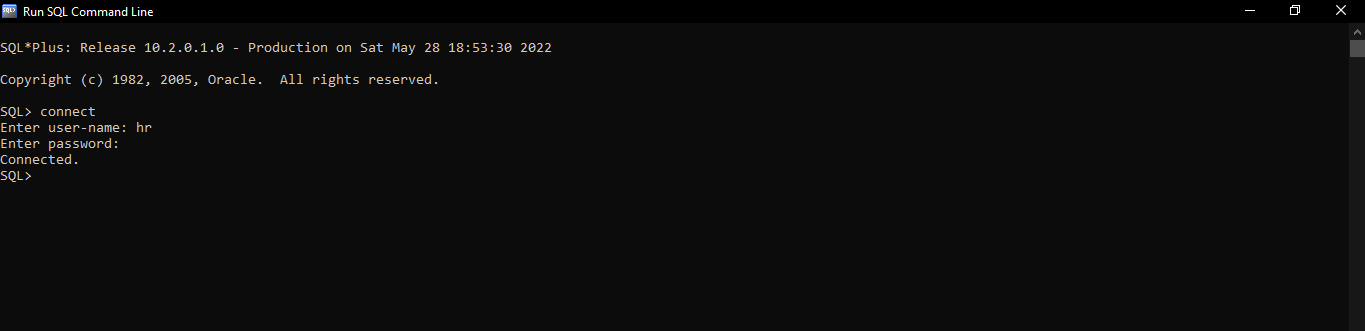
**INDEX**

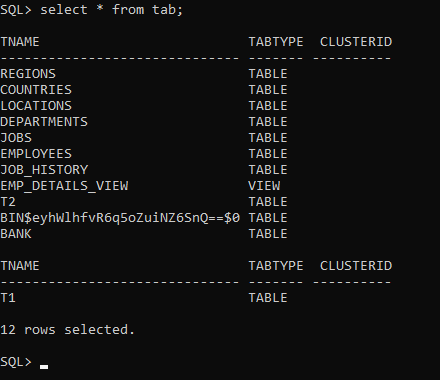
|  |  |  |  |
| --- | --- | --- | --- |
| **S.no.** | **Title** | **Page No.** | **Remarks** |
|  | Connect | 1 |  |
|  | Show all tables | 1 |  |
|  | Create new table in Database | 2 |  |
|  | Insert into table | 2 |  |
|  | Viewing Data in Tables | 3 |  |
|  | Eliminatedistinct rows | 4 |  |
|  | Sorting | 5 |  |
|  | UPDATE Contents of a table | 6 |  |
|  | MOIDFYING STRUCTURE OF TABLE:  Adding new columns, Modifying existing columns, Renaming table, DELETING ROWS, Delete all rows, Deleting Table | 8 |  |
| I/O Constraint: | | | |
|  | Primary Key | 12 |  |
|  | Foreign Key | 12 |  |
|  | Unique Key | 14 |  |
| Business Rule Constraint | | | |
|  | Not Null | 15 |  |
|  | Creating View: | 16 |  |
|  | Renaming Column of a view | 17 |  |
|  | Updating views: insert, update and delete | 18 |  |
|  | Dropping view | 19 |  |
| PL/SQL | | | |
|  | Program to find greatest among three | 20 |  |
|  | Program to print 1 to 10 using while loop | 22 |  |
|  | Program to print factorial of a number | 23 |  |
|  | Program to print 1 to 10 using for loop | 25 |  |
|  | Program to print if number is even or odd using goto statement. | 26 |  |
|  | Program to print if number is positive or negative. | 27 |  |
|  | Program to debit an amount of Rs. 2000 from account if the account has a minimum balance of 500 after the amount is debited. | 29 |  |
|  | Calculating and storing value of areas varying from radius 3 to 7 into table Areas. | 32 |  |
|  | 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. | 34 |  |
|  | Implicit Cursors | 37 |  |
|  | Explicit Cursors | 39 |  |
|  | Triggers | 41 |  |
|  | Standalone Procecdures | 43 |  |
|  | Procedures IN OUT program | 44 |  |
|  | Function to find factorial | 45 |  |

**SQL Queries**

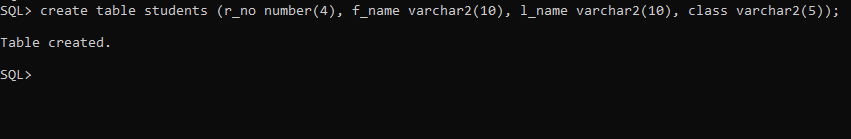
**Connect:**



**Show all tables:**

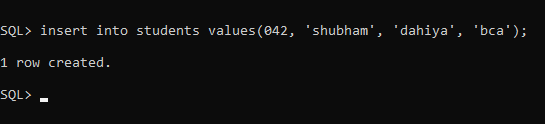


**Create new table in Database:**

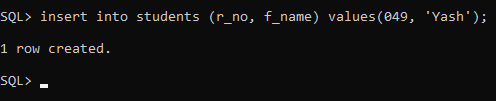


**Insert into table:**

//1st way:

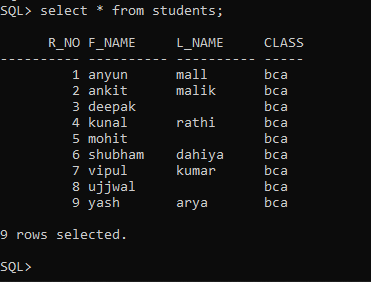


//2nd way:

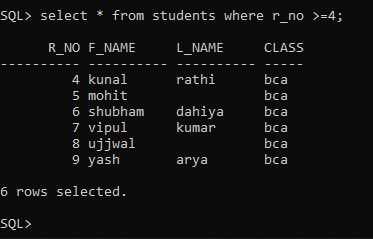


**Viewing Data in Tables:**

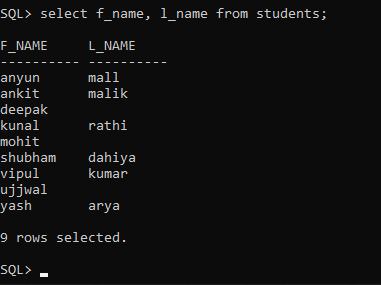
// all rows and columns



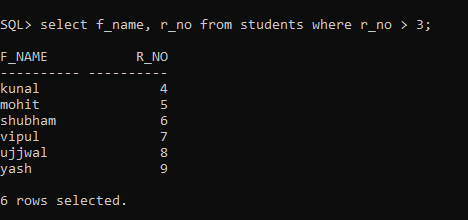
// selected rows, all colums;



// selected columns and all rows;

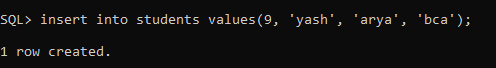


// selected rows and selected columns;

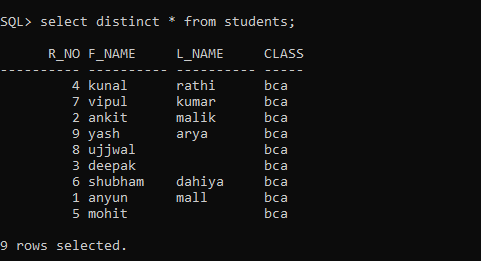


**Eliminate distinct rows:**

First creating duplicate row

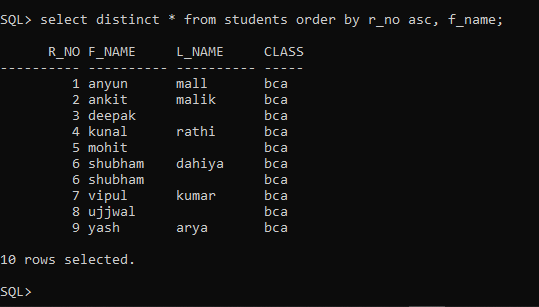


Now eliminating duplicate row

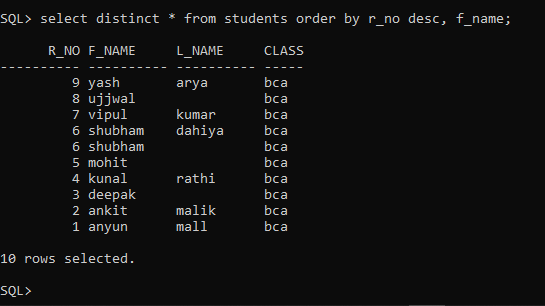


**Sorting:**

//Ascending by r\_no, then by f\_name

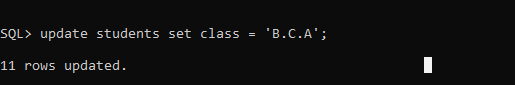


//Descending by r\_no, then by f\_name

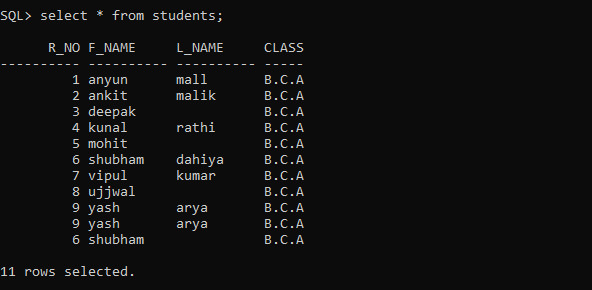


**UPDATE Contents of a table:**

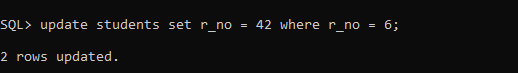
Update all rows ( class from ‘bca’ to ‘B.C.A’ )



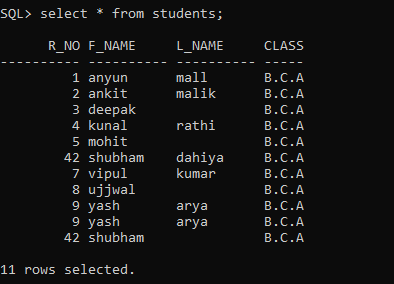
**Output:**



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

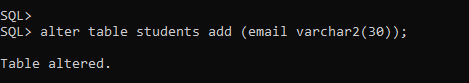


**OUTPUT:**

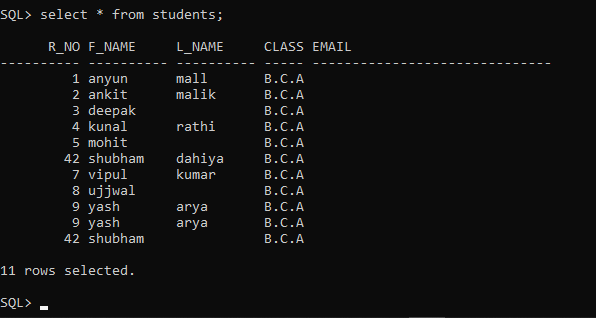
**herer**

**MOIDFYING STRUCTURE OF TABLE**

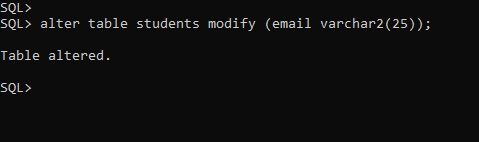
**Adding new columns**



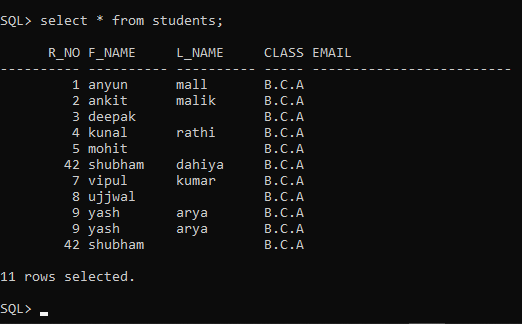
**OUTPUT:**



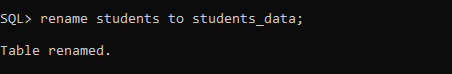
**Modifying existing columns**



**OUTPUT:**

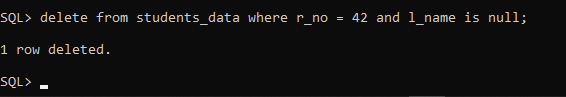


**Renaming table**

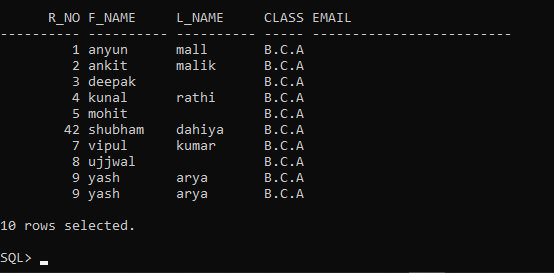


**DELETING ROWS**

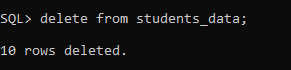
Deleting duplicate r\_no = 42 with l\_name = null;



**OUTPUT:**



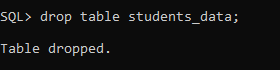
**Delete all rows**



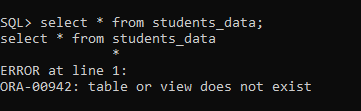
**OUTPUT:**



**Deleting Table**



**OUTPUT:**



**DATA CONSTRAINTS**

Two types:

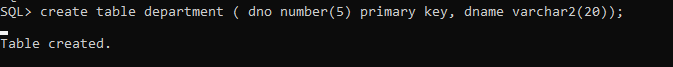
* I/O Constraint
* Business Rule Constraint

**I/O Constraint:**

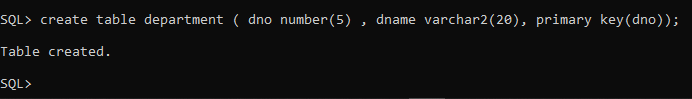
* **Primary Key**

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

*-Column level*



-*Table level*

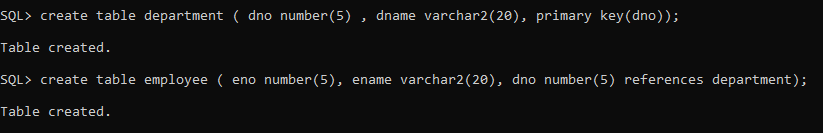


* **Foreign Key**

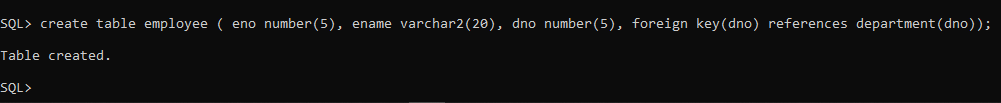
Columns whose value are derived from primary key

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

***-Column level:***

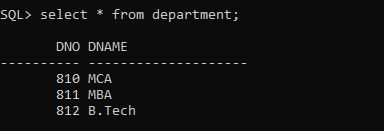


***Table Level***

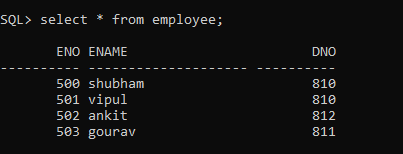


**OUTPUT:**

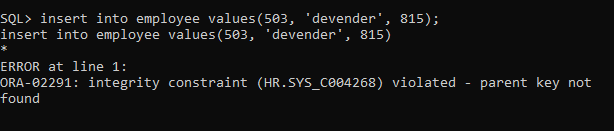
*Department table;*



*Employee table;*

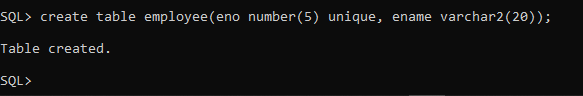


*Error on inserting inserting values out of dno range in employee table;*

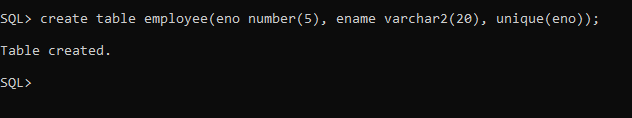


* **Unique Key**

*-Column level*

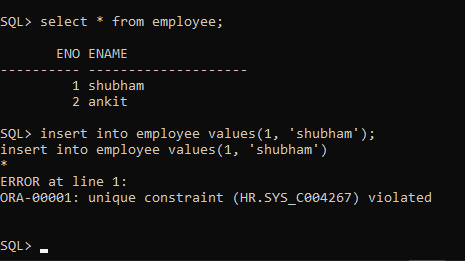


-*Table level*



**OUTPUT:**

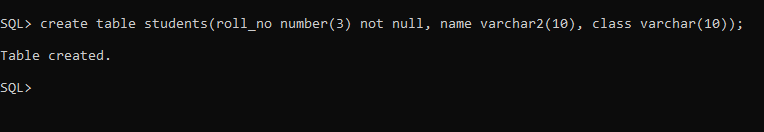
*Error on inserting duplicate ENO;*



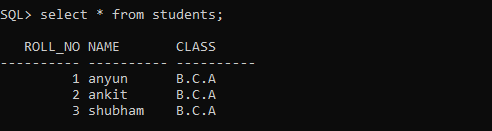
**Business Rule Constraint**

**Not Null**

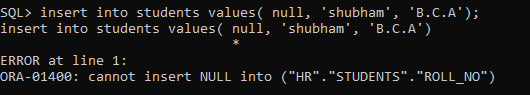
Only applied at column level



**OUTPUT**:

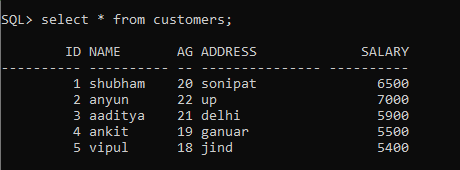


*Error on inserting null values in not null column*



**Views**

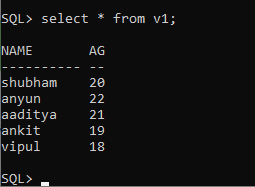
*Customers table:*



**Creating View:**

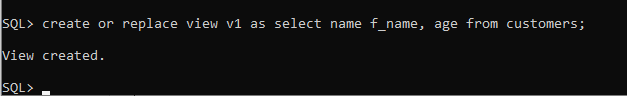


**OUTPUT:**

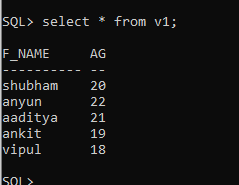


**Renaming Column of a view:**

*Renaming name column to f\_name for view:*



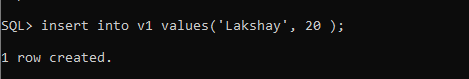
**OUTPUT:**



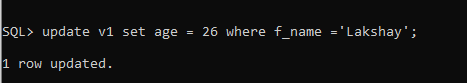
**Updating views:**

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

**Insert command**

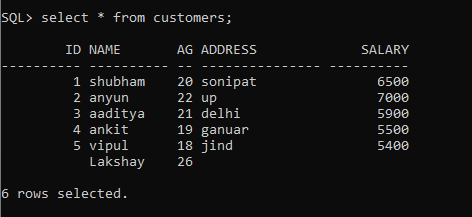


**Update command**

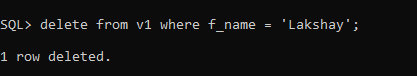


**OUTPUT:**

**Customers table:**

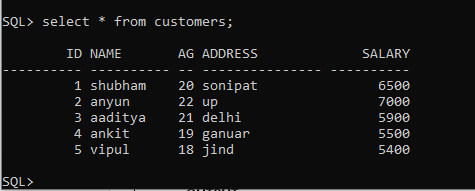


**Delete command:**



**OUTPUT:**

**Customers table:**



**Dropping view:**



**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

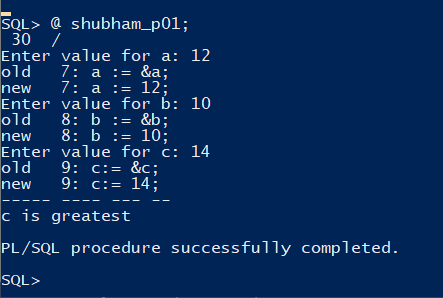
dbms\_output.put\_line('c is greatest');

end if;

end if;

end;

**OUTPUT:**



**Problem Statement:**

**Program to print 1 to 10 using while loop**

***Program:***

declare

i number;

begin

i := 1;

while i<=10

loop

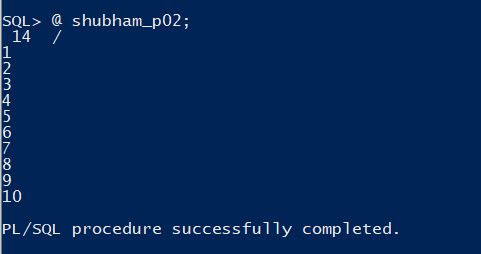
dbms\_output.put\_line(i);

i:= i+1;

end loop;

end;

**OUTPUT:**



**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;

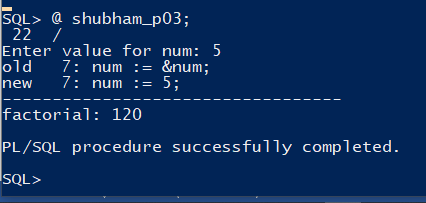
dbms\_output.put\_line('----------------------------------');

dbms\_output.put('factorial: ');

dbms\_output.put\_line(result);

end;

**OUTPUT:**



**Problem Statement:**

**Program to print 1 to 10 using for loop**

***Program:***

begin

for i in 1..10

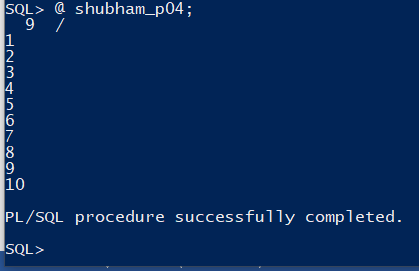
loop

dbms\_output.put\_line(i);

end loop;

end;

**OUTPUT:**



**Problem Statement:**

**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>>

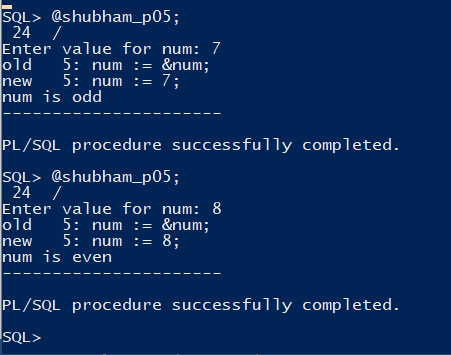
dbms\_output.put\_line('num is odd');

<<programend>>

dbms\_output.put\_line('----------------------');

end;

**OUTPUT:**



**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');

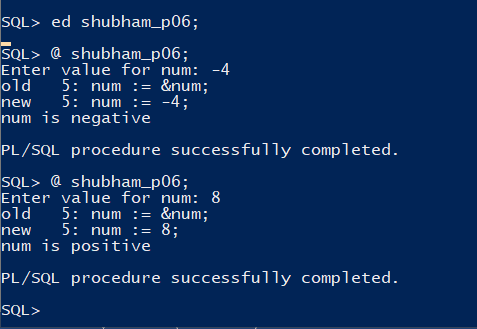
else

dbms\_output.put\_line( ' num is negative');

end if;

end;

**OUTPUT:**



**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

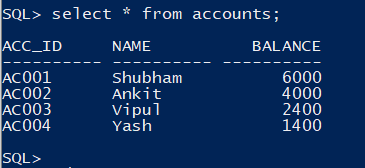
dbms\_output.put\_line('amount not debited. Account balance must be above Rs.2500');

end if;

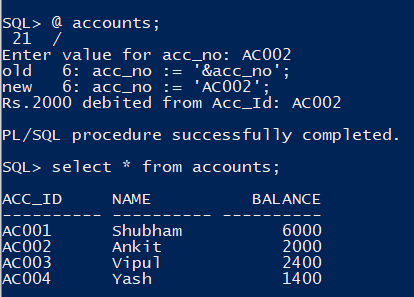
end;

**OUTPUT:**

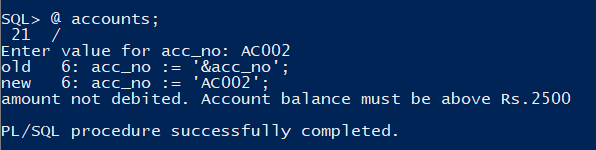
*Initially accounts table*



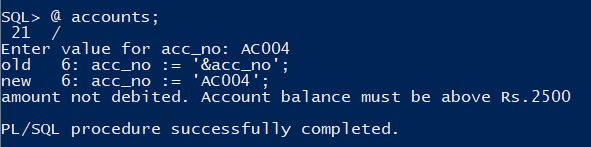
*After program execution for AC002*



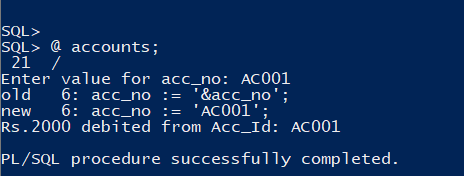
*After program execution for AC002 again*



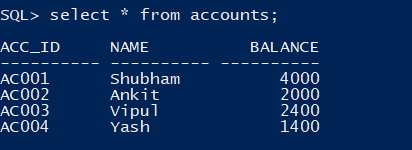
*After program execution for AC004*



*After program execution for AC001*



*Table Output:*



**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);

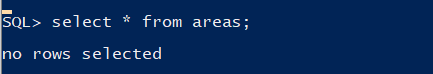
r:= r+1;

end loop;

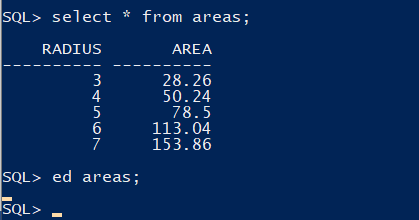
end;

**OUTPUT:**

*Before areas table;*



*After executing program*



**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;

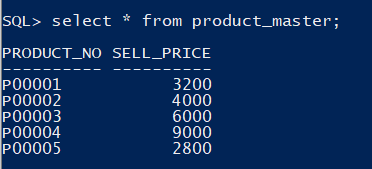
insert into old\_price\_table values(product\_noo, sysdate, old\_price);

dbms\_output.put\_line(' the price of product is 4000');

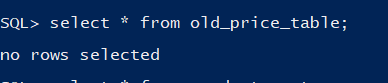
end;

**OUTPUT:**

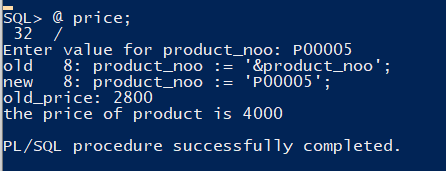
*Product\_master table:*

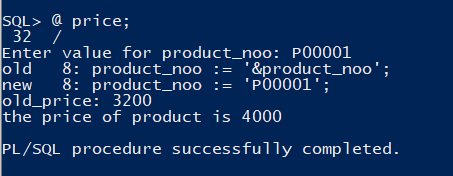


*Old\_price\_table:*

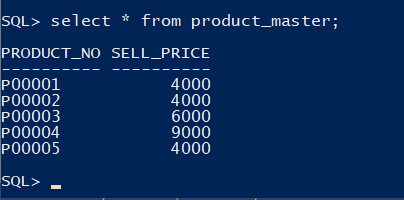


*Running program for ‘P00005’ & ‘P00001’*

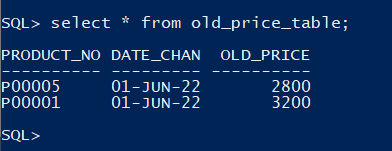




*Product\_master table:*



*Old\_price\_table*



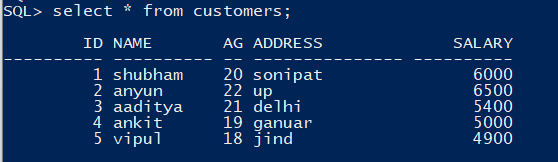
**Cursors**

Two types:

* Implicit Cursors
* Explicit Cursors

**Implicit Cursors:**

*Initially Customers table:*



***Program to increment salary by 500:***

begin

update customers set salary = salary + 500;

if sql%notfound then

dbms\_output.put\_line('no records updated');

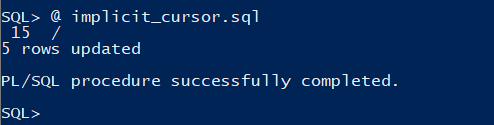
elsif sql%found then

dbms\_output.put\_line( sql%rowcount || ' rows updated');

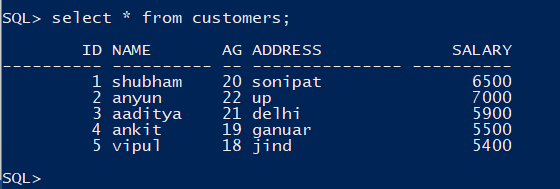
end if;

end;

**OUTPUT:**

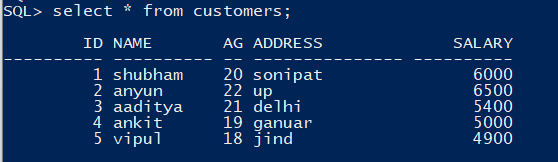


**Now customers table**



**Explicit Cursors:**

*Customers table*



*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;

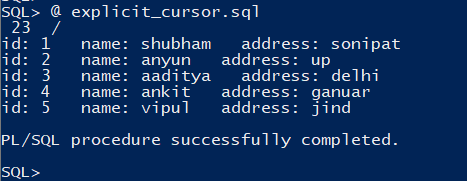
dbms\_output.put\_line('id: ' || c\_id ||' name: ' || c\_name || ' address: ' || c\_address );

end loop;

close mycursor;

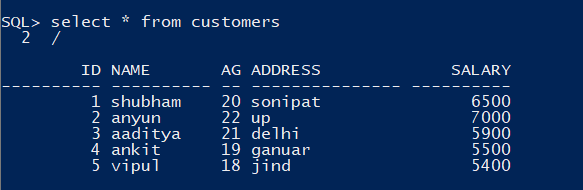
end;

**OUTPUT:**



**Triggers**

***Customers table:***



***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);

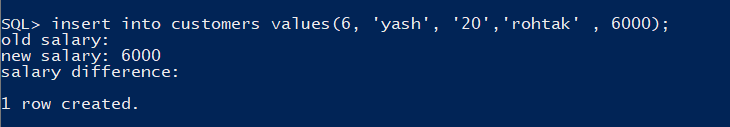
dbms\_output.put\_line('new salary: ' || :NEW.salary);

dbms\_output.put\_line('salary difference: ' || sal\_diff);

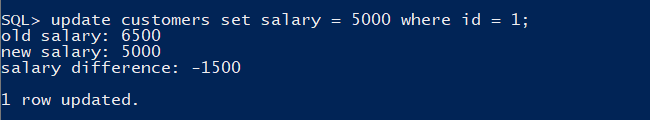
end;

**OUTPUT:**

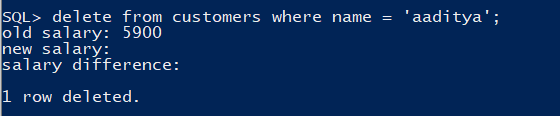
Inserting values



Updating values



Deleting values:



**Procedures & Functions**

**Standalone procecdures**

CREATE OR REPLACE PROCEDURE firstProcedure

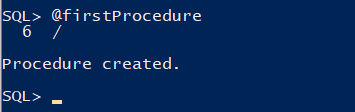
AS

BEGIN

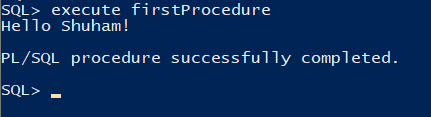
dbms\_output.put\_line('Hello Shuham!');

END;

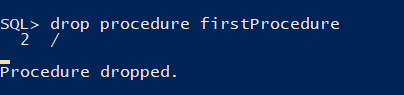
**OUTPUT:**



*Executing firstProcedure*



**Deleting Standalone Procedure**



**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;

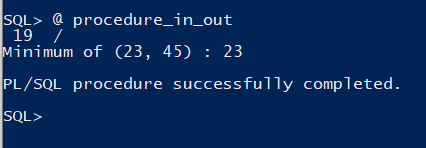
b:= 45;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);

END;

**OUTPUT:**



**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:**

