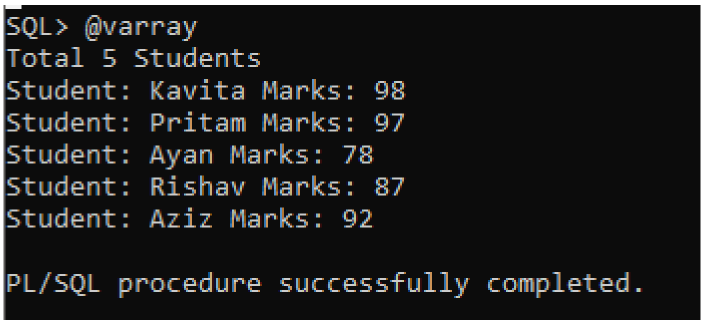
ANUBHAV SINGH
IC2K1654
Assignment 2

1. VARRAY

```
DECLARE
type namesarray
IS VARRAY(5) OF VARCHAR2(10);
type grades
IS VARRAY(5) OF INTEGER;
names namesarray;
marks grades;
total integer;
BEGIN
names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
marks:= grades(98, 97, 78, 87, 92);
total := names.count;
dbms_output.put_line('Total '|| total || ' Students');
 FOR i in 1 .. total
LOOP
dbms_output.put_line('Student: ' || names(i) || ' Marks: ' || marks(i));
END LOOP;
END;
/
```

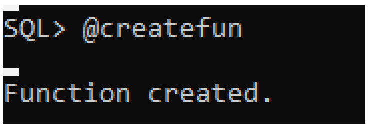


```
SQL> @varray
Total 5 Students
Student: Kavita Marks: 98
Student: Pritam Marks: 97
Student: Ayan Marks: 78
Student: Rishav Marks: 87
Student: Aziz Marks: 92

PL/SQL procedure successfully completed.
```

2. Create function

```
CREATE OR REPLACE FUNCTION totalCustomers
RETURN
 number IS total number(2) := 0;
BEGIN
SELECT count(*) into total FROM customers;
RETURN total;
END;
/
```



```
SQL> @createfun

Function created.
```

3. **Call function**

   DECLARE

   c number(2);

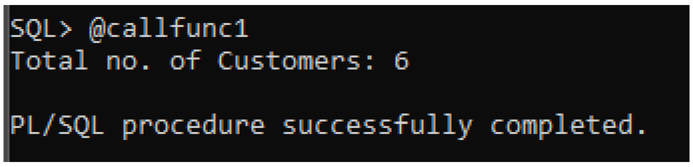
   BEGIN

   c := totalCustomers();

   dbms_output.put_line('Total no. of Customers: ' || c);

   END;

   /

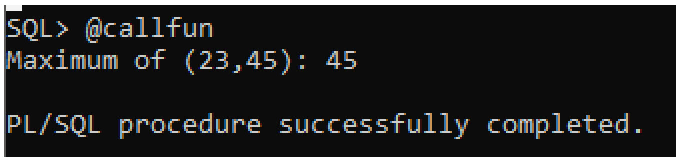


```
SQL> @callfunc1
Total no. of Customers: 6

PL/SQL procedure successfully completed.
```

4. **PLSQL FUNCTION**

```
DECLARE
a number;
b number;
c number;
FUNCTION findMax(x IN number, y IN number)
RETURN number IS z number;
BEGIN
IF x > y
THEN z:= x;
ELSE
Z:= y;
END IF;
RETURN z;
END;
BEGIN
a:= 23;
 b:= 45;
c := findMax(a, b);
dbms_output.put_line(' Maximum of (23,45): ' || c);
END;
/
```

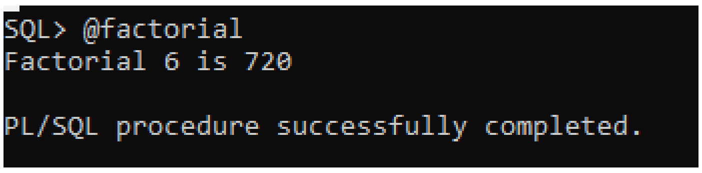


```
SQL> @callfun
Maximum of (23,45): 45

PL/SQL procedure successfully completed.
```

5. **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;
/
```

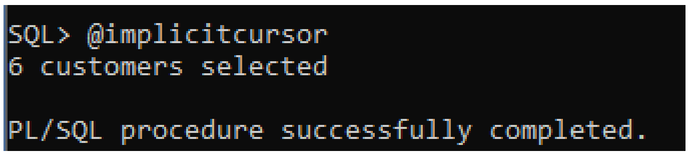


```
SQL> @factorial
Factorial 6 is 720

PL/SQL procedure successfully completed.
```

6. **Implicit cursor**

```
DECLARE
total_rows number(2);
 BEGIN
UPDATE customers SET salary = salary + 500;
IF sql%notfound
THEN
dbms_output.put_line('no customers selected');
 ELSIF sql%found
THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' customers selected ');
END IF;
END;
/
```

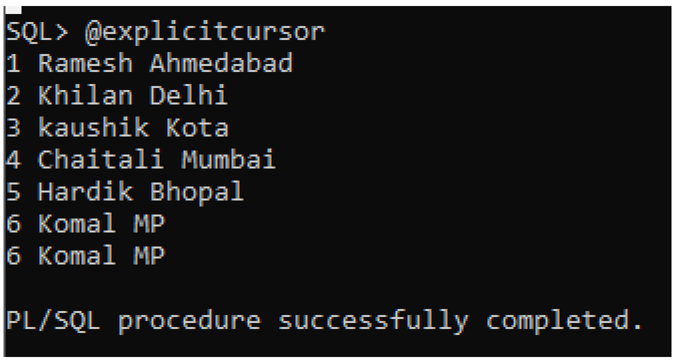


```
SQL> @implicitcursor
6 customers selected

PL/SQL procedure successfully completed.
```

7. **Explicit cursor**

```
DECLARE
c_id customers.id%type;
c_name customers.name%type;
c_addr customers.address%type;
CURSOR
c_customers is SELECT id, name,
address FROM customers;
BEGIN
OPEN c_customers;
LOOP
FETCH c_customers into c_id, c_name, c_addr;
dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
EXIT
WHEN c_customers%notfound;
END LOOP;
CLOSE c_customers;
END;
/
```

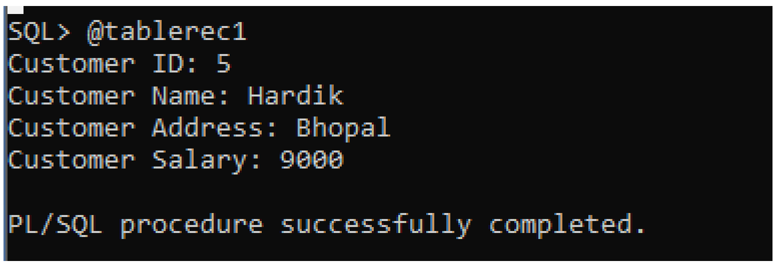


```
SQL> @explicitcursor
1 Ramesh Ahmedabad
2 Khilan Delhi
3 kaushik Kota
4 Chaitali Mumbai
5 Hardik Bhopal
6 Komal MP
6 Komal MP

PL/SQL procedure successfully completed.
```

8. **Table based records**

```
DECLARE
customer_rec customers%rowtype;
BEGIN
SELECT * into customer_rec FROM customers WHERE id = 5;
dbms_output.put_line('Customer ID: ' || customer_rec.id);
dbms_output.put_line('Customer Name: ' || customer_rec.name);
dbms_output.put_line('Customer Address: ' || customer_rec.address);
dbms_output.put_line('Customer Salary: ' || customer_rec.salary);
end;
/
```

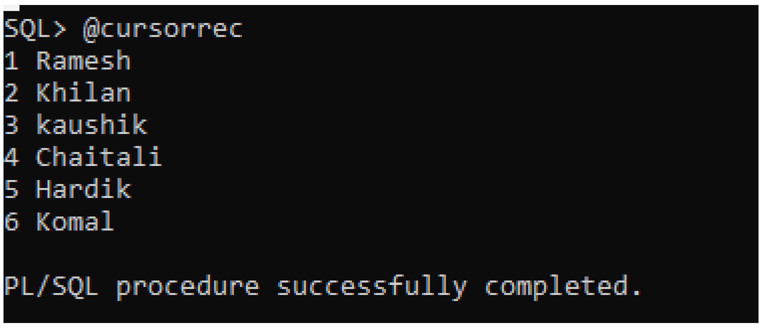


```
SQL> @tablerec1
Customer ID: 5
Customer Name: Hardik
Customer Address: Bhopal
Customer Salary: 9000

PL/SQL procedure successfully completed.
```

9. Cursor based records

```
DECLARE
CURSOR customer_cur is SELECT id,
name,
address FROM customers;
customer_rec customer_cur%rowtype;
BEGIN
 OPEN customer_cur;
LOOP FETCH customer_cur into customer_rec;
EXIT
WHEN customer_cur%notfound;
DBMS_OUTPUT.put_line(customer_rec.id || ' ' || customer_rec.name);
END LOOP;
END;
/
```

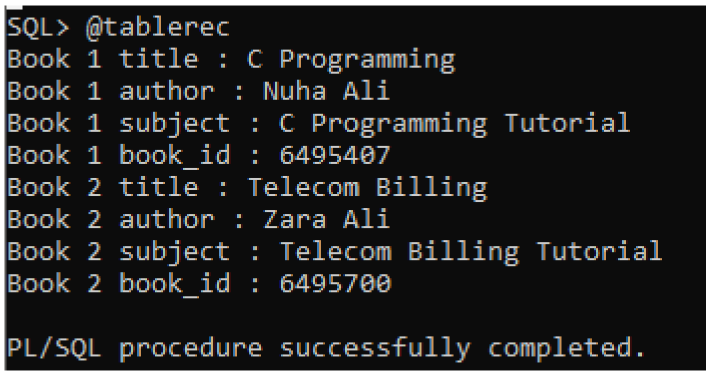


```
SQL> @cursorrec
1 Ramesh
2 Khilan
3 kaushik
4 Chaitali
5 Hardik
6 Komal

PL/SQL procedure successfully completed.
```

10. DEFINING A RECORD.

```
DECLARE
type books is record
(title varchar(50),
author varchar(50),
subject varchar(100),
book_id number);
book1 books;
book2 books;
BEGIN
book1.title := 'C Programming';
book1.author := 'Nuha Ali ';
book1.subject := 'C Programming Tutorial';
book1.book_id := 6495407;
book2.title := 'Telecom Billing';
book2.author := 'Zara Ali';
book2.subject := 'Telecom Billing Tutorial';
book2.book_id := 6495700;
dbms_output.put_line('Book 1 title : '|| book1.title);
dbms_output.put_line('Book 1 author : '|| book1.author);
dbms_output.put_line('Book 1 subject : '|| book1.subject);
dbms_output.put_line('Book 1 book_id : ' || book1.book_id);
dbms_output.put_line('Book 2 title : '|| book2.title);
dbms_output.put_line('Book 2 author : '|| book2.author);
dbms_output.put_line('Book 2 subject : '|| book2.subject);
dbms_output.put_line('Book 2 book_id : '|| book2.book_id);
END;
/
```



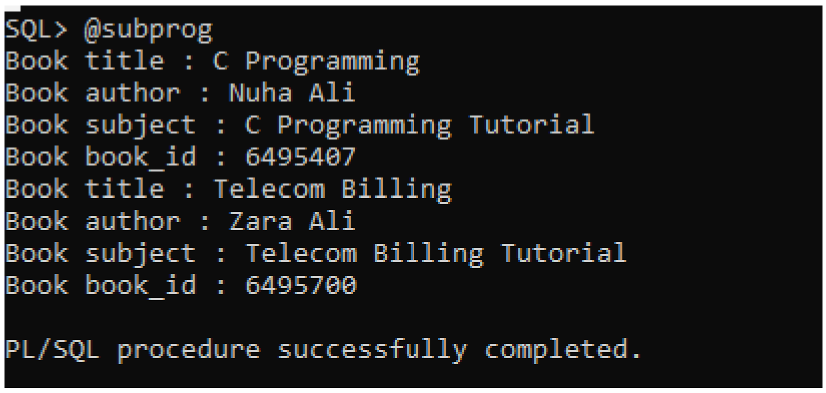
```
SQL> @tablerec
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700

PL/SQL procedure successfully completed.
```

11. **RECORD AS SUBPROGRAM PARAMETERS**

```
DECLARE
type books is record
(title varchar(50),
author varchar(50),
subject varchar(100),
book_id number);
book1 books;
book2 books;
PROCEDURE printbook (book books) IS
BEGIN
dbms_output.put_line ('Book title : ' || book.title);
dbms_output.put_line('Book author : ' || book.author);
dbms_output.put_line( 'Book subject : ' || book.subject);
dbms_output.put_line( 'Book book_id : ' || book.book_id);
END;
BEGIN
book1.title := 'C Programming';
book1.author := 'Nuha Ali ';
book1.subject := 'C Programming Tutorial';
book1.book_id := 6495407;
book2.title := 'Telecom Billing';
book2.author := 'Zara Ali';
book2.subject := 'Telecom Billing Tutorial';
book2.book_id := 6495700;
printbook(book1);
printbook(book2);
END;
```

/

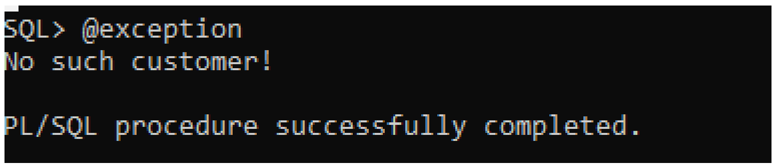


```
SQL> @subprog
Book title : C Programming
Book author : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Telecom Billing
Book author : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700

PL/SQL procedure successfully completed.
```

12. **Exception handling**

```
DECLARE
c_id customers.id%type := 8;
c_name customers.name%type;
c_addr customers.address%type;
BEGIN
SELECT name, address INTO c_name, c_addr FROM customers WHERE id = c_id;
DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
EXCEPTION WHEN no_data_found
THEN
dbms_output.put_line('No such customer!');
WHEN others
THEN
dbms_output.put_line('Error!');
END;
/
```



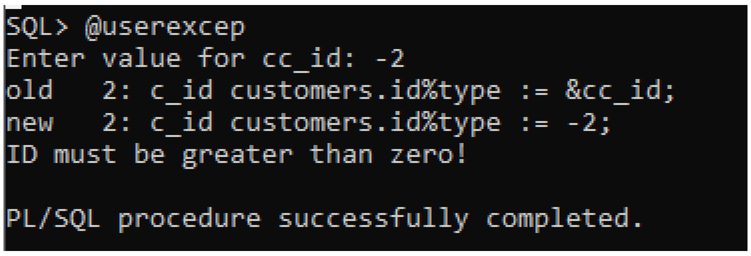
```
SQL> @exception
No such customer!

PL/SQL procedure successfully completed.
```

13. **User defined Exceeption**

```
DECLARE
c_id customers.id%type := &cc_id;
c_name customers.name%type;
c_addr customers.address%type;
ex_invalid_id EXCEPTION;
 BEGIN
IF c_id <= 0 THEN RAISE ex_invalid_id;
 ELSE
SELECT name, address INTO c_name, c_addr FROM customers WHERE id = c_id;
DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
```

```
END IF;
EXCEPTION
 WHEN ex_invalid_id
 THEN dbms_output.put_line('ID must be greater than zero!');
 WHEN no_data_found
 THEN dbms_output.put_line('No such customer!');
 WHEN others
THEN dbms_output.put_line('Error!');
END;
/
```

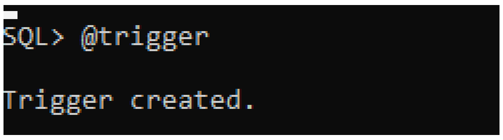


```
SQL> @userexcep
Enter value for cc_id: -2
old   2: c_id customers.id%type := &cc_id;
new   2: c_id customers.id%type := -2;
ID must be greater than zero!

PL/SQL procedure successfully completed.
```

14. **Creating Triggers**

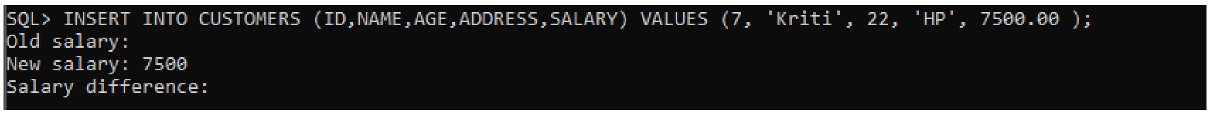
```
CREATE OR REPLACE TRIGGER
display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON
customers
FOR EACH ROW WHEN (NEW.ID > 0) DECLARE
sal_diff number;
BEGIN
sal_diff := :NEW.salary - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);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



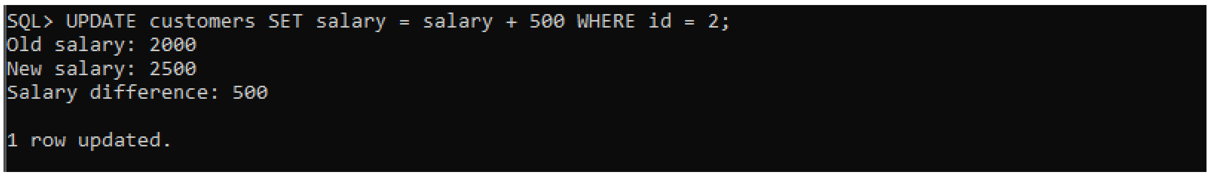
```
SQL> @trigger

Trigger created.
```

15. **Triggering a Trigger**



```
SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
Old salary:
New salary: 7500
Salary difference:
```



```
SQL> UPDATE customers SET salary = salary + 500 WHERE id = 2;
Old salary: 2000
New salary: 2500
Salary difference: 500

1 row updated.
```