**PROCEDURES QUIZ**

**Problem #1:**

This code is set up to handle the EXCEPTION if user enter the dept no. which does not exist.

1. Declaring variables
2. Promting user to enter dept no.
3. Begin selecting records from dept table.
4. WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Invalid Department ' || TO\_CHAR(v\_deptno)). This will handle the case when no record is found.

When user entered 20 the record with DEPTNO 20 is selected.

When DEPTNO 24 is entered exception will be displayed as there is no record with deptno. 24.

**Problem #2:**

Assuming sal was 3000 before update.

In the first update when salary was set to 2000 the program will throw an error **Salary reduction** as new sal is less than old.

In the second update when salary is set to 4000 it will be completed without any error as new salary is greater than old.

**Problem #3**

**Quiz\_donation table**

**Output from the code**

IDNO DRIVENO CONDATE CONTAMT

11111 100 12-01-14 25

12121 200 12-01-14 40

23456 100 12-01-14 20

33333 300 12-01-14 10

22222 100 12-01-14 10

12121 100 12-01-14 50

23456 300 12-01-14 10

11111 200 12-01-14 35

1. The program declares a series of variables in DECLARE.
2. A donor\_cursor is set up to take all records from table copy\_donation.
3. A function calcNewDon is set with variables f\_code, f\_contamt . They get the values from variables v\_code and v\_contamt(v\_code and v\_contamt get values from table copy\_donation). Return variable f\_newcontamt is also defined which gives value to v\_newcontamt. In this code no contamt satisfies the first three conditions so v-code is always ‘DD’. So no change in contamt.
4. Procedure AddDonProc is also defined which is to insert values in table quiz\_donation. Variables P\_idno, p\_driveno and p\_contamt are defined in this procedure to insert records in the table.
5. At the end of the code procedure AddDonProc is called and variables v\_idno, v\_driveno and v\_contamt are passed to give values to the variables P\_idno, p\_driveno and p\_contamt and finally to the table.

**Problem #4:**

**EXCEPTIONS:**

set verify off

set serveroutput on

DECLARE

**new\_bonus\_over8000 EXCEPTION;**

e\_id emp\_bk2.emp\_id%type := '&input\_id';

e\_name emp\_bk2.name%type;

e\_title emp\_bk2.title%type;

e\_bonus emp\_bk2.bonus%type;

e\_salary emp\_bk2.salary%type;

BEGIN

select emp\_id, name, title, bonus, salary into

e\_id, e\_name, e\_title, e\_bonus, e\_salary

from emp\_bk2

where emp\_id = e\_id;

IF e\_title = 'Senior Manager' THEN

e\_bonus := e\_bonus + 500;

ELSIF e\_title = 'CEO' THEN

e\_bonus := e\_bonus + 200;

ELSIF e\_title = 'Salesperson' THEN

e\_bonus := e\_bonus + 150;

ELSE

e\_bonus := e\_bonus;

END IF;

IF e\_bonus > 8000 then

RAISE new\_bonus\_over8000;

END IF;

update emp\_bk2

set bonus = e\_bonus

where emp\_id = e\_id;

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

dbms\_output.put\_line('Not a valid id #');

**WHEN new\_bonus\_over8000 THEN**

dbms\_output.put\_line('Bonus exceeds 8000');

end;

/

set serveroutput off

set verify on

SQL> @ exep\_prob

Enter value for input\_id: 07

**Not a valid id #**

SQL> @ exep\_prob

Enter value for input\_id: 04

**Bonus exceeds 8000**

**PROBLEM #5:**

Creating procedure

CREATE OR REPLACE PROCEDURE **per\_week\_sal**

(

P\_IDNO weekly\_salary.idno%type,

P\_NAME weekly\_salary.name%type,

P\_WEEK\_SAL weekly\_salary.week\_sal%type)AS

BEGIN

insert into weekly\_salary(idno, name, week\_sal)

values(p\_idno, p\_name, p\_week\_sal);

END;

/

SQL> @ procedure1

Procedure created.

set serveroutput on

DECLARE

e\_name paydata1.name%type;

e\_id paydata1.IDNO%type;

e\_salary paydata1.salary%type;

salary\_per\_week paydata1.salary%type;

job\_code paydata1.jobcode%type;

cursor paydata\_cursor is

select name, idno, salary,jobcode from paydata1

where JOBCODE = 'S'

order by idno;

BEGIN

open paydata\_cursor;

fetch paydata\_cursor into e\_name, e\_id, e\_salary, job\_code;

while paydata\_cursor%FOUND LOOP

salary\_per\_week := e\_salary/52;

**per\_week\_sal** (e\_id, e\_name, salary\_per\_week);

fetch paydata\_cursor into e\_name, e\_id, e\_salary, job\_code;

end loop;

close paydata\_cursor;

end;

/

set serveroutput off

SQL> @ procedure1

PL/SQL procedure successfully completed.

SQL> select \* from weekly\_salary;

IDNO NAME WEEK\_SAL

---- ------------------------- ----------

1111 Ann French 1442.31

4444 Scott Brooks 1500

5555 Susan Ash 1096.15

6666 James Smith 1057.69

**PROBLEM #6:**

Using cursors, procedure and function to calculate overtime salary of employees. I created a table overtime to show overtime salries.

Create table overtime

(idno varchar2(4),

name varchar2(25),

sal number(7,2));

set verify off

set serveroutput on

DECLARE

e\_name paydata1.name%type;

e\_id paydata1.IDNO%type;

e\_salary paydata1.salary%type;

pay\_hr paydata1.payhr%type;

job\_code paydata1.jobcode%type;

w\_hrs paytran1.HOURSWK%type;

total\_hrs paytran1.HOURSWK%type;

over\_time\_hrs paytran1.HOURSWK%type;

e\_code varchar2(4);

CURSOR paydata\_cursor is

select name, idno, payhr, jobcode from paydata1

where jobcode ='H'

order by idno;

CURSOR paytran\_cursor is

select HOURSWK from paytran1

where e\_id = idno

order by idno;

**FUNCTION CalcSal**

(f\_code varchar2,f\_total\_hrs number, f\_pay\_hr number, f\_overtime\_hrs number)

return number is

f\_NewSal overtime.sal%type;

Begin

IF f\_code ='OT'then

f\_NewSal := f\_overtime\_hrs\*(f\_pay\_hr \*1.5);

ElSIF f\_code ='NO' then

f\_NewSal:= 0;

END IF;

RETURN f\_NewSal;

END CalcSal;

**PROCEDURE Over\_time**

(p\_id overtime.idno%type,

p\_name overtime.name%type,

p\_sal overtime.sal%type)AS

BEGIN

INSERT into overtime

values(p\_id,p\_name,p\_sal);

END Over\_time;

BEGIN

OPEN paydata\_cursor;

FETCH paydata\_cursor into e\_name, e\_id, pay\_hr, job\_code;

WHILE paydata\_cursor%FOUND LOOP

OPEN paytran\_cursor;

total\_hrs := 0;

FETCH paytran\_cursor into w\_hrs;

WHILE paytran\_cursor%FOUND LOOP

total\_hrs := total\_hrs + w\_hrs;

FETCH paytran\_cursor into w\_hrs;

END LOOP;

IF total\_hrs > 40 then

over\_time\_hrs := total\_hrs - 40;

ELSE

over\_time\_hrs := 0;

end if;

IF over\_time\_hrs >0 then

e\_code:= 'OT';

e\_salary := CalcSal(e\_code,total\_hrs,pay\_hr,over\_time\_hrs);

ELSE

e\_code := 'NO';

e\_salary := CalcSal(e\_code,total\_hrs,pay\_hr,over\_time\_hrs) ;

END IF;

**Over\_time**(e\_id,e\_name,e\_salary);

CLOSE paytran\_cursor;

FETCH paydata\_cursor into e\_name, e\_id, pay\_hr, job\_code;

END LOOP;

CLOSE paydata\_cursor;

end;

/

set serveroutput off

set verify on

**SQL> @ procedure2**

PL/SQL procedure successfully completed.

SQL> select \* from overtime;

IDNO NAME SAL

---- ------------------------- ----------

2222 Robert Costa 1417.5

3333 Linda Ames 0

7777 Mary Jones 270

8888 John Morse 0

**Problem #7**

Creating trigger to check on quantity.Inventory reduction error will be raised if onhand quantity is less than 5.

**CREATE OR REPLACE TRIGGER** inven\_check

before UPDATE OF onhand ON inven\_bk

for each row

WHEN(new.onhand<5)

BEGIN

RAISE\_APPLICATION\_ERROR(-20002,'inventory reduction');

END inven\_check;

/

Trigger created.

Updating table invent\_bk.

set serveroutput on

set verify off

DECLARE

t\_code transaction. TRANS\_NO %type :='&input\_no';

item\_no inventory.itemno%type :='&input\_item';

quantity\_onhand inventory.onhand%type ;

change\_quantity transaction.quantity%type :=&input\_quantity;

new\_quantity inventory.onhand%type;

i\_code transaction.TRANS\_CODE %type := '&input\_code';

CURSOR inven\_cursor is

select itemno,onhand from inven\_bk

where itemno= item\_no;

BEGIN

insert into trans\_bk

values(t\_code , item\_no, i\_code,change\_quantity );

OPEN inven\_cursor;

FETCH inven\_cursor into item\_no, quantity\_onhand;

WHILE inven\_cursor%FOUND LOOP

new\_quantity:= quantity\_onhand ;

IF i\_code = 'p' then

new\_quantity := new\_quantity + change\_quantity;

quantity\_onhand := new\_quantity;

ELSIF i\_code = 'r' then

new\_quantity := new\_quantity + change\_quantity;

quantity\_onhand := new\_quantity;

ELSIF i\_code = 's'then

new\_quantity := new\_quantity - change\_quantity;

quantity\_onhand := new\_quantity;

END IF;

update inven\_bk

set onhand = new\_quantity

where itemno = item\_no;

FETCH inven\_cursor into item\_no, quantity\_onhand;

END LOOP;

CLOSE inven\_cursor;

end;

/

set serveroutput off

set verify on

Select \* from invent\_bk;

ITEM ITEMNAME ONHAND PRICE

---- --------------- ---------- ----------

1111 Good Night Moon 86 12.99

2222 Heidi 2 15

3333 Adven Reddy Fox 6 20

5555 Building Blocks 27 16

6666 Doll House 7 55

7777 Basketball 24 25

6 rows selected.

SQL> @ trigger

Enter value for input\_no: 15

old 2: t\_code transaction. TRANS\_NO %type :='&input\_no';

new 2: t\_code transaction. TRANS\_NO %type :='15';

Enter value for input\_item: 6666

old 3: item\_no inventory.itemno%type :='&input\_item';

new 3: item\_no inventory.itemno%type :='6666';

Enter value for input\_quantity: 4

old 5: change\_quantity transaction.quantity%type :=&input\_quantity;

new 5: change\_quantity transaction.quantity%type :=4;

Enter value for input\_code: s

old 7: i\_code transaction.TRANS\_CODE %type := '&input\_code';

new 7: i\_code transaction.TRANS\_CODE %type := 's';

**DECLARE**

**\***

**ERROR at line 1:**

**ORA-20002: inventory reduction**

**ORA-06512: at "EGUPTA11.INVEN\_CHECK", line 2**

**ORA-04088: error during execution of trigger 'EGUPTA11.INVEN\_CHECK'**

**ORA-06512: at line** 31

**Problem #8:** Do something with cursors, tables and records.

**Using Records**

set verify off

set serveroutput on

**DECLARE**

counter number

CURSOR customer\_cur is

SELECT cname, street, city ,state,zip

FROM customer;

**customer\_rec customer\_cur%rowtype;**

**BEGIN**

OPEN customer\_cur;

counter := 0;

LOOP

FETCH customer\_cur into customer\_rec;

EXIT WHEN customer\_cur%notfound;

counter := counter +1;

insert into customer\_address

values(counter,customer\_rec.cname, customer\_rec.street,customer\_rec.city,customer\_rec.state,customer\_rec.zip );

END LOOP;

END;

/

set serveroutput off

set verify on

**SQL> @record**

PL/SQL procedure successfully completed.

**select \* from customer\_address**

ID NAME STREET CITY ST ZIP

-- -------------------- --------------- --------------- -- -----

1 Stephen Jones 2 State Street Beantown MA 12345

2 Stacy Brown 4 High Street Newtown CA 21345

3 John Smith 10 New Street Anytown MI 34567

4 Kimbell Jones 65 East Street Happytown CA 42345

5 Charles Wood 34 State Street Beantown MA 12345

6 DAvid Manson 15 Mark Street Georgetown MA 32345

6 rows selected.

**Using tables**

set verify off

set serveroutput on

DECLARE

TYPE emp\_table IS TABLE of employee.name%type index by binary\_integer;

emp\_name emp\_table ;

counter number(5);

e\_name employee.name%type;

CURSOR emp\_cursor is

SELECT name FROM employee;

BEGIN

open emp\_cursor;

counter := 0;

fetch emp\_cursor into e\_name;

while emp\_cursor %FOUND LOOP

emp\_name(counter) := e\_name;

counter := counter+1;

dbms\_output.put\_line(counter|| ' '|| emp\_name(counter) );

fetch emp\_cursor into e\_name;

END LOOP;

close emp\_cursor;

END;

/

set serveroutput off

set verify on