amey','Manager',20000)
  INTO emp  VALUES (3,'shweta','Developer',30000)
  INTO emp  VALUES (4,'Namita','Tester',40000)
select * from dual;
```

Q4. Write a query for create a table emp1 from an existing table employee with all the fields.

```
CREATE TABLE EMP1 AS SELECT * FROM EMPLOYEE;
```

Q.5. Write a query for create a table emp2 from an existing table with selected fields(empno,ename)

```
CREATE TABLE EMP2 AS SELECT EMPNO, ENAME FROM EMPLOYEE;
```

Q.6. Write a query for create a new table from an existing table without any record

```
CREATE TABLE EMP3 AS SELECT * FROM EMP WHERE  1>2;
```

ALTER

Q.1. Write a Query to Alter the column EMPNO NUMBER (4) TO EMPNO NUMBER (6)

```
ALTER TABLE EMP MODIFY EMPNO NUMBER (6);
```

Q.2. Write a Query to Alter the table employee with multiple columns (EMPNO, ENAME)

```
ALTER TABLE EMP

MODIFY (EMPNO NUMBER (7), ENAME VARCHAR2(12));
```

Q.3. Write a query to add a new column in to employee.

```
ALTER TABLE EMP
ADD QUALIFICATION VARCHAR2(6);
```

Q.4. Write a query to add multiple columns in to employee

```
ALTER TABLE EMP

ADD (DOB DATE, DOJ DATE);
```

Q.5. Write a query to drop a column from an existing table employee

```
ALTER TABLE EMP

DROP COLUMN DOJ;
```

Q.6. Write a query to drop multiple columns from employee

```
ALTER TABLE EMP

DROP (DOB, QUALIFICATION);
```

Q.7. Write a query to rename table emp to employee

```
ALTER TABLE EMP

RENAME TO EMPLOYEE;
```

DROP

Q.1. Write a query to Drop employee table

```
DROP TABLE emp;
```

TRUNCATE

Q.1. Write a query to truncate student table

```
Truncate table student1
```

RENAME

```
RENAME old_table_name TO new_table_name;
```

DATABASE CONSTRAINTS

PRIMARY KEY

Q.1. Write a query to create primary constraints with column level

```
CREATE TABLE EMPLOYEE
(
        EMPNO NUMBER(4) PRIMARY KEY,
        ENAME VARCHAR2(10),
        JOB VARCHAR2(6),
        SAL NUMBER(5),
        DEPTNO NUMBER(7)
);
```

Q.2. Write a query to create primary constraints with table level

```
CREATE TABLE EMPLOYEE2
(
        EMPNO NUMBER(6),
        ENAME VARCHAR2(20),
        JOB VARCHAR2(6),
        SAL NUMBER(7),
        DEPTNO NUMBER(5),
        PRIMARY KEY(empno,ename)
);
```

Q.3. Write a query to create primary constraints with alter command

```
CREATE TABLE EMP3
(
        EMPNO NUMBER(5),
        ENAME VARCHAR2(6),
        JOB VARCHAR2(6),
        SAL NUMBER(6),
        DEPTNO NUMBER(6)
);
```

```
ALTER TABLE EMP3
ADD PRIMARY KEY (EMPNO);
```

FOREIGN KEY

Q.4. Write a query to create foreign key constraints with column level

```
CREATE TABLE DEPT
(
        DEPTNO NUMBER(2) PRIMARY KEY,
        DNAME VARCHAR2(20),
        LOCATION VARCHAR2(15)
);

CREATE TABLE EMP4
(
        EMPNO NUMBER(3),
        DEPTNO NUMBER(2) REFERENCES DEPT(DEPTNO),
        DESIGN VARCHAR2(10)
);
```

Q.5. Write a query to create foreign key constraints with Table level

```
CREATE TABLE DEPT1
(
        DEPTNO NUMBER(2),
        DNAME VARCHAR2(20),
        LOCATION VARCHAR2(15),
        PRIMARY KEY(Deptno,dname)
);

CREATE TABLE EMP5
(
        EMPNO NUMBER(3),
        DEPTNO NUMBER(2),
        DNAME VARCHAR(40),
        DESIGN VARCHAR2(10),
        FOREIGN KEY(DEPTNO,dname) REFERENCES DEPT1(DEPTNO,dname)
```

```
);
```

Q.6.Write a query to create foreign key constraints with Table level with alter command.

```
CREATE TABLE EMP6
(
        EMPNO NUMBER(3),
        DEPTNO NUMBER(2),
        DESIGN VARCHAR2(10)
);

ALTER TABLE EMP6 ADD
FOREIGN KEY(DEPTNO)REFERENCES DEPT(DEPTNO);
```

CHECK CONSTRAINT

Q.7 Write a query to create Check constraints with column level

```
CREATE TABLE EMP7
(
        EMPNO NUMBER(3),
        ENAME VARCHAR2(20),
        DESIGN VARCHAR2(15),
        SAL NUMBER(5)
        CHECK(SAL>500 AND SAL<10000),
        DEPTNO NUMBER(2)
);
```

Q.8. Write a query to create Check constraints using alter command.

```
CREATE TABLE EMP9
(
        EMPNO NUMBER,
        ENAME VARCHAR2(20),
        DESIGN VARCHAR2(15),
        SAL NUMBER(5)
);
```

```
ALTER TABLE EMP9 ADD
CHECK(SAL>500 AND SAL<10001)
```

UNIQUE CONSTRAINT

Q.9. Write a query to create unique constraints with column level

```
CREATE TABLE EMP10
(
        EMPNO NUMBER(3),
        ENAME VARCHAR2(20),
        DESGIN VARCHAR2(15) UNIQUE,
        SAL NUMBER(5)
);
```

Q.10. Write a query to create unique constraints with table level

```
CREATE TABLE EMP11
(
        EMPNO NUMBER(3),
        ENAME VARCHAR2(20),
        DESIGN VARCHAR2(15),
        SAL NUMBER(5),
        UNIQUE(DESIGN, SAL)
);
```

Q.11. Write a query to create unique constraints with table level using alter command.

```
CREATE TABLE EMP12
(
        EMPNO NUMBER(3),
        ENAME VARCHAR2(20),
        DESIGN VARCHAR2(15),
        SAL NUMBER(5)
);

ALTER TABLE EMP12
ADD UNIQUE(DESING,SAL);
```

NOT NULL CONSTRAINT

Q.12. Write a query to create Not Null constraints with column level

```
CREATE TABLE EMP13
(
        EMPNO NUMBER(4),
        ENAME VARCHAR2(20) NOT NULL,
        DESIGN VARCHAR2(20),
        SAL NUMBER(3)
);
```

Q.13. Write a query to create Null constraints with column level

```
CREATE TABLE EMP14
(
    EMPNO NUMBER(4),
    ENAME VARCHAR2(20) CONSTRAINT EMP13_ENAME_NN NOT NULL,
    DESIGN VARCHAR2(20),
    SAL NUMBER(3)
);
```

MANIPULATING DATA

```
create table student
(
        sid int,
        sname varchar (100),
        branch varchar(50),
        marks int
);
```

INSERT

INSERTING A SINGLE RECORD USING THE VALUES KEYWORD

```
INSERT INTO table
VALUES  (expression1, expression2, ... expression_n );
```

Q.1. Write a query to insert values in student table

```
insert into student values( 1, 'namita ','IT', 70);
insert into student values( 2, 'amey','BBA', 50);
insert into student values( 3, 'shweta','Bcom', 80);
insert into student values( 4, 'rohit','IT', 75);
insert into student values( 5, 'siddhesh','IT', 60);
```

INSERTING A SINGLE RECORD FOR SELECTIVE COLUMNS

```
INSERT INTO table (column1, column2, ... column_n )
VALUES (expression1, expression2, ... expression_n );
```

Q.2. Write a query to insert values in student table for only sid and sname

```
insert into student(sid,sname) values( 6, 'harsh');
insert into student(sid,sname) values( 7, 'gauri');
```

INSERTING A MULTIPLE RECORD

```
CREATE TABLE EMP
(
        EMPNO NUMBER (4),
        ENAME VARCHAR2 (10),
        DESIGNATIN VARCHAR2 (10),
        SALARY NUMBER (8,2)
);
```

Q.3. Write a query to insert multiple values in emp table

```
INSERT ALL
          INTO emp  VALUES (1,'akshara','HR',10000)
          INTO emp  VALUES (2,'amey','Manager',20000)
          INTO emp  VALUES (3,'shweta','Developer',30000)
          INTO emp  VALUES (4,'Namita','Tester',40000)
select * from dual;
```

DELETE

TO DELETE ALL VALUES

Q.1. Write a query to delete all the values from the student table;

```
delete from student;
```

TO DELETE SELECTIVE VALUES

Q.2. Write a query to delete student records whose sid is 5 and marks is 60;

```
delete from student
where sid=5 AND marks=60;
```

SELECT

TO SELECT DATA FROM SELECTIVE COLUMN

Q.1. Write a query to fetch student id and name of students;

```
select sid,sname
from student;
```

Q.2. Write a query to select student id, name and marks of student whose marks is 60;

```
select name,marks
from student
where marks=60;
```

Q.3. Write a query to fetch the name,marks and branch of a student whose name is namita.

```
select name,marks,branch
from student
where name='namita';
```

Q.4. Write a query to fetch the faculty data whose department id is 2;

```
Select *
from faculty
where depid=2;
```

Q.5. Write a query to fetch details of faculty whose qualification is B.Tech;

Q.6. Write a query to fetch fid and fname of faculties whose qualification is B.Tech and depid is 1;

Q.7. Write a query to fetch all details of faculties whose qualification is B.Tech and depid is 1;

Q.8. Write a query to fetch student details whose marks are greater than or equal to 75;

Q.9. Write a query to fetch student details whose branch is IT or BBA;

Q.10. Write a query to fetch student details whose marks is either 50,75 or 80;

Q.11.Find students name who is not in BBA

```
select sname
 from student
 where branch != 'BBA';
```

Q.12. Find name of students whose is from IT AND have more than 50 marks

```
select sname
from student
where branch='IT' and marks>50;
```

Q.13.Find name of students whose is from BCom OR have more than 50 marks

```
select sname
from student
where branch='BCom' or marks>50;
```

Q.14.Find faculty names who are from Department 1 or 3

```
select sname
from student
where branch='BCom' or marks>50;
```

Q.15. Find faculty details having either of the following qualification B.Tech, Ph.d

```
Select *
From faculty
Where qualification IN('B.Tech','Ph.D');
```

Q.16. Find student details who are not from IT or BCom branch

```
Select *
From student
Where branch NOT IN ('IT' , 'BCOM');
```

Q.17. Find student details whose marks are between 60 and 90

```
Select *
From student
Where marks BETWEEN 60 and 90;
```

Q.18. Find student details whose marks are not between 50 and 60

```
Select *
From student
Where marks NOT BETWEEN 50 and 60;
```

Q.19. Find student name who is not allocated to any branch

```
Select sname
From student
Where branch IS NULL;
```

Q.20.Find faculty details whose name starts with letter 'A'

```
Select *
From faculty
Where fname like 'A%';
```

Q.21.Find faculty details whose name end with letter 'N'

```
Select *
From faculty
Where fname like '%N'
```

Q.22.Find students marks whose name contain letter 'A' as second letter

```
Select sname,marks
From student
Where sname like '_A%';
```

Q.23.Find students marks whose name starts with letter 'p' and contain 5 letters in name.

```
Select sname, marks
From student
Where sname like 'P____';
```

UPDATE

Q.1. Write a query to update student name to vishal whose student id is 1;

```
update student
set name='vishal'
where sid=1;
```

Q.2. Write a query to update student marks of students by 20 who has student id greater than 3;

```
update student
set marks=marks+20;
where sid>3;
```

Q.3 Add 2 columns percentage and marks2 in student table;

```
Alter table student
Add (marks number, percentage number);
```

Q.4. Write a query to update student percentage calculate percentage and add into percentage column of all students;

```
update student
set percentage=(marks+marks1/200)*100;
```

Q.5. update the percentage of student by 10% whose branch is IT

Q.6 update the percentage of student by -20% whose both subjects marks are between 70 and 80;

Q.7 . add salary column in faculty table.

Q.8. update the salary of a faculty to 20000 whose name starts with ‘a’.

PL/SQL (Loops and Statements)

Q.1.WAP for addition of two numbers in plsql.

```
declare
   a number;
   b number;
   c number;
begin
   a:=&a;
   b:=&b;
   c:=a+b;
   dbms_output.put_line('sum of '||a||' and '||b||' is '||c);
end;
 /
```

Q.2. Write a PL/SQL Program using if condition to check the maximum number.

```
DECLARE
   b number;
   c number;
BEGIN
   B:=10;
   C:=20;
   if(C>B) THEN
      dbms_output.put_line('C is maximum');
   end if;
end;
/
```

Q.3.Write a PL/SQL Program using if and else condition to check if number is greater than 5.

```
declare
n number;
begin
   dbms_output. put_line('enter a number');
   n:=&number;
   if n<5 then
      dbms_output.put_line('entered number is less than 5');
   else
      dbms_output.put_line('entered number is greater than 5');
   end if;
 end;
 /
```

Q.4.Write a PL/SQL Program GREATEST OF THREE NUMBERS USING IF ELSEIF.

```
declare
   a number;
   b number;
   c number;
begin
   a:=&a;
   b:=&b;
   c:=&c;
   if(a>b)and(a>c) then
      dbms_output.put_line('A is maximum');
   elsif(b>a)and(b>c)then
      dbms_output.put_line('B is maximum');
   elsif(c>a)and (c>b)then
      dbms_output.put_line('C is maximum');
   elsif(a=b) and (b=a) and (a<>c) then
      dbms_output.put_line('a and b is equal');
```

```
    elsif(c=a) and (a=c) and (a<>b) then
        dbms_output.put_line('a and c is equal');
    elsif(b=c) and (c=b) and (b<>a) then
        dbms_output.put_line('c and b is equal');
    else
        dbms_output.put_line('values are equal');
    end if;
end;
/
```

Q.5. Write a PL/SQL Program to find Even and odd program.

```
accept num number prompt 'Enter a number';
Declare
    n number:=&num;
Begin
    if mod(n,2)=0 Then
        dbms_output.put_line(n || ' is Even');
    else
        dbms_output.put_line(n || ' is Odd');
    end if;
end;
/
```

Q.6. Write a PL/SQL Program to check if character is vowel or consonants

```
declare
    a char(1):= '&character';
Begin
    if upper(a) in ('A','E','I','O','U') THEN
        dbms_output.put_line('The character is in english vowels');
    else
        dbms_output.put_line('The character is in english consonants');
    end if;
end; /
```

Q.7. Write a PL/SQL Program to take 3 marks from the user and calculate the percentage and accordingly show grades.

```
declare
   a number:=&maths;
   b number:=&English;
   c number:=&marathi;
   percen number:=0;
begin
   percen := ((a+b+c)/300)*100;
   dbms_output.put_line(round(percen));
   IF( percen >= 70) THEN
      dbms_output.put_line('Grade A');
   ELSIF(percen >= 40 AND percen < 70) THEN
      dbms_output.put_line('Grade B');
   ELSIF(percen>=35 AND percen < 40) THEN
      dbms_output.put_line('Grade C');
   Else
       dbms_output.put_line('Fail');
   END IF;
end;
/
```

Q.8. Write a program to print numbers from 1 to 5 using simple loop.

```
declare
   a Number;
begin
   a:=1;
   loop
      DBMS_OUTPUT.PUT_LINE(a);
      a:=a+1;
      EXIT WHEN(a>5);
   end loop;
End;
/
```

Q.9. Write a program to print numbers from 1 to 5 using while loop.

```
DECLARE
   a NUMBER :=1;
BEGIN
   dbms_output.put_line('Program started');
   WHILE (a <= 5)
      LOOP
      dbms_output.put_line(a);
      a:=a+1;
      END LOOP;
      dbms_output.put_line('Program completed' );
END;
/
```

Q.10. Write a program to print table of given number take input from user using while loop.

```
declare
   num number;
   i number;
Begin
   i:=1;
   num:=&number;
   dbms_output.put_line('Multiplication');
   while(i<=10)
      loop
         dbms_output.put_line(i*num);
         i:=i+1;
      end loop;
end;
/
```

Q.11. Write a program to print numbers from 1 to 5 using for loop.

```
BEGIN
  FOR l_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE( l_counter );
  END LOOP;
END;
/
```

Q.12. Write a program to print table of given number take input from user.

```
Declare
   num number;
Begin
   num:=&number;
   dbms_output.put_line('Multiplication');
   for counter In 1..10
      loop
         dbms_output.put_line(counter*num);
      end loop;
end;
/
```

Q.13. Write a program to print table of given number in reverse take input from user.

```
Declare
   num number;
Begin
   num:=&number;
   dbms_output.put_line('Multiplication');
   for counter In reverse 1..10
      loop
         dbms_output.put_line(counter*num);
      end loop;
end;
/
```

EXCEPTION HANDLING

Q.1. NO_DATA_FOUND Exception

```
DECLARE
  temp varchar(20);
BEGIN
  Select employee_name into temp
  from employee2  where employee_name='rohit';

exception
  WHEN no_data_found THEN
    dbms_output.put_line('Specified data not found');
end;
/
```

Q.2. TOO_MANY_ROWS Exception

```
DECLARE
  temp varchar(20);
BEGIN
  SELECT employee_name into temp from employee2;
  dbms_output.put_line(temp);
EXCEPTION
  WHEN too_many_rows THEN
    dbms_output.put_line('error trying to SELECT too many rows');
end;
/
```

Q.3. Value Error Exception

```
DECLARE
  temp number;
BEGIN
  SELECT employee_name into temp from employee2 where
employee_name='KAVI';
  dbms_output.put_line('The employee_name is '||temp);
EXCEPTION
  WHEN value_error THEN
  dbms_output.put_line('Error');
  dbms_output.put_line('Change data type of temp to varchar(20)');
END;/
```

Q.4. Using variable of exception type

```
Declare
   dividend number:=24;
   divisor number:=0;
   result number;
   div_zero exception;
Begin
   If divisor=0 then
      Raise div_zero;
   end if;
   Result:=dividend/divisor;
   dbms_output.put_line('result is:' || Result);
   Exception
      when Div_zero Then
         dbms_output.put_line('Your Divisor is zero');
End;
/
```

Q.5. Raise_application_error method

```
Accept age_v number prompt 'what is your age?';
Declare
   age number:=&age_v;
Begin
   if age < 18 then
      Raise_application_error(-20008,'you should be 18 or above for the
drink');
   end if;
      Dbms_output.put_line('Sure, what would you like to have?');
   Exception when Others then
       Dbms_output.put_line(SQLERRM); --this is the utility function provided
by oracle which retrieve the error msg from last occured exception
end;
/
```

Q.6. Pragma exception_init

```
Accept age_v number prompt 'what is your age?';
Declare
   ex_age exception;
   age Number:=&age_v;
   pragma exception_init(ex_age,-20008);
Begin
   if age<18 then
      Raise_Application_error(-20008,'you should be 18 or above for the
drinks!');
   end if;
   dbms_output.put_line('Sure! what would you like to have?');
   Exception when ex_age then
      dbms_output.put_line(SQLERRM);
end;
/
```

FUNCTIONS

-----SYNTAX OF FUNCTION----------

```
CREATE[OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
   variable declaration;
BEGIN
< function_body >
END [function_name];
```

Q.1. Basic calculator program

```
create or replace function calc(a number, b number,op char)
return number
is
begin
   if op='+' then
      return(a+b);
   elsif op='-' then
      return(a-b);
   elsif op='*' then
      return(a*b);
   else
      return(a/b);
   end if;
end;
/

--1.calling function
   select calc(10,20,'*') "Answer"
   from dual;

--2.calling function from program
declare
   a int;
   b int;
   c char;
   d int;
   begin
```

```
        a:=&enter_first_value;
        b:=&enter_second_value;
        c:='&enter_operator';
        d:=calc(a,b,c);
        dbms_output.put_line('result:'||d);
end;
/
```

Q.1. Factorial of number.

```
DECLARE
  num number;
  factorial number;

   FUNCTION fact(x number)
   RETURN number
      IS
         f number;
      BEGIN
         IF x=0 THEN
            f := 1;
         ELSE
            f := x * fact(x-1);
         END IF;
      RETURN f;
   END;
BEGIN
  num:= &num;
  factorial := fact(num);
  dbms_output.put_line(' Factorial of '|| num || ' is ' || factorial);
END;
/
```

PROCEDURES

Static Procedure

```
create table emp_new
(
   eid number,
   ename varchar(30),
   designation varchar(30),
   salary number,
   dno number
);

insert into emp_new values(1,'abcd','HR',20000,2);

create or replace procedure procedure_static
is
   emp_count int;
begin
   select count(*) into emp_count
   from emp_new;
   dbms_output.put_line('Number of employee: '||emp_count);
end procedure_static;
/
exec procedure_static;
```

Dynamic Procedure

Q. Procedure to display number of employees from given department number.

```
create or replace procedure pro_count(depno int)
is
   e_count int;
begin
   select count(eid) into e_count
   from emp_new
   where dno=depno;
   dbms_output.put_line('Number of employee from depatment
number '||depno||' is '||e_count);
end pro_count; /
```

```
exec pro_count(2);
```

Q.3. write a program to reverse the given number

```
create or replace procedure proc_rev(n int)
is
   rno int;
begin
   dbms_output.put_line('Given number:'|| n);
   for x in reverse 1..length(n)
      loop
         rno:= rno || substr(n,x,1);
      end loop;
    dbms_output.put_line('reverse number: ' || rno);
end proc_rev;
/

exec proc_rev(56789);
```

Q.4. INOUT Parameter

```
create table customer
(
   accno number,
   cust_name varchar(50),
   amount number
);

  insert into customer values (1002,'reshma',20000);

create or replace procedure proc_deposite(vactno in customer.accno%type,
vamt in out customer.amount%type)
```

```
is
begin
   update customer
   set amount=amount+vamt
   where accno=vactno;

   select amount into vamt
   from customer
   where accno=vactno;

   dbms_output.put_line('Account number: ' ||vactno||'  Updated
   amount: '||vamt);
end;
/
```

While Executing:

```
var abc number;
exec :abc :=2000;
exec proc_deposite(1002,:abc);
```

TRIGGERS

----SYNTAX--------

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER } triggering_event ON table_name
[FOR EACH ROW]
[ENABLE / DISABLE ]
[WHEN condition]
DECLARE
   declaration statements
BEGIN
   executable statements
EXCEPTION
   exception_handling statements
END;
```

------Example--------

```
create table customer1
(
   id int,
   name varchar(50),
   age int,
   address varchar(100),
   salary int
);
```

:NEW – It holds a new value for the columns of the base table/view during the trigger execution

:OLD – It holds old value of the columns of the base table/view during the trigger execution

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customer1
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/

INSERT INTO CUSTOMEr1 VALUES (8, 'Krish', 23, 'HPA', 9500.00 );

UPDATE customer1
SET salary = salary + 500
WHERE id = 7;
```

----------BEFORE INSERT TRIGGER----------

```
CREATE TABLE orders
(
   order_id number(5),
  quantity number(4),
  cost_per_item number(6,2),
  total_cost number(8,2),
  create_date date,
  created_by varchar2(10)
);
```

```
CREATE OR REPLACE TRIGGER orders_before_insert
BEFORE INSERT ON orders
FOR EACH ROW

DECLARE
  v_username varchar2(10);

BEGIN

  -- Find username of person performing INSERT into table
  SELECT user INTO v_username
  FROM dual;

  -- Update create_date field to current system date
  :new.create_date := sysdate;

  -- Update created_by field to the username of the person performing the
INSERT
  :new.created_by := v_username;

END;
/
```

----------AFTER INSERT TRIGGER----------

```
create table orders_audit
  (
    order_id int,
    quantity int,
    quantity_after int,
    cost_per_item int,
    total_cost int,
    username varchar(50)
);

drop table orders_audit;
```

```
CREATE OR REPLACE TRIGGER orders_after_insert
AFTER INSERT ON orders
FOR EACH ROW
DECLARE
  v_username varchar2(10);
BEGIN
  -- Find username of person performing the INSERT into the table
  SELECT user INTO v_username
  FROM dual;
  -- Insert record into audit table
  INSERT INTO orders_audit
  (order_id,quantity,cost_per_item,total_cost,username)
  VALUES
  ( :new.order_id,
    :new.quantity,
    :new.cost_per_item,
    :new.total_cost,
    v_username );

END;
/
```

----------BEFORE UPDATE----------

```
CREATE OR REPLACE TRIGGER orders_before_update
BEFORE UPDATE
  ON orders
  FOR EACH ROW

DECLARE
  v_username varchar2(10);
   v_total_cost int;
BEGIN

  -- Find username of person performing UPDATE on the table
  SELECT user INTO v_username
  FROM dual;
```

```
    --update the total cost
     v_total_cost:= :new.quantity * :old.cost_per_item;
    :new.total_cost :=v_total_cost;
   -- Update updated_date field to current system date
   :new.create_date := sysdate;

   -- Update updated_by field to the username of the person performing the
UPDATE
   :new.created_by := v_username;

END;

/
update orders
set quantity=15
where order_id=2;
```

----------AFTER UPDATE----------

```
CREATE OR REPLACE TRIGGER orders_after_update
AFTER UPDATE ON orders
FOR EACH ROW

DECLARE
   v_username varchar2(10);

BEGIN

   -- Find username of person performing UPDATE into table
   SELECT user INTO v_username
   FROM dual;

   -- Insert record into audit table
   INSERT INTO orders_audit
   ( order_id,
     quantity,
     cost_per_item,
     total_cost,
     quantity_after,
```

```
    username )
  VALUES
  ( :new.order_id,
    :old.quantity,
    :new.cost_per_item,
    :new.total_cost,
    :new.quantity,
    v_username );
END;
/
```