

Assignment 5 :--PL/SQL(Basic commands)

/*basic eg of print,fro loop,while loop

SQL> set serveroutput on

SQL> declare

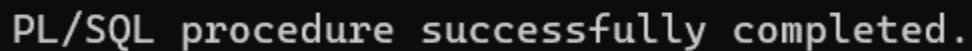
2 text varchar(20):='hii';

3 begin

4 dbms_output.put_line(text);

5 end;

6 /

hiiPL/SQL procedure successfully completed.

SQL> declare

2 a number(2);

3 begin

4 for a in 0..10 loop

5 dbms_output.put_line('value of a: '||a);

6 end loop;

7 end;

8 /

```
value of a: 0  
value of a: 1  
value of a: 2  
value of a: 3  
value of a: 4  
value of a: 5  
value of a: 6  
value of a: 7  
value of a: 8  
value of a: 9  
value of a: 10
```

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

SQL> declare

2 a int(2);

3 b int(2);

4 begin

5 a:=0;

6 b:=&b;

7 while a<=b loop

8 a:=a+2;

9 dbms_output.put_line('value of a: '||a);

10 end loop;

11 end;

12 /

```

12 /
Enter value for b: 20
old 6: b:=&b;
new 6: b:=20;
value of a: 2
value of a: 4
value of a: 6
value of a: 8
value of a: 10
value of a: 12
value of a: 14
value of a: 16
value of a: 18
value of a: 20
value of a: 22

PL/SQL procedure successfully completed.

```

*/

Exempl1 1—print a message

SQL> set serveroutput on

SQL> DECLARE

2 message varchar2(20):= 'Hello, World!';

3 BEGIN

4 dbms_output.put_line(message);

5 END;

6 /

```

Hello, World!

PL/SQL procedure successfully completed.

```

Example 2—create a procedure & execute it

```
SQL> CREATE OR REPLACE PROCEDURE greetings
```

```
2 AS
```

```
3 BEGIN
```

```
4   dbms_output.put_line('Hello World!');
```

```
5 END;
```

```
6 /
```

```
Procedure created.
```

```
SQL> execute greetings;
```

```
Hello World!
```

Example 3—create a function & execute it

```
//create a table named customers
```

```
SQL> create table customers(
```

```
2 id int primary key,
```

```
3 name varchar(20),
```

```
4 age int,
```

```
5 address char(50),
```

```
6 salary decimal(18,2));
```

```
Table created.
```

```
//show the table
```

```
SQL> desc customers;
```

Name	Null?	Type
ID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)
ADDRESS		CHAR(50)
SALARY		NUMBER(18,2)

```
//insert datas
```

```
SQL> insert into customers values(1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
```

```
SQL> insert into customers values(2, 'Khilan', 25, 'Delhi', 1500.00 );
```

```
SQL> insert into customers values(3, 'kaushik', 23, 'Kota', 2000.00 );
```

```
SQL> insert into customers values(4, 'Chaitali', 25, 'Mumbai', 6500.00 );
```

```
SQL> insert into customers values(5, 'Hardik', 27, 'Bhopal', 8500.00 );
```

```
SQL> insert into customers values(6, 'Komal', 22, 'MP', 4500.00 );
```

```
//print the table
```

```
SQL> select * from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000
2	Khilan	25	Delhi	1500
3	kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	MP	4500

6 rows selected.

//creating the func

SQL> CREATE OR REPLACE FUNCTION totalCustomers

```
2 RETURN number IS
3   total number(2) := 0;
4 BEGIN
5   SELECT count(*) into total
6   FROM customers;
7
8   RETURN total;
9 END;
10 /
```

Function created.

//calling the func & executing necessary task

SQL> DECLARE

```
2   c number(2);
3 BEGIN
4   c := totalCustomers();
5   dbms_output.put_line('Total no. of Customers: ' || c);
6 END;
7 /
```

Total no. of Customers: 6

PL/SQL procedure successfully completed.

Example—4:create a PL/SQL stored procedure named **print_contact &** print out contact's information

//create table

SQL> CREATE TABLE contacts (

```

2  customer_id NUMBER PRIMARY KEY,
3  first_name VARCHAR2(50),
4  last_name VARCHAR2(50),
5  email VARCHAR2(100) UNIQUE,
6  phone_number VARCHAR2(20),
7  address VARCHAR2(200),
8  birth_date DATE
9 );

```

Table created.

```
//show table
```

```
SQL> desc contacts;
```

Name	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER
FIRST_NAME		VARCHAR2(50)
LAST_NAME		VARCHAR2(50)
EMAIL		VARCHAR2(100)
PHONE_NUMBER		VARCHAR2(20)
ADDRESS		VARCHAR2(200)
BIRTH_DATE		DATE

```
//insert datas into table
```

```
SQL> insert into contacts values(1, 'John', 'Doe', 'john.doe@email.com', '555-1234', '123 Main St, Cityville', TO_DATE('1990-05-15', 'YYYY-MM-DD'));
```

```
SQL> insert into contacts values(2, 'Jane', 'Smith', 'jane.smith@email.com', '555-5678', '456 Oak St, Townsville', TO_DATE('1985-08-22', 'YYYY-MM-DD'));
```

```
SQL> insert into contacts values(3, 'Bob', 'Johnson', 'bob.johnson@email.com', '555-9876', '789 Pine St, Villagetown', TO_DATE('1978-03-10', 'YYYY-MM-DD'));
```

```
SQL> insert into contacts values(4, 'Alice', 'Williams', 'alice.williams@email.com', '555-4321', '987 Cedar St, Hamletville', TO_DATE('1995-11-28', 'YYYY-MM-DD'));
```

```
//print the table
```

```
SQL> select * from contacts;
```

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

1 John

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

Doe

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

Doe

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

john.doe@email.com

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

555-1234

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

123 Main St, Cityville

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

15-MAY-90

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

2 Jane

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

Smith

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

jane.smith@email.com

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

555-5678

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

456 Oak St, Townsville

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

22-AUG-85

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

3 Bob

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

```
ADDRESS
-----
BIRTH_DAT
-----
Johnson

CUSTOMER_ID FIRST_NAME
-----
LAST_NAME
-----
EMAIL
-----
PHONE_NUMBER
-----
ADDRESS
-----
BIRTH_DAT
-----
bob.johnson@email.com

CUSTOMER_ID FIRST_NAME
-----
LAST_NAME
-----
EMAIL
-----
PHONE_NUMBER
-----
ADDRESS
-----
BIRTH_DAT
-----
555-9876

CUSTOMER_ID FIRST_NAME
-----
LAST_NAME
-----
```

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

789 Pine St, Villagetown

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

10-MAR-78

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

4 Alice

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

Williams

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

alice.williams@email.com

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

555-4321

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

BIRTH_DAT

987 Cedar St, Hamletville

CUSTOMER_ID FIRST_NAME

LAST_NAME

EMAIL

PHONE_NUMBER

ADDRESS

```

BIRTH_DAT
-----
987 Cedar St, Hamletville

CUSTOMER_ID FIRST_NAME
-----
LAST_NAME
-----
EMAIL
-----
PHONE_NUMBER
-----
ADDRESS
-----
BIRTH_DAT
-----
28-NOV-95

CUSTOMER_ID FIRST_NAME
-----
LAST_NAME
-----
EMAIL
-----
PHONE_NUMBER
-----
ADDRESS
-----
BIRTH_DAT
-----

```

//create the procedure "print_contact" & print contact infos

SQL> CREATE OR REPLACE PROCEDURE print_contact(in_customer_id NUMBER) IS

```

2  r_contact contacts%ROWTYPE;
3  BEGIN
4  SELECT *
5  INTO r_contact
6  FROM contacts
7  WHERE customer_id = in_customer_id; -- Fix the parameter name here
8
9  dbms_output.put_line(r_contact.first_name || ' ' || r_contact.last_name || '<' ||
r_contact.email || '>');
10 EXCEPTION
11  WHEN OTHERS THEN
12    dbms_output.put_line(SQLERRM);
13 END;

```

14 /

```
Procedure created.
```

```
//execute procedure
```

```
SQL> EXECUTE print_contact(1);
```

```
John Doe<john.doe@email.com>
```

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

```
SQL> EXECUTE print_contact(2);
```

```
Jane Smith<jane.smith@email.com>
```

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

```
SQL> EXECUTE print_contact(3);
```

```
Bob Johnson<bob.johnson@email.com>
```

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

```
SQL> EXECUTE print_contact(4);
```

```
Alice Williams<alice.williams@email.com>
```

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

Example 5—cursore

```
//create a table named t
```

```
SQL> CREATE TABLE t (
```

```
2 id NUMBER,
```

```
3 name VARCHAR2(50),
```

```
4 date_col DATE
```

```
5 );
```

```
Table created.
```

```
//show the table
```

```
SQL> desc t;
```

Name	Null?	Type
ID		NUMBER
NAME		VARCHAR2(50)
DATE_COL		DATE

```
//insert dates
```

```
SQL> insert into t values(1, 'John Doe', TO_DATE('2023-01-15', 'YYYY-MM-DD'));
```

```
SQL> insert into t values(2, 'Jane Smith', NULL);
```

```
SQL> insert into t values(3, 'Bob Johnson', TO_DATE('2023-03-25', 'YYYY-MM-DD'));
```

```
SQL> insert into t values(4, 'Alice Brown', NULL);
```

```
//print the table
```

```
SQL> select * from t;
```

ID	NAME	DATE_COL
1	John Doe	15-JAN-23
2	Jane Smith	
3	Bob Johnson	25-MAR-23
4	Alice Brown	

```
//cursore
```

```
SQL> DECLARE
```

```
2  name varchar2(20);
```

```
3  Cursor c1 is
```

```
4  select t.name
```

```
5  from t t
```

```
6  where t.date_col is not null;
```

```
7 BEGIN
```

```
8  OPEN c1;
```

```
9  LOOP
10  FETCH c1 into name;
11  dbms_output.put_line(name);
12  exit when c1%NOTFOUND;
13  END LOOP;
14  CLOSE c1;
15  END;
16 /
```

```
John Doe
Bob Johnson
Bob Johnson

PL/SQL procedure successfully completed.
```

Example 6—trigger(statement trigger)

//create the table

```
CREATE TABLE Personnel (
    employee_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    job_title VARCHAR2(50),
    salary NUMBER
);
```

```
Table created.
```

//show the table

```
desc Personnel;
```

TABLE PERSONNEL

Column	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER
FIRST_NAME	-	VARCHAR2(50)
LAST_NAME	-	VARCHAR2(50)
JOB_TITLE	-	VARCHAR2(50)
SALARY	-	NUMBER

```
//insert datas
```

```
INSERT INTO Personnel VALUES (1, 'John', 'Doe', 'Manager', 50000);
```

```
INSERT INTO Personnel VALUES (2, 'Jane', 'Smith', 'Developer', 40000);
```

```
INSERT INTO Personnel VALUES (3, 'Bob', 'Johnson', 'Analyst', 35000);
```

```
INSERT INTO Personnel VALUES (4, 'Alice', 'Williams', 'Designer', 45000);
```

```
//print the table
```

```
Select * from personnel;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	JOB_TITLE	SALARY
1	John	Doe	Manager	50000
2	Jane	Smith	Developer	40000
3	Bob	Johnson	Analyst	35000
4	Alice	williams	Designer	45000

```
//create backup table test_t
```

```
CREATE TABLE test_t (
```

```
    info VARCHAR2(50),
```

```
    date_col date
```

```
);
```

Table created.

```
//create trigger trig_test_t
CREATE OR REPLACE TRIGGER trig_test_t
AFTER INSERT or UPDATE ON Personnel
BEGIN
If Inserting
Then INSERT into test_t values ('insert done', SYSDATE) ;
Else
INSERT into test_t values ('update done', SYSDATE) ;
End If;
END;
/
```

Trigger created.

```
//checking the trigger fired or not
INSERT INTO Personnel VALUES (5, 'Joh', 'Det', 'Engineer ', 80000);
```

1 row(s) inserted.

```
select * from test_t;
```

INFO	DATE_COL
insert done	14-NOV-23

/*

```
//eg 2[actual eg]:
```

```
//create the table
```

```
CREATE TABLE personnel_2(  
    emp_id NUMBER PRIMARY KEY,  
    name VARCHAR2(50),  
    salary NUMBER  
);
```

Table created.

```
//show the table
```

```
desc Personnel_2;
```

TABLE PERSONNEL_2

Column	Null?	Type
EMP_ID	NOT NULL	NUMBER
NAME	-	VARCHAR2(50)
SALARY	-	NUMBER

```
//insert datas
```

```
insert into personnel_2 values(1,'joe',70000);  
insert into personnel_2 values(2,'harry',75000);  
insert into personnel_2 values(3,'sun',70000);  
insert into personnel_2 values(4,'lou',60000);
```



```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
//print the table
```

```
select * from personnel_2;
```

EMP_ID	NAME	SALARY
1	joe	70000
2	harry	75000
3	sun	70000
4	lou	60000

```
//write the statement trigger
```

```
create or replace trigger trig
```

```
after delete on Personnel_2
```

```
begin
```

```
dbms_output.put_line('1 row deleted by you');
```

```
end;
```

```
/
```

```
Trigger created.
```

```
//fire the trigger trig (no need to fire the trigger it will automatically happen if the mentioned event occurs)
```

```
delete from Personnel_2 where salary=70000;
```

```
1 row deleted by you
```

```
2 rows deleted.
```

EMP_ID	NAME	SALARY
2	harry	75000
4	lou	60000

--two rows are deleted here but trigger fired for only once & will show the same record that 1 row has been deleted by you instead of '1 row has been deleted by you, 1 row has been deleted by you'.

```
*/
```

Example 7—trigger(row trigger)

```
//create table
```

```
CREATE TABLE personnel_3(  
    emp_id NUMBER PRIMARY KEY,  
    name VARCHAR2(50),  
    salary NUMBER  
);
```

```
Table created.
```

```
//show the table
```

```
desc Personnel_3;
```

TABLE PERSONNEL_3		
Column	Null?	Type
EMP_ID	NOT NULL	NUMBER
NAME	-	VARCHAR2(50)
SALARY	-	NUMBER

```
//insert datas
```

```
insert into personnel_3 values(1,'Tom',30000);
```

```
insert into personnel_3 values(2,'Richal',40000);
```

```
insert into personnel_3 values(3,'Hiran',30000);
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
//show the table
```

```
select * from personnel_3;
```

EMP_ID	NAME	SALARY
1	Tom	30000
2	Richal	40000
3	Hiran	30000

```
//write the row trigger
```

```
create or replace trigger trig
```

```
after delete on Personnel_2
```

```
for each row
```

```

begin
    dbms_output.put_line('1 row deleted by you');
end;
/

```

Trigger created.

//fire the trigger trig (no need to fire the trigger it will automatically happen if the mentioned event occurs)

```
delete from Personnel_3 where salary=30000;
```

```

1 row deleted by you
1 row deleted by you

2 rows deleted.

```

EMP_ID	NAME	SALARY
2	Richal	40000

/* --two rows are deleted here & trigger fired for each rows & will show the same record that 2 row has been deleted by you in the way ->'1 row has been deleted by you, 1 row has been deleted by you'.*/

Example—8:referential integrity trigger

```
//create table named author
```

```
create table author(
```

```
    aID number,
```

```
    book varchar2(50),
```

```
    pg number
```

```
);
```

```
Table created.
```

```
//show table
```

```
desc author;
```

```
TABLE AUTHOR
```

Column	Null?	Type
AID	-	NUMBER
BOOK	-	VARCHAR2(50)
PG	-	NUMBER

```
//insert datas
```

```
insert into author values(11,'aaa',123);
```

```
insert into author values(15,'bbb',458);
```

```
insert into author values(21,'eee',900);
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
//print table
```

```
select * from author;
```

AID	BOOK	PG
11	aaa	123
15	bbb	458
21	eee	900

```
//create table named Book
```

```
create table Book(
  authID number,
  publihed_b varchar2(50)
);
```

```
Table created.
```

```
//show table
```

```
desc Book;
```

TABLE BOOK		
Column	Null?	Type
AUTHID	-	NUMBER
PUBLIHED_B	-	VARCHAR2(50)

```
//insert datas
```

```
insert into Book values(11,'aaa');
```

```
insert into Book values(14,'ggg');
```

```
insert into Book values(22,'fff');
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
//print table
```

```
select * from Book;
```

AUTHID	PUBLISHED_B
11	aaa
14	ggg
22	fff

```
//write Referential integrity trigger
```

```
CREATE OR REPLACE TRIGGER author_trig
```

```
AFTER UPDATE OF aID ON author
```

```
FOR EACH ROW
```

```
BEGIN
```

```
UPDATE Book SET authID = :new.aID WHERE authID= :old.aID;
```

```
dbms_output.put_line('old id:=' || :old.aID || 'new id:=' || :new.aID);
```

```
END;
```

```
/
```

```
Trigger created.
```

```
//fire the trigger
```

```
UPDATE author SET aID='9' WHERE aID='11';
```

```
1 row(s) updated.  
old id:=11new id:=9
```

```
//after trigged
```

AUTHID	PUBLIHED_B
9	aaa
14	ggg
22	fff

AID	BOOK	PG
9	aaa	123
15	bbb	458
21	eee	900

//Example –9 trigger: business rules

```
//create another table named book_2 (with column publisher)
```

```
create table book_2(
```

```
    authID number,
```

```
    publihed_b varchar2(50),
```

```
    publisher varchar2(30)
```

```
);
```

```
//show table
```

```
desc book_2;
```

TABLE BOOK_2		
Column	Null?	Type
AUTHID	-	NUMBER
PUBLIHED_B	-	VARCHAR2(50)
PUBLISHER	-	VARCHAR2(30)

```
//insert datas
```



```
insert into book_2 values(11,'aaa','A');
```

```
insert into book_2 values(14,'ggg','B');
```

```
insert into book_2 values(22,'fff','A');
```

```
//print table
```

```
select * from book_2;
```

AUTHID	PUBLISHED_B	PUBLISHER
11	aaa	A
14	ggg	B
22	fff	A

```
//write trigger
```

```
CREATE OR REPLACE TRIGGER publish_trig
```

```
BEFORE INSERT OR UPDATE ON book_2
```

```
FOR EACH ROW
```

```
DECLARE
```

```
how_many NUMBER;
```

```
BEGIN
```

```
SELECT COUNT(*) INTO how_many FROM book_2
```

```
WHERE publisher = :new.publisher;
```

```
IF how_many >= 3 then
```

```
Raise_application_error(-20000,'Publisher' ||
```

```
:new.publisher || 'already has 3 books');
```

```
END IF;
```

```
END;
```

```
/
```

```
Trigger created.
```

```
//fire the trigger
```

(trigger not fired yet as how_many==2,before insert)

```
insert into book_2 values(24,'sss','A');
```

```
1 row(s) inserted.
```

(trigger fired as how_many==3,same thing will happen if how_many>3)

```
insert into book_2 values(12,'ttt','A');
```

```
ORA-20000: PublisherAalready has 3 books ORA-06512: at "SQL_CULHWYDATGPTDMRHATWJEONFP.PUBLISH_TRIG", line 7  
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

More Details: <https://docs.oracle.com/error-help/db/ora-20000>

