

ANUBHAV SINGH

IC2K1654

Assignment 3

1. PACKAGE

```
CREATE PACKAGE
cust_sal AS PROCEDURE
find_sal(c_id customers.id%type);
END cust_sal;
/
```

```
SQL> @package;

Package created.
```

2. Package Body

```
CREATE OR REPLACE PACKAGE BODY
cust_sal AS PROCEDURE
find_sal(c_id customers.id%TYPE) IS c_sal customers.salary%TYPE;
BEGIN
SELECT salary INTO c_sal FROM customers WHERE id = c_id;
dbms_output.put_line('Salary: ' || c_sal);
END find_sal;
END cust_sal;
/
```

```
SQL> @packagebody;

Package body created.
```

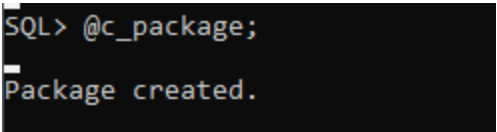
3. Using Package Elements

```
SQL> declare
2  code customers.id%type := &cc_id;
3  BEGIN
4  cust_sal.find_sal(code);
5  END;
6  /
Enter value for cc_id: 1
old 2: code customers.id%type := &cc_id;
new 2: code customers.id%type := 1;
Salary: 2500

PL/SQL procedure successfully completed.
```

4. Creating package

```
CREATE OR REPLACE PACKAGE c_package AS
PROCEDURE addCustomer(c_id customers.id%type,
c_name customers.name%type,
c_age customers.age%type,
c_addr customers.address%type,
c_sal customers.salary%type);
PROCEDURE delCustomer(c_id customers.id%TYPE);
PROCEDURE listCustomer;
END c_package;
/
```

A screenshot of a SQL command prompt window. The prompt shows the command '@c_package;' being entered, followed by a confirmation message 'Package created.' on the next line.

```
SQL> @c_package;
Package created.
```

5. Creating Package body;

```
CREATE OR REPLACE PACKAGE BODY c_package AS
PROCEDURE addCustomer(c_id customers.id%type,
c_name customers.name%type,
c_age customers.age%type,
c_addr customers.address%type,
c_sal customers.salary%type)
IS BEGIN
INSERT INTO customers (id,name,age,address,salary)
VALUES
(c_id, c_name, c_age, c_addr, c_sal);
END addCustomer;
PROCEDURE delCustomer(c_id customers.id%type)
IS BEGIN
DELETE FROM customers WHERE id = c_id;
END delCustomer;
PROCEDURE
listCustomer IS CURSOR c_customers IS SELECT name FROM customers;
TYPE c_list IS TABLE OF customers.name%type;
name_list c_list := c_list();
counter integer :=0;
BEGIN
FOR n IN c_customers LOOP counter := counter +1;
name_list.extend;
name_list(counter) := n.name;
dbms_output.put_line('Customer(' || counter || ') ' || name_list(counter));
END LOOP;
END listCustomer;
```

```
END c_package;
```

```
/
```

```
SQL> @c_packagebody;
```

```
Package body created.
```

6. Using Package

```
DECLARE
```

```
code customers.id%type:= 8;
```

```
BEGIN
```

```
c_package.addcustomer(7, 'Rajnish', 25, 'Chennai', 3500);
```

```
c_package.addcustomer(8, 'Subham', 32, 'Delhi', 7500);
```

```
c_package.listcustomer;
```

```
c_package.delcustomer(code);
```

```
c_package.listcustomer;
```

```
END;
```

```
/
```

```
SQL> @c_packageuse;
```

```
Old salary:
```

```
New salary: 3500
```

```
Salary difference:
```

```
Old salary:
```

```
New salary: 7500
```

```
Salary difference:
```

```
Customer(1)Ramesh
```

```
Customer(2)Khilan
```

```
Customer(3)kaushik
```

```
Customer(4)Chaitali
```

```
Customer(5)Hardik
```

```
Customer(6)Komal
```

```
Customer(7)Rajnish
```

```
Customer(8)Subham
```

```
Customer(1)Ramesh
```

```
Customer(2)Khilan
```

```
Customer(3)kaushik
```

```
Customer(4)Chaitali
```

```
Customer(5)Hardik
```

```
Customer(6)Komal
```

```
Customer(7)Rajnish
```

```
PL/SQL procedure successfully completed.
```

Collections

7. Index By table

```
DECLARE TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
salary_list salary;
name VARCHAR2(20);
BEGIN
salary_list('Rajnish') := 62000;
salary_list('Minakshi') := 75000;
salary_list('Martin') := 100000;
salary_list('James') := 78000;
name := salary_list.FIRST;
WHILE name IS NOT null LOOP
dbms_output.put_line ('Salary of ' || name || ' is ' || TO_CHAR(salary_list(name)));
name := salary_list.NEXT(name);
END LOOP;
END;
```

```
/
SQL> @indexbytable;
Salary of James is 78000
Salary of Martin is 100000
Salary of Minakshi is 75000
Salary of Rajnish is 62000

PL/SQL procedure successfully completed.
```

8. Elements of Index By Table

```
DECLARE CURSOR
c_customers is select name from customers;
TYPE c_list IS TABLE of customers.name%type INDEX BY binary_integer;
name_list c_list;
counter integer :=0;
BEGIN
FOR n IN c_customers LOOP counter := counter +1;
name_list(counter) := n.name;
dbms_output.put_line('Customer(' || counter || '):' || name_list(counter));
END LOOP;
END;
```

```
/
SQL> @eleindexbytable;
Customer(1):Ramesh
Customer(2):Khilan
Customer(3):kaushik
Customer(4):Chaitali
Customer(5):Hardik
Customer(6):Komal

PL/SQL procedure successfully completed.
```

9. Nested Tables

```
DECLARE TYPE names_table IS TABLE OF VARCHAR2(10);
TYPE grades IS TABLE OF INTEGER;
names names_table;
marks grades;
total integer;
BEGIN
names := names_table('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> @nestedtable;
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.
```

10.Elements of Nested Tables

```
DECLARE
CURSOR c_customers is
SELECT name FROM customers;
TYPE c_list IS TABLE of customers.name%type;
name_list c_list := c_list();
counter integer :=0;
BEGIN
FOR n IN c_customers
LOOP
counter := counter +1;
name_list.extend;
name_list(counter) := n.name;
dbms_output.put_line('Customer(' || counter || '):' || name_list(counter));
END LOOP;
END;
/
```

```
SQL> @elenested;
Customer(1):Ramesh
Customer(2):Khilan
Customer(3):kaushik
Customer(4):Chaitali
Customer(5):Hardik
Customer(6):Komal

PL/SQL procedure successfully completed.
```

Commit

11.Commit a Table

```
SQL> CREATE TABLE CUSTOMERS( ID INT NOT NULL, NAME VARCHAR (20) NOT NULL, AGE INT NOT NULL, ADDRESS CHAR (25), SALARY DECIMAL (18, 2), PRIMARY KEY (ID) );
Table created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'Komal', 22, 'MP', 4500.00 );
1 row created.

SQL> COMMIT;
```

12.Rollback and Savepoints

```
SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'Rajnish', 27, 'HP', 9500.00 );
1 row created.

SQL> INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (8, 'Riddhi', 21, 'WB', 4500.00 );
1 row created.

SQL> SAVEPOINT sav1;
Savepoint created.

SQL> UPDATE CUSTOMERS SET SALARY = SALARY + 1000;
8 rows updated.

SQL> ROLLBACK TO sav1;
Rollback complete.

SQL> UPDATE CUSTOMERS SET SALARY = SALARY + 1000 WHERE ID = 7;
1 row updated.

SQL> UPDATE CUSTOMERS SET SALARY = SALARY + 1000 WHERE ID = 8;
1 row updated.

SQL> COMMIT;
Commit complete.
```