INDEX

SQL Queries

Connect:

```
Run SQL Command Line

SQL*Plus: Release 10.2.0.1.0 - Production on Sat May 28 18:53:30 2022

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL> connect
Enter user-name: hr
Enter password:
Connected.
SQL>
```

Show all tables:

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
REGIONS                        TABLE
COUNTRIES                      TABLE
LOCATIONS                      TABLE
DEPARTMENTS                    TABLE
JOBS                           TABLE
EMPLOYEES                      TABLE
JOB_HISTORY                    TABLE
EMP_DETAILS_VIEW               VIEW
T2                             TABLE
BIN$eyhWlhfvR6q5oZuiNZ6SnQ==$0 TABLE
BANK                           TABLE

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
T1                             TABLE

12 rows selected.

SQL> _
```

Create new table in Database:

```
SQL> create table students (r_no number(4), f_name varchar2(10), l_name varchar2(10), class varchar2(5));

Table created.

SQL>
```

Insert into table:

//1st way:

```
SQL> insert into students values(042, 'shubham', 'dahiya', 'bca');

1 row created.

SQL>
```

//2nd way:

```
SQL> insert into students (r_no, f_name) values(049, 'Yash');

1 row created.

SQL>
```

Viewing Data in Tables:

// all rows and columns

```
SQL> select * from students;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         1 anyun      mall       bca
         2 ankit      malik      bca
         3 deepak                bca
         4 kunal      rathi      bca
         5 mohit                 bca
         6 shubham    dahiya     bca
         7 vipul      kumar      bca
         8 ujjwal                bca
         9 yash       arya       bca

9 rows selected.

SQL>
```

// selected rows, all colums;

```
SQL> select * from students where r_no >=4;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         4 kunal      rathi      bca
         5 mohit                 bca
         6 shubham    dahiya     bca
         7 vipul      kumar      bca
         8 ujjwal                bca
         9 yash       arya       bca

6 rows selected.

SQL>
```

// selected columns and all rows;

```
SQL> select f_name, l_name from students;

F_NAME     L_NAME
---------- ----------
anyun      mall
ankit      malik
deepak
kunal      rathi
mohit
shubham    dahiya
vipul      kumar
ujjwal
yash       arya

9 rows selected.

SQL> _
```

// selected rows and selected columns;

```
SQL> select f_name, r_no from students where r_no > 3;

F_NAME           R_NO
---------- ----------
kunal               4
mohit               5
shubham             6
vipul               7
ujjwal              8
yash                9

6 rows selected.
```

<u>Eliminate distinct rows:</u>

First creating duplicate row

```
SQL> insert into students values(9, 'yash', 'arya', 'bca');

1 row created.
```

Now eliminating duplicate row

```
SQL> select distinct * from students;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         4 kunal      rathi      bca
         7 vipul      kumar      bca
         2 ankit      malik      bca
         9 yash       arya       bca
         8 ujjwal                bca
         3 deepak                bca
         6 shubham    dahiya     bca
         1 anyun      mall       bca
         5 mohit                 bca

9 rows selected.
```

Sorting:

//Ascending by r_no, then by f_name

```
SQL> select distinct * from students order by r_no asc, f_name;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         1 anyun      mall       bca
         2 ankit      malik      bca
         3 deepak                bca
         4 kunal      rathi      bca
         5 mohit                 bca
         6 shubham    dahiya     bca
         6 shubham               bca
         7 vipul      kumar      bca
         8 ujjwal                bca
         9 yash       arya       bca

10 rows selected.

SQL>
```

//Descending by r_no, then by f_name

```
SQL> select distinct * from students order by r_no desc, f_name;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         9 yash       arya       bca
         8 ujjwal                bca
         7 vipul      kumar      bca
         6 shubham    dahiya     bca
         6 shubham               bca
         5 mohit                 bca
         4 kunal      rathi      bca
         3 deepak                bca
         2 ankit      malik      bca
         1 anyun      mall       bca

10 rows selected.

SQL>
```

<u>UPDATE Contents of a table:</u>

Update all rows (class from ‘bca’ to ‘B.C.A’)

```
SQL> update students set class = 'B.C.A';

11 rows updated.
```

Output:

```
SQL> select * from students;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         1 anyun      mall       B.C.A
         2 ankit      malik      B.C.A
         3 deepak                B.C.A
         4 kunal      rathi      B.C.A
         5 mohit                 B.C.A
         6 shubham    dahiya     B.C.A
         7 vipul      kumar      B.C.A
         8 ujjwal                B.C.A
         9 yash       arya       B.C.A
         9 yash       arya       B.C.A
         6 shubham               B.C.A

11 rows selected.
```

Update records conditionally (from ‘bca’ to ‘B.C.A’)

```
SQL> update students set r_no = 42 where r_no = 6;

2 rows updated.
```

OUTPUT:

```
SQL> select * from students;

      R_NO F_NAME     L_NAME     CLASS
---------- ---------- ---------- -----
         1 anyun      mall       B.C.A
         2 ankit      malik      B.C.A
         3 deepak                B.C.A
         4 kunal      rathi      B.C.A
         5 mohit                 B.C.A
        42 shubham    dahiya     B.C.A
         7 vipul      kumar      B.C.A
         8 ujjwal                B.C.A
         9 yash       arya       B.C.A
         9 yash       arya       B.C.A
        42 shubham               B.C.A

11 rows selected.
```

MOIDFYING STRUCTURE OF TABLE

Adding new columns

```
SQL>
SQL> alter table students add (email varchar2(30));

Table altered.
```

OUTPUT:

```
SQL> select * from students;

      R_NO F_NAME     L_NAME     CLASS EMAIL
---------- ---------- ---------- ----- ------------------------------
         1 anyun      mall       B.C.A
         2 ankit      malik      B.C.A
         3 deepak                B.C.A
         4 kunal      rathi      B.C.A
         5 mohit                 B.C.A
        42 shubham    dahiya     B.C.A
         7 vipul      kumar      B.C.A
         8 ujjwal                B.C.A
         9 yash       arya       B.C.A
         9 yash       arya       B.C.A
        42 shubham               B.C.A

11 rows selected.

SQL> _
```

Modifying existing columns

```
SQL>
SQL> alter table students modify (email varchar2(25));

Table altered.

SQL>
```

OUTPUT:

```
SQL> select * from students;

      R_NO F_NAME     L_NAME     CLASS EMAIL
---------- ---------- ---------- ----- ------------------------
         1 anyun      mall       B.C.A
         2 ankit      malik      B.C.A
         3 deepak                B.C.A
         4 kunal      rathi      B.C.A
         5 mohit                 B.C.A
        42 shubham    dahiya     B.C.A
         7 vipul      kumar      B.C.A
         8 ujjwal                B.C.A
         9 yash       arya       B.C.A
         9 yash       arya       B.C.A
        42 shubham               B.C.A

11 rows selected.

SQL> _
```

Renaming table

```
SQL> rename students to students_data;

Table renamed.
```

DELETING ROWS

Deleting duplicate r_no = 42 with l_name = null;

```
SQL> delete from students_data where r_no = 42 and l_name is null;

1 row deleted.

SQL> _
```

OUTPUT:

```
      R_NO F_NAME     L_NAME     CLASS EMAIL
---------- ---------- ---------- ----- -------------------------
         1 anyun      mall       B.C.A
         2 ankit      malik      B.C.A
         3 deepak                B.C.A
         4 kunal      rathi      B.C.A
         5 mohit                 B.C.A
        42 shubham    dahiya     B.C.A
         7 vipul      kumar      B.C.A
         8 ujjwal                B.C.A
         9 yash       arya       B.C.A
         9 yash       arya       B.C.A

10 rows selected.

SQL> _
```

Delete all rows

```
SQL> delete from students_data;

10 rows deleted.
```

OUTPUT:

```
SQL> select * from students_data;

no rows selected
```

Deleting Table

```
SQL> drop table students_data;

Table dropped.
```

OUTPUT:

```
SQL> select * from students_data;
select * from students_data
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

DATA CONSTRAINTS

Two types:

- I/O Constraint
- Business Rule Constraint

I/O Constraint:

- **Primary Key**

  Two ways to create primary key – at column level, at table level;

  -Column level

```
SQL> create table department ( dno number(5) primary key, dname varchar2(20));

Table created.
```

  -Table level

```
SQL> create table department ( dno number(5) , dname varchar2(20), primary key(dno));

Table created.

SQL>
```

- **Foreign Key**

  Columns whose value are derived from primary key

  Two ways to create primary key – at column level, at table level;

-Column level:

```
SQL> create table department ( dno number(5) , dname varchar2(20), primary key(dno));

Table created.

SQL> create table employee ( eno number(5), ename varchar2(20), dno number(5) references department);

Table created.
```

Table Level

```
SQL> create table employee ( eno number(5), ename varchar2(20), dno number(5), foreign key(dno) references department(dno));

Table created.

SQL>
```

OUTPUT:

Department table;

```
SQL> select * from department;

       DNO DNAME
---------- --------------------
       810 MCA
       811 MBA
       812 B.Tech
```

Employee table;

```
SQL> select * from employee;

       ENO ENAME                       DNO
---------- -------------------- ----------
       500 shubham                     810
       501 vipul                       810
       502 ankit                       812
       503 gourav                      811
```

Error on inserting inserting values out of dno range in employee table;

```
SQL> insert into employee values(503, 'devender', 815);
insert into employee values(503, 'devender', 815)
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.SYS_C004268) violated - parent key not
found
```

- **Unique Key**

-Column level

```
SQL> create table employee(eno number(5) unique, ename varchar2(20));

Table created.

SQL>
```

-Table level

```
SQL> create table employee(eno number(5), ename varchar2(20), unique(eno));

Table created.

SQL>
```

OUTPUT:

Error on inserting duplicate ENO;

```
SQL> select * from employee;

       ENO ENAME
---------- --------------------
         1 shubham
         2 ankit

SQL> insert into employee values(1, 'shubham');
insert into employee values(1, 'shubham')
*
ERROR at line 1:
ORA-00001: unique constraint (HR.SYS_C004267) violated


SQL>
```

Business Rule Constraint

Not Null

Only applied at column level

```
SQL> create table students(roll_no number(3) not null, name varchar2(10), class varchar(10));

Table created.

SQL>
```

OUTPUT:

```
SQL> select * from students;

   ROLL_NO NAME       CLASS
---------- ---------- ----------
         1 anyun      B.C.A
         2 ankit      B.C.A
         3 shubham    B.C.A
```

Error on inserting null values in not null column

```
SQL> insert into students values( null, 'shubham', 'B.C.A');
insert into students values( null, 'shubham', 'B.C.A')
                            *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("HR"."STUDENTS"."ROLL_NO")
```

Views

Customers table:

```
SQL> select * from customers;

        ID NAME       AG ADDRESS              SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6500
         2 anyun      22 up                    7000
         3 aaditya    21 delhi                 5900
         4 ankit      19 ganuar                5500
         5 vipul      18 jind                  5400
```

Creating View:

```
SQL> create view v1 as select name, age from customers;

View created.
```

OUTPUT:

```
SQL> select * from v1;

NAME       AG
---------- --
shubham    20
anyun      22
aaditya    21
ankit      19
vipul      18

SQL>
```

Renaming Column of a view:

Renaming name column to f_name for view:

```
SQL> create or replace view v1 as select name f_name, age from customers;

View created.

SQL>
```

OUTPUT:

```
SQL> select * from v1;

F_NAME     AG
---------- --
shubham    20
anyun      22
aaditya    21
ankit      19
vipul      18

SQL>
```

Updating views:

For updating views, the primary key + all not null must be included in the view.

Insert command

```
SQL> insert into v1 values('Lakshay', 20 );

1 row created.
```

Update command

```
SQL> update v1 set age = 26 where f_name ='Lakshay';

1 row updated.
```

OUTPUT:

Customers table:

```
SQL> select * from customers;

        ID NAME       AG ADDRESS              SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6500
         2 anyun      22 up                    7000
         3 aaditya    21 delhi                 5900
         4 ankit      19 ganuar                5500
         5 vipul      18 jind                  5400
           Lakshay    26

6 rows selected.
```

Delete command:

```
SQL> delete from v1 where f_name = 'Lakshay';

1 row deleted.
```

OUTPUT:

Customers table:

```
SQL> select * from customers;

        ID NAME       AG ADDRESS             SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6500
         2 anyun      22 up                    7000
         3 aaditya    21 delhi                 5900
         4 ankit      19 ganuar                5500
         5 vipul      18 jind                  5400

SQL>
```

Dropping view:

```
SQL> drop view v1;

View dropped.

SQL>
```

PL/SQL

Problem Statement:

Program to find greatest among three

Program:

```
declare
a number;
b number;
c number;

begin
a := &a;
b := &b;
c:= &c;

dbms_output.put_line(' ----- ---- --- -- ');

if a>b then
if a>c then
dbms_output.put_line('a is greatest');
else
```

```
dbms_output.put_line('c is greatest');
end if;

else
if b>c then
dbms_output.put_line('b is greatest');
else
dbms_output.put_line('c is greatest');
end if;
end if;

end;
```

OUTPUT:

```
SQL> @ shubham_p01;
 30  /
Enter value for a: 12
old   7: a := &a;
new   7: a := 12;
Enter value for b: 10
old   8: b := &b;
new   8: b := 10;
Enter value for c: 14
old   9: c:= &c;
new   9: c:= 14;
----- ---- --- --
c is greatest

PL/SQL procedure successfully completed.

SQL>
```

<u>Problem Statement:</u>

Program to print 1 to 10 using while loop

Program:

```
declare
i number;

begin
i := 1;

while i<=10
loop
dbms_output.put_line(i);
i:= i+1;
end loop;

end;
```

OUTPUT:

```
SQL> @ shubham_p02;
 14   /
1
2
3
4
5
6
7
8
9
10

PL/SQL procedure successfully completed.
```

Problem Statement:

Program to print factorial of a number

Program:

```
declare
num number;
result number;

begin
result := 1;
num := &num;

while num > 0
loop
```

```
result := result * num;

num := num - 1;


end loop;


dbms_output.put_line('----------------------------------');


dbms_output.put('factorial: ');


dbms_output.put_line(result);


end;
```

OUTPUT:

```
SQL> @ shubham_p03;
 22  /
Enter value for num: 5
old   7: num := &num;
new   7: num := 5;
----------------------------------
factorial: 120

PL/SQL procedure successfully completed.

SQL>
```

Problem Statement:

Program to print 1 to 10 using for loop

Program:

```
begin

for i in 1..10
loop
dbms_output.put_line(i);
end loop;

end;
```

OUTPUT:

```
SQL> @ shubham_p04;
  9  /
1
2
3
4
5
6
7
8
9
10

PL/SQL procedure successfully completed.

SQL>
```

<u>Problem Statement:</u>

Program to print if number is even or odd using goto statment;

Program:

```
declare
num number;

begin
num := &num;

if mod(num,2) = 0 then
goto even;

else
goto odd;
end if;

<<even>>
dbms_output.put_line('num is even');
goto programend;

<<odd>>
dbms_output.put_line('num is odd');
```

```
<<programend>>

dbms_output.put_line('----------------------');


end;
```

OUTPUT:

```
SQL> @shubham_p05;
 24   /
Enter value for num: 7
old   5: num := &num;
new   5: num := 7;
num is odd
----------------------

PL/SQL procedure successfully completed.

SQL> @shubham_p05;
 24   /
Enter value for num: 8
old   5: num := &num;
new   5: num := 8;
num is even
----------------------

PL/SQL procedure successfully completed.

SQL>
```

Problem Statement:

Program to print if number is positive or negative

Program:

```
declare
num number;

begin
num := &num;
if num > 0 then
dbms_output.put_line( ' num is positive');
else
dbms_output.put_line( ' num is negative');

end if;
end;
```

OUTPUT:

```
SQL> ed shubham_p06;

SQL> @ shubham_p06;
Enter value for num: -4
old   5: num := &num;
new   5: num := -4;
num is negative

PL/SQL procedure successfully completed.

SQL> @ shubham_p06;
Enter value for num: 8
old   5: num := &num;
new   5: num := 8;
num is positive

PL/SQL procedure successfully completed.

SQL>
```

Problem Statement:

Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs. 2000 from the account if the account has a minimum balance of 500 after the amount is debited. The process is to be fired on the Accounts table.

Program:

```
declare
acc_no accounts.acc_id%type;
balance accounts.balance%type;

begin
acc_no := '&acc_no';

select balance into balance from accounts where acc_id = acc_no;

if balance >=2500 then
update accounts set balance = balance - 2000 where acc_id = acc_no;
dbms_output.put_line('Rs.2000 debited from Acc_Id: ' || acc_no );

else
dbms_output.put_line('amount not debited. Account balance must be above
Rs.2500');

end if;
end;
```

OUTPUT:

Initially accounts table

```
SQL> select * from accounts;

ACC_ID     NAME          BALANCE
---------- ---------- ----------
AC001      Shubham          6000
AC002      Ankit            4000
AC003      Vipul            2400
AC004      Yash             1400

SQL>
```

After program execution for AC002

```
SQL> @ accounts;
 21  /
Enter value for acc_no: AC002
old   6: acc_no := '&acc_no';
new   6: acc_no := 'AC002';
Rs.2000 debited from Acc_Id: AC002

PL/SQL procedure successfully completed.

SQL> select * from accounts;

ACC_ID     NAME          BALANCE
---------- ---------- ----------
AC001      Shubham          6000
AC002      Ankit            2000
AC003      Vipul            2400
AC004      Yash             1400
```

After program execution for AC002 again

```
SQL> @ accounts;
 21  /
Enter value for acc_no: AC002
old   6: acc_no := '&acc_no';
new   6: acc_no := 'AC002';
amount not debited. Account balance must be above Rs.2500

PL/SQL procedure successfully completed.
```

After program execution for AC004

```
SQL> @ accounts;
 21  /
Enter value for acc_no: AC004
old   6: acc_no := '&acc_no';
new   6: acc_no := 'AC004';
amount not debited. Account balance must be above Rs.2500

PL/SQL procedure successfully completed.
```

After program execution for AC001

```
SQL>
SQL> @ accounts;
 21  /
Enter value for acc_no: AC001
old   6: acc_no := '&acc_no';
new   6: acc_no := 'AC001';
Rs.2000 debited from Acc_Id: AC001

PL/SQL procedure successfully completed.
```

Table Output:

```
SQL> select * from accounts;

ACC_ID     NAME          BALANCE
---------- ---------- ----------
AC001      Shubham          4000
AC002      Ankit            2000
AC003      Vipul            2400
AC004      Yash             1400
```

Problem Statement:

Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in a table, Areas.

Program:

```
declare

pi constant number(4,2) := 3.14;
r areas.radius%type;
a areas.area%type;

begin

r:= 3;

while r<=7
loop

a := pi * power(r, 2);

insert into areas values(r, a);

r:= r+1;
```

```
end loop;

end;
```

OUTPUT:

Before areas table;

```
SQL> select * from areas;

no rows selected
```

After executing program

```
SQL> select * from areas;

    RADIUS       AREA
---------- ----------
         3      28.26
         4      50.24
         5       78.5
         6     113.04
         7     153.86

SQL> ed areas;

SQL>
```

Problem Statement:

Write a PL/SQL block of cod to achieve the following: If the price of product 'P00001' is less than 4000, then change the price to 4000. The price change is to be recorded in the old_price_table along with the product_no and the date on which the price was last changed.

Program:

```
declare
product_noo product_master.product_no%type;
product_pricee product_master.sell_price%type;
old_price number(6);

begin

product_noo := '&product_noo';

select sell_price into old_price from product_master where product_no =
product_noo;

dbms_output.put_line('old_price: ' || old_price);

if old_price < 4000 then
goto lessthen;

else
```

```
insert into product_master values(product_noo, product_pricee);



end if;

<<lessthen>>

update product_master set sell_price = 4000 where product_no = product_noo;

insert into old_price_table values(product_noo, sysdate, old_price);

dbms_output.put_line(' the price of product is 4000');

end;
```

OUTPUT:

Product_master table:

```
SQL> select * from product_master;

PRODUCT_NO SELL_PRICE
---------- ----------
P00001           3200
P00002           4000
P00003           6000
P00004           9000
P00005           2800
```

Old_price_table:

```
SQL> select * from old_price_table;

no rows selected
```

Running program for 'P00005' & 'P00001'

```
SQL> @ price;
  32  /
Enter value for product_noo: P00005
old   8: product_noo := '&product_noo';
new   8: product_noo := 'P00005';
old_price: 2800
the price of product is 4000

PL/SQL procedure successfully completed.
```

```
SQL> @ price;
  32  /
Enter value for product_noo: P00001
old   8: product_noo := '&product_noo';
new   8: product_noo := 'P00001';
old_price: 3200
the price of product is 4000

PL/SQL procedure successfully completed.
```

Product_master table:

```
SQL> select * from product_master;

PRODUCT_NO SELL_PRICE
---------- ----------
P00001           4000
P00002           4000
P00003           6000
P00004           9000
P00005           4000

SQL>
```

Old_price_table

```
SQL> select * from old_price_table;

PRODUCT_NO DATE_CHAN  OLD_PRICE
---------- --------- ----------
P00005     01-JUN-22       2800
P00001     01-JUN-22       3200

SQL>
```

Cursors

Two types:

- Implicit Cursors
- Explicit Cursors

<u>Implicit Cursors:</u>

Initially Customers table:

```
SQL> select * from customers;

        ID NAME       AG ADDRESS             SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6000
         2 anyun      22 up                    6500
         3 aaditya    21 delhi                 5400
         4 ankit      19 ganuar                5000
         5 vipul      18 jind                  4900
```

Program to increment salary by 500:

begin

```
update customers set salary = salary + 500;



if sql%notfound then

dbms_output.put_line('no records updated');


elsif sql%found then

dbms_output.put_line( sql%rowcount || ' rows updated');


end if;


end;
```

OUTPUT:

```
SQL> @ implicit_cursor.sql
 15  /
5 rows updated

PL/SQL procedure successfully completed.

SQL>
```

Now customers table

```
SQL> select * from customers;

        ID NAME       AG ADDRESS              SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6500
         2 anyun      22 up                    7000
         3 aaditya    21 delhi                 5900
         4 ankit      19 ganuar                5500
         5 vipul      18 jind                  5400

SQL>
```

Explicit Cursors:

Customers table

```
SQL> select * from customers;

        ID NAME       AG ADDRESS              SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6000
         2 anyun      22 up                    6500
         3 aaditya    21 delhi                 5400
         4 ankit      19 ganuar                5000
         5 vipul      18 jind                  4900
```

Program:

```
declare
c_id customers.id%type;
c_name customers.name%type;
c_address customers.address%type;

cursor mycursor is select id, name, address from customers;

begin

open mycursor;

loop
```

```
fetch mycursor into c_id, c_name, c_address;

exit when mycursor%notfound;


dbms_output.put_line('id: ' || c_id ||'  name: ' || c_name || '  address: ' ||
c_address );


end loop;


close mycursor;

end;
```

OUTPUT:

```
SQL> @ explicit_cursor.sql
 23  /
id: 1   name: shubham   address: sonipat
id: 2   name: anyun   address: up
id: 3   name: aaditya   address: delhi
id: 4   name: ankit   address: ganuar
id: 5   name: vipul   address: jind

PL/SQL procedure successfully completed.

SQL>
```

Triggers

Customers table:

```
SQL> select * from customers
  2  /

        ID NAME       AG ADDRESS              SALARY
---------- ---------- -- --------------- ----------
         1 shubham    20 sonipat               6500
         2 anyun      22 up                    7000
         3 aaditya    21 delhi                 5900
         4 ankit      19 ganuar                5500
         5 vipul      18 jind                  5400
```

trigger.sql Program:

```
create or replace trigger salarydifference before delete or insert or update on
customers for each row

declare

sal_diff number(7);

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;
```

OUTPUT:

Inserting values

```
SQL> insert into customers values(6, 'yash', '20','rohtak' , 6000);
old salary:
new salary: 6000
salary difference:

1 row created.
```

Updating values

```
SQL> update customers set salary = 5000 where id = 1;
old salary: 6500
new salary: 5000
salary difference: -1500

1 row updated.
```

Deleting values:

```
SQL> delete from customers where name = 'aaditya';
old salary: 5900
new salary:
salary difference:

1 row deleted.
```

Procedures & Functions

Standalone procecdures

```
CREATE OR REPLACE PROCEDURE firstProcedure
AS
BEGIN
  dbms_output.put_line('Hello Shuham!');
END;
```

OUTPUT:

```
SQL> @firstProcedure
  6  /

Procedure created.

SQL>
```

Executing firstProcedure

```
SQL> execute firstProcedure
Hello Shuham!

PL/SQL procedure successfully completed.

SQL>
```

Deleting Standalone Procedure

```
SQL> drop procedure firstProcedure
  2  /

Procedure dropped.
```

IN OUT MODE in Procedures

```
DECLARE
  a number;
  b number;
  c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
  IF x < y THEN
    z:= x;
  ELSE
    z:= y;
  END IF;
END;
BEGIN
  a:= 23;
  b:= 45;
  findMin(a, b, c);
  dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
```

OUTPUT:

```
SQL> @ procedure_in_out
 19  /
Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.

SQL>
```

Function to find factorial

```
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:= 6;
  factorial := fact(num);
  dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
```

OUTPUT:

```
SQL> @ funciton_factorail
 23  /
Factorial 6 is 720

PL/SQL procedure successfully completed.

SQL>
```