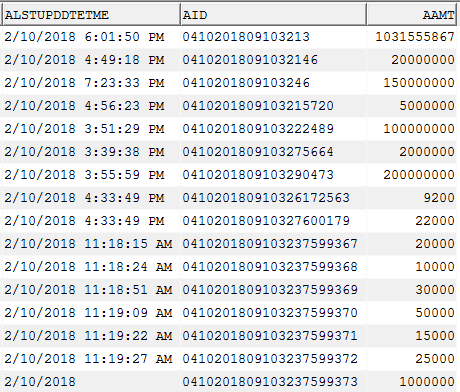
**EXAM 3**

1. **See the tTEMP table**



Write a **best query** to display all information that is updated on 02-Oct-2018

Select t.alstupddtetme,t.aid,t.aamt

from tTEMP t

where t.alstupddtetme = ‘02-Oct-2018’

1. **There are three tables: tTableA, tTableB, tTableC**



Write **best query** to insert data from tTableA, tTableB, tTableC into tTableD

Insert into tTableD

select \* from tTableA

union all

select \* from tTableB

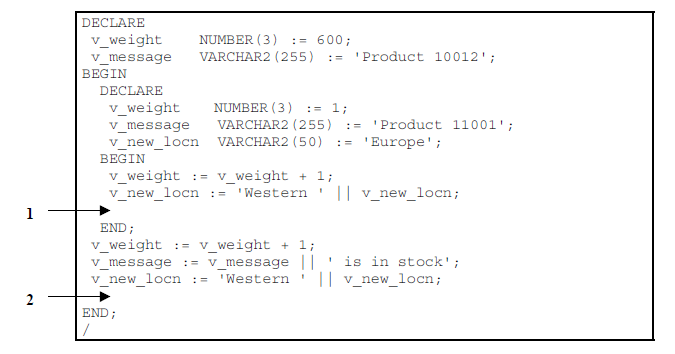
union all

select \* from tTableC;

1. **Which SQL command performs best (results is very large)**



1. Select \* from temployees where aempid in (select aempid from tdepartments);
2. Select \* from temployees a where exists (select 1 from tdepartments b where a.aempid =b.aempid);
3. Select distinct a.\* from temployees a, tdepartments b where a.aempid = b.aempid;
4. **In this practice, you examine and write executable statements.**



Evaluate the preceding PL/SQL block and determine the data type and value of each of the following variables, according to the rules of scoping.

a) The value of v\_weight at position 1 is: 2

b) The value of v\_new\_locn at position 1 is: Western Europe

c) The value of v\_weight at position 2 is: 3

d) The value of v\_message at position 2 is: Product 11001 is in stock

e) The value of v\_new\_locn at position 2 is: Western Western Europe

1. **See tDepartments table**



Declare a cursour c\_dept to retrieve adepid and adepname for those departments with adepid less than 100. Order by adepid

**DECLARE**

**CURSOR** c\_dept **IS**

**SELECT** t.adepid, t.adepname

**FROM** tdepartments t

**WHERE** t.adepid < 100

**ORDER** **BY** t.adepid;

v\_adepid tdepartments.adepid%**TYPE**;

v\_adepname tdepartments.adepname%**TYPE**;

**BEGIN**

**OPEN** c\_dept;

**LOOP**

**FETCH** c\_dept **INTO** v\_adepid, v\_adepname;

EXIT **WHEN** c\_dept%**NOTFOUND**;

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || v\_adepid || ', Department Name: ' || v\_adepname);

**END** **LOOP**;

**CLOSE** c\_dept;

**END**;

1. See temployees table



In the executable section, get all information from the temployees table by v\_aempname. Display selected information about employee. The sample output is as followes:

Employee ID: 1 Employee Name: Joseph

**DECLARE**

v\_empid temployees.emp\_id%**TYPE**;

v\_empname temployees.emp\_name%**TYPE**;

**BEGIN**

-- lay in4

**SELECT** emp\_id, emp\_name

**INTO** v\_empid, v\_empname

**FROM** temployees

**WHERE** emp\_name = v\_aempname;

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_empid || ' Employee Name: ' || v\_empname);

**END**;

1. See tEmployees table:



1. In the declarative section, declare two variable: v\_aempname of type tEmployees.aempname and v\_aempsal of tEmployees.aempsal. Initialize the latter to 1500.

**DECLARE**

v\_aempname tEmployees.aempname;

v\_aempsal tEmployees.aempsal := 1500;

1. In the executable section, retrieve the names of employees whose salaries are equal to the values in v\_aempsal.

* If the salary entered returns only one row, insert into the tMessage table the employee’s name and the salary amount.

Note: Do not use explicit cursors.

* If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert the tMessage the message “No employee with salary of <salary>.”
* If the salary entered returns multiple rows, handle the exception with an appropriate exception handler and insert into the tMessage table the message “More than one employee with a salary of <salary>.”

**DECLARE**

v\_employee\_count **NUMBER**;

v\_aempsal **number** := 2000;

v\_aempname **varchar2**(20);

**BEGIN**

**SELECT** **COUNT**(\*)

**INTO** v\_employee\_count

**FROM** tEmployees

**WHERE** aempsal = v\_aempsal;

**IF** v\_employee\_count = 1 **THEN**

**SELECT** aempname **INTO** v\_aempname

**FROM** tEmployees

**WHERE** aempsal = v\_aempsal;

**INSERT** **INTO** tMessage **VALUES** (v\_aempname,v\_aempsal);

**END** **IF**;

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

**INSERT** **INTO** tMessage **VALUES** ('No employee with salary of '|| v\_aempsal,v\_aempsal);

--DBMS\_OUTPUT.PUT\_LINE('No employee with salary of '|| v\_aempsal);

**WHEN** TOO\_MANY\_ROWS **THEN**

**INSERT** **INTO** tMessage **VALUES** ('More than one employee with a salary of '|| v\_aempsal,v\_aempsal);

--DBMS\_OUTPUT.PUT\_LINE('More than one employee with a salary of '|| v\_aempsal);

* **END**;

1. See tRegions table:



1. Create procedure sptregions\_i to insert information according to template.

CREATE OR REPLACE PROCEDURE sptregions\_i(

p\_regname IN tRegions.aregname%TYPE

)

AS

v\_next\_id tRegions.aregid%TYPE;

SELECT MAX(aregid) + 1

INTO v\_next\_id

FROM tRegions;

INSERT INTO tRegions(aregid, aregname)

VALUES (v\_next\_id, p\_regname);

DBMS\_OUTPUT.PUT\_LINE('Record inserted successfully');

END;

1. Create and execute an anonymous block to invoke the sptregions\_i procedure with parameter value.

DECLARE

v\_regname tRegions.aregname%TYPE := 'Region X';

BEGIN

sptregions\_i(v\_regname);

END;

1. Drop the sptregions\_i procedure by issuing the following command.

Drop procedure sptregions\_i;

1. Rename sptregions\_i to sptregions\_insert.

ALTER PROCEDURE sptregions\_i RENAME TO sptregions\_insert;