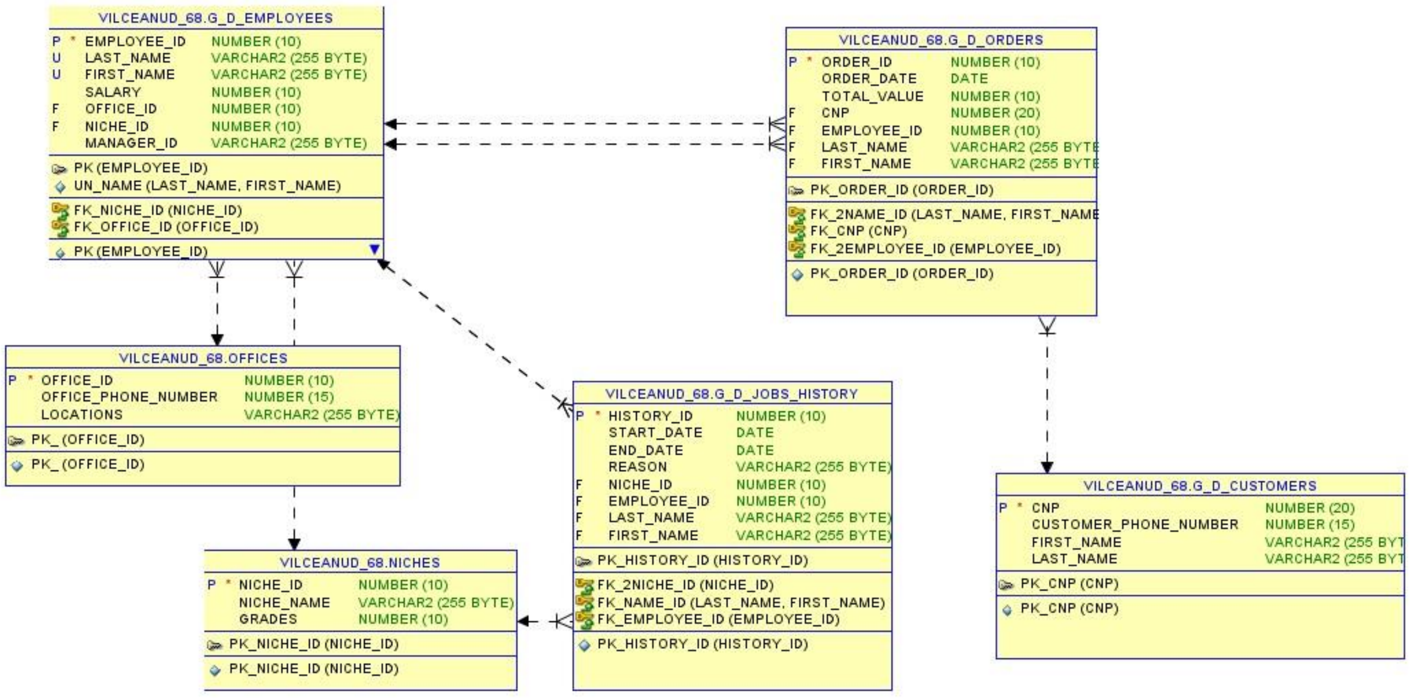
SGBD Oracle Project

Vîlceanu Diana-Maria
group 1068

Introduction

I started this project from my previous schema from the first semester.
The schema was based on the idea that I had about how a graphic design department would look like, the tables were structured correctly and the links to them are shown in the picture below.
This is the schema which describes the design of the database:


VILCEANUD_68.G_D_EMPLOYEES
P * EMPLOYEE_ID NUMBER (10)
U LAST_NAME VARCHAR2 (255 BYTE)
U FIRST_NAME VARCHAR2 (255 BYTE)
SALARY NUMBER (10)
F OFFICE_ID NUMBER (10)
F NICHE_ID NUMBER (10)
MANAGER_ID VARCHAR2 (255 BYTE)
PK (EMPLOYEE_ID)
UN_NAME (LAST_NAME, FIRST_NAME)
FK_NICHE_ID (NICHE_ID)
FK_OFFICE_ID (OFFICE_ID)
PK (EMPLOYEE_ID)
VILCEANUD_68.G_D_ORDERS
P * ORDER_ID NUMBER (10)
ORDER_DATE DATE
TOTAL_VALUE NUMBER (10)
F CNP NUMBER (20)
F EMPLOYEE_ID NUMBER (10)
F LAST_NAME VARCHAR2 (255 BYTE
F FIRST_NAME VARCHAR2 (255 BYTE
PK_ORDER_ID (ORDER_ID)
FK_2NAME_ID (LAST_NAME, FIRST_NAME
FK_CNP (CNP)
FK_2EMPLOYEE_ID (EMPLOYEE_ID)
PK_ORDER_ID (ORDER_ID)
VILCEANUD_68.OFFICES
P * OFFICE_ID NUMBER (10)
OFFICE_PHONE_NUMBER NUMBER (15)
LOCATIONS VARCHAR2 (255 BYTE)
PK_ (OFFICE_ID)
PK_ (OFFICE_ID)
VILCEANUD_68.G_D_JOBS_HISTORY
P * HISTORY_ID NUMBER (10)
START_DATE DATE
END_DATE DATE
REASON VARCHAR2 (255 BYTE)
F NICHE_ID NUMBER (10)
F EMPLOYEE_ID NUMBER (10)
F LAST_NAME VARCHAR2 (255 BYTE)
F FIRST_NAME VARCHAR2 (255 BYTE)
PK_HISTORY_ID (HISTORY_ID)
FK_2NICHE_ID (NICHE_ID)
FK_NAME_ID (LAST_NAME, FIRST_NAME)
FK_EMPLOYEE_ID (EMPLOYEE_ID)
PK_HISTORY_ID (HISTORY_ID)
VILCEANUD_68.G_D_CUSTOMERS
P * CNP NUMBER (20)
CUSTOMER_PHONE_NUMBER NUMBER (15)
FIRST_NAME VARCHAR2 (255 BYT
LAST_NAME VARCHAR2 (255 BYT
PK_CNP (CNP)
PK_CNP (CNP)
VILCEANUD_68.NICHES
P * NICHE_ID NUMBER (10)
NICHE_NAME VARCHAR2 (255 BYTE)
GRADES NUMBER (10)
PK_NICHE_ID (NICHE_ID)
PK_NICHE_ID (NICHE_ID)

Note: the schema was not modified

The exercises

1. **Requirement**

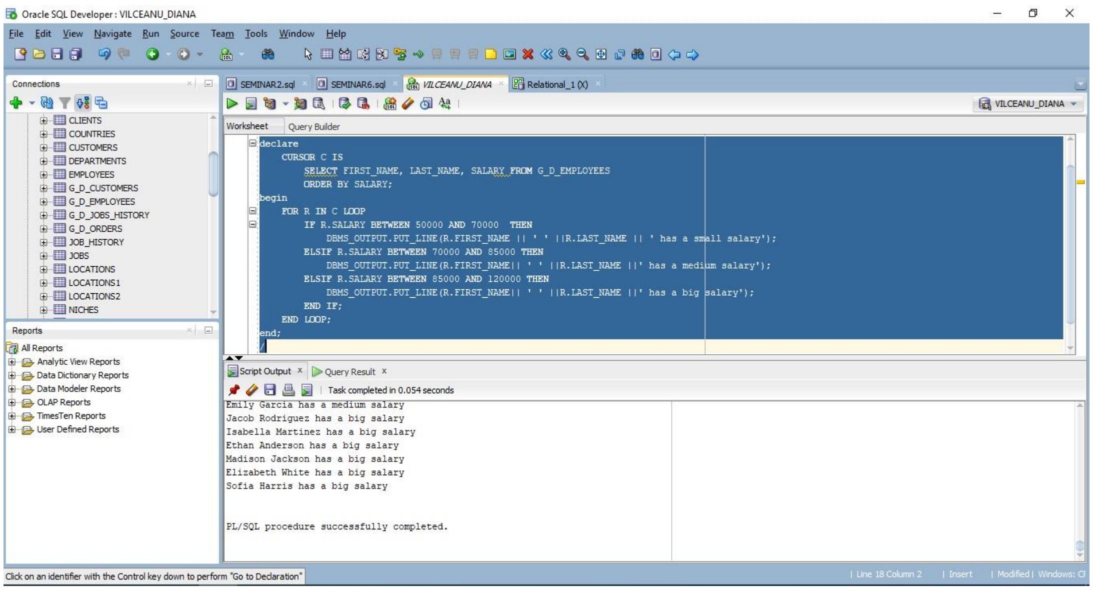
Declare a cursor named c which selects the FIRST_NAME, LAST_NAME and SALARY from the G_D_EMPLOYEES table and orders them by SALARY, then printing a message for each of them regarding their salary range: small salary for salaries between 50,000 and 70,000, medium salary for salaries between 70,000 and 85,000 and big salary for salaries between 85,000 and 120,000.

Solution

```
set serveroutput on

declare
   CURSOR C IS
      SELECT FIRST_NAME, LAST_NAME, SALARY FROM G_D_EMPLOYEES
      ORDER BY SALARY;
begin
   FOR R IN C LOOP
      IF R.SALARY BETWEEN 50000 AND 70000  THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME || ' ' ||R.LAST_NAME || ' has a
small salary');
      ELSIF R.SALARY BETWEEN 70000 AND 85000 THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME|| ' ' ||R.LAST_NAME ||' has a
medium salary');
      ELSIF R.SALARY BETWEEN 85000 AND 120000 THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME|| ' ' ||R.LAST_NAME ||' has a
big salary');
      END IF;
   END LOOP;
end;
/
```

Screenshot



```
declare
   CURSOR C IS
      SELECT FIRST_NAME, LAST_NAME, SALARY FROM G_D_EMPLOYEES
      ORDER BY SALARY;
begin
   FOR R IN C LOOP
      IF R.SALARY BETWEEN 50000 AND 70000  THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME || ' ' ||R.LAST_NAME || ' has a small salary');
      ELSIF R.SALARY BETWEEN 70000 AND 85000 THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME|| ' ' ||R.LAST_NAME ||' has a medium salary');
      ELSIF R.SALARY BETWEEN 85000 AND 120000 THEN
         DBMS_OUTPUT.PUT_LINE(R.FIRST_NAME|| ' ' ||R.LAST_NAME ||' has a big salary');
      END IF;
   END LOOP;
end;
/
```

Task completed in 0.054 seconds

```
Emily Garcia has a medium salary
Jacob Rodriguez has a big salary
Isabella Martinez has a big salary
Ethan Anderson has a big salary
Madison Jackson has a big salary
Elizabeth White has a big salary
Sofia Harris has a big salary


PL/SQL procedure successfully completed.
```

2. Requirement

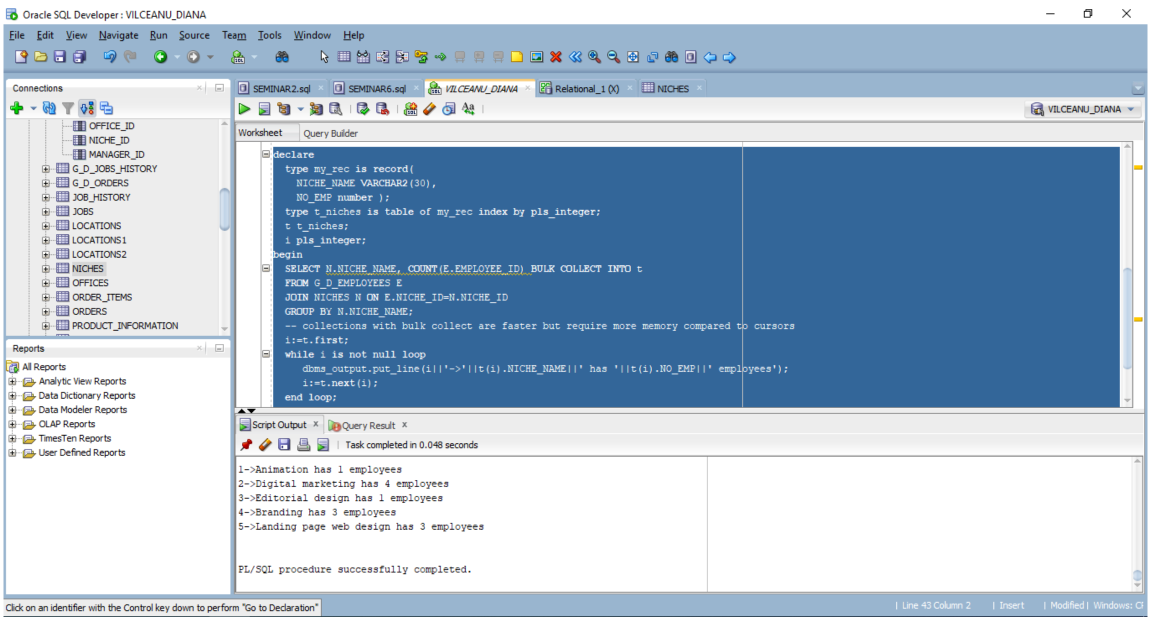
Declare a collection type named “t_niches” that stores records of niche names and number of employees, indexed by pls_integer, which counts the number of employees from G_D_EMPLOYEES table working on each niche.

Solution

```
declare
  type my_rec is record(
    NICHE_NAME VARCHAR2(30),
    NO_EMP number );
  type t_niches is table of my_rec index by pls_integer;
  t t_niches;
  i pls_integer;
begin
  SELECT N.NICHE_NAME, COUNT(E.EMPLOYEE_ID) BULK COLLECT INTO t
  FROM G_D_EMPLOYEES E
  JOIN NICHES N ON E.NICHE_ID=N.NICHE_ID
  GROUP BY N.NICHE_NAME;
  -- collections with bulk collect are faster but require more memory compared to cursors
  i:=t.first;
  while i is not null loop
     dbms_output.put_line(i||'->'||t(i).NICHE_NAME||' has '||t(i).NO_EMP||' employees');
     i:=t.next(i);
  end loop;
  t.delete;

end;
/
```

Screenshot



```
declare
  type my_rec is record(
    NICHE_NAME VARCHAR2(30),
    NO_EMP number );
  type t_niches is table of my_rec index by pls_integer;
  t t_niches;
  i pls_integer;
begin
  SELECT N.NICHE_NAME, COUNT(E.EMPLOYEE_ID) BULK COLLECT INTO t
  FROM G_D_EMPLOYEES E
  JOIN NICHES N ON E.NICHE_ID=N.NICHE_ID
  GROUP BY N.NICHE_NAME;
  -- collections with bulk collect are faster but require more memory compared to cursors
  i:=t.first;
  while i is not null loop
     dbms_output.put_line(i||'->'||t(i).NICHE_NAME||' has '||t(i).NO_EMP||' employees');
     i:=t.next(i);
  end loop;
```

Task completed in 0.048 seconds

```
1->Animation has 1 employees
2->Digital marketing has 4 employees
3->Editorial design has 1 employees
4->Branding has 3 employees
5->Landing page web design has 3 employees


PL/SQL procedure successfully completed.
```

Click on an identifier with the Control key down to perform "Go to Declaration"

3. Requirement

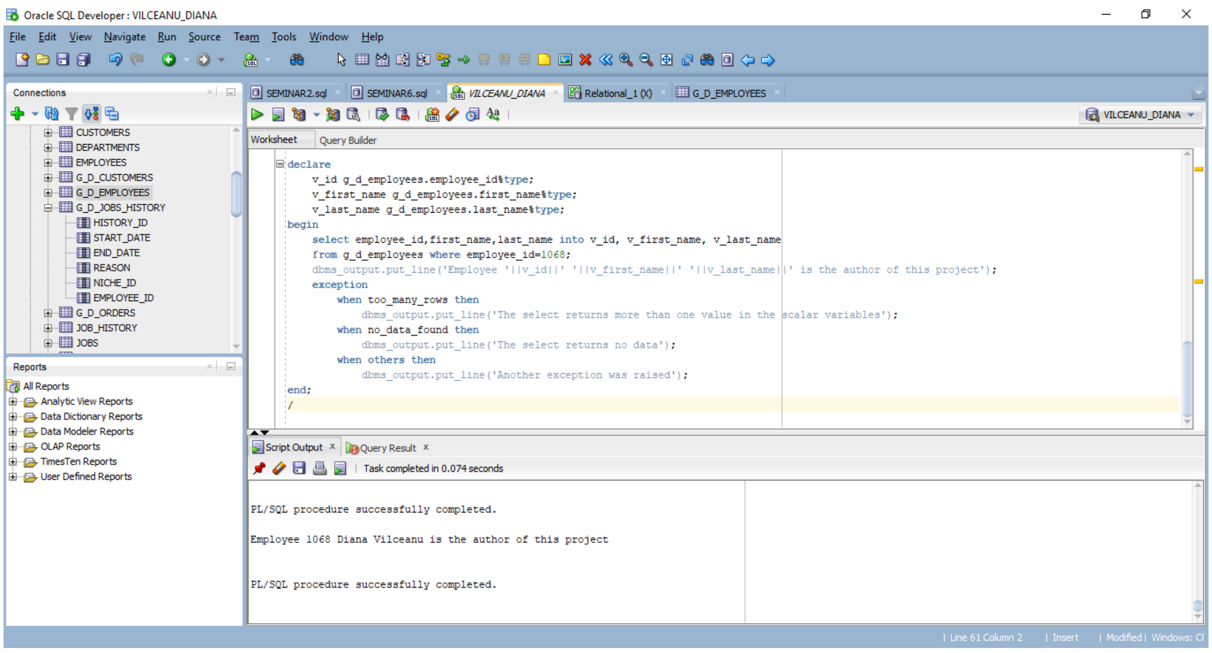
Declare 3 variables to store the EMPLOYEE_ID, FIRST_NAME, LAST_NAME of the employee with the id 1068. Handle the exceptions when the select statement returns too many rows, no rows, or any other type of exception.

Note: this is the exercise where my group and my full name is displayed.

Solution

```
declare
   v_id g_d_employees.employee_id%type;
   v_first_name g_d_employees.first_name%type;
   v_last_name g_d_employees.last_name%type;
begin
   select employee_id,first_name,last_name into v_id, v_first_name, v_last_name
   from g_d_employees where employee_id=1068;
   dbms_output.put_line('Employee '||v_id||' '||v_first_name||' '||v_last_name||' is the author of
this project');
   exception
      when too_many_rows then
         dbms_output.put_line('The select returns more than one value in the scalar
variables');
      when no_data_found then
         dbms_output.put_line('The select returns no data');
      when others then
         dbms_output.put_line('Another exception was raised');
end;
/
```

Screenshot



```
declare
    v_id g_d_employees.employee_id%type;
    v_first_name g_d_employees.first_name%type;
    v_last_name g_d_employees.last_name%type;
begin
    select employee_id,first_name,last_name into v_id, v_first_name, v_last_name
    from g_d_employees where employee_id=1068;
    dbms_output.put_line('Employee '||v_id||' '||v_first_name||' '||v_last_name||' is the author of this project');
    exception
        when too_many_rows then
            dbms_output.put_line('The select returns more than one value in the scalar variables');
        when no_data_found then
            dbms_output.put_line('The select returns no data');
        when others then
            dbms_output.put_line('Another exception was raised');
end;
/
```

Task completed in 0.074 seconds

```
PL/SQL procedure successfully completed.

Employee 1068 Diana Vilceanu is the author of this project


PL/SQL procedure successfully completed.
```

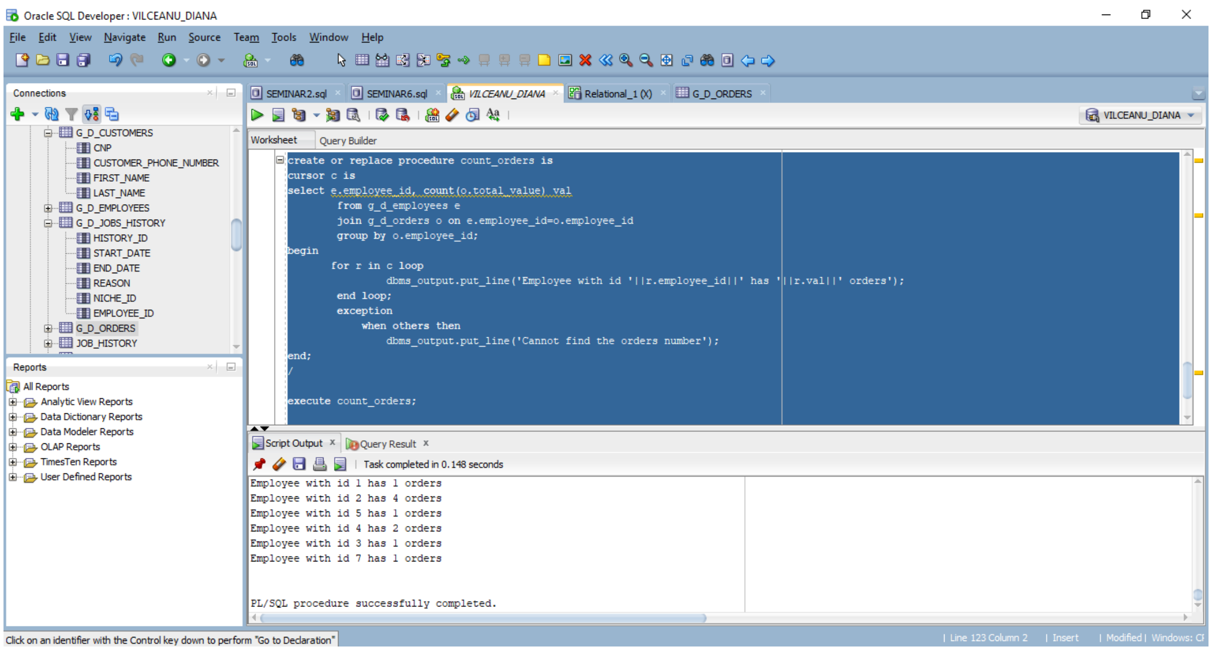
4. Requirement

Create a procedure which counts the orders from the G_D_ORDERS table and groups them by the employee_id using a cursor. Raise an exception if the procedure cannot be executed and use execute immediate for that procedure.

Solution

```
create or replace procedure count_orders is
cursor c is
select e.employee_id, count(o.total_value) val
        from g_d_employees e
        join g_d_orders o on e.employee_id=o.employee_id
        group by o.employee_id;
begin
       for r in c loop
              dbms_output.put_line('Employee with id '||r.employee_id||' has '||r.val||' orders');
        end loop;
        exception
            when others then
              dbms_output.put_line('Cannot find the orders number');
end;
/
execute count_orders;
declare
  v_query varchar2(100);
begin
  v_query := 'begin count_orders; end;';
  execute immediate v_query;
end;
/
```

Screenshot



```
create or replace procedure count_orders is
cursor c is
select e.employee_id, count(o.total_value) val
        from g_d_employees e
        join g_d_orders o on e.employee_id=o.employee_id
        group by o.employee_id;
begin
       for r in c loop
              dbms_output.put_line('Employee with id '||r.employee_id||' has '||r.val||' orders');
        end loop;
        exception
            when others then
              dbms_output.put_line('Cannot find the orders number');
end;
/

execute count_orders;
```

Task completed in 0.148 seconds

```
Employee with id 1 has 1 orders
Employee with id 2 has 4 orders
Employee with id 5 has 1 orders
Employee with id 4 has 2 orders
Employee with id 3 has 1 orders
Employee with id 7 has 1 orders


PL/SQL procedure successfully completed.
```

5. **Requirement**

Create a procedure which uses a cursor to select how many people are employed per each location. The procedure will show the locations with few and with many employees and will raise a user defined exception if the location cannot be found.

Solution

```
create or replace procedure employees_per_location is
e_no_emp exception;
pragma exception_init(e_no_emp,-20999);
cursor c is
select upper(o.locations) locations, count(e.employee_id) no_of_emp from g_d_employees e
join offices o on e.office_id=o.office_id
group by o.locations;
BEGIN
for r in c loop
   case
      when r.no_of_emp between 1 and 3 then
         dbms_output.put_line('Location '||r.locations||' has few employees');
      when r.no_of_emp > 3 then
         dbms_output.put_line('Location '||r.locations||' has many employees');
   end case;
end loop;

exception
   when e_no_emp then
     DBMS_OUTPUT.PUT_LINE('Exception->Location with no employees');
END;
/

execute employees_per_location;
```

6. **Requirement**

Create a procedure called sum_salaries which will calculate the total sum of the employees salaries using a cursor and a loop. Raise an explicit exception when no data will be found in the G_D_EMPLOYEES table.

Solution

```
CREATE OR REPLACE PROCEDURE sum_salaries
IS
  e_no_data_found exception;
  PRAGMA EXCEPTION_INIT(e_no_data_found,-00955);
  total_salaries NUMBER := 0;
  current_salary NUMBER;
  cursor c_employees IS
    SELECT salary FROM employees;
BEGIN
  OPEN c_employees;
  LOOP
```

```
    FETCH c_employees INTO current_salary;
    EXIT WHEN c_employees%NOTFOUND;
    total_salaries := total_salaries + current_salary;
  END LOOP;
  CLOSE c_employees;
  DBMS_OUTPUT.PUT_LINE('Total salaries: ' || total_salaries);
  EXCEPTION
    when e_no_data_found then
       dbms_output.put_line('Exception->Cannot find any data!');
END;
/

execute sum_salaries;
```

7. **Requirement**

Create a function named total_sales_value which will return the total value of sales from the G_D_ORDERS table. The function will use a cursor and will be called in a declare block.

Solution

```
create or replace function total_sales_value
return number is
total number(30):=0;
current_value number(30);
 cursor c IS
    SELECT total_value FROM g_d_orders;
begin
  OPEN c;
  LOOP
    FETCH c INTO current_value;
    EXIT WHEN c%NOTFOUND;
    total := total + current_value;
    END LOOP;
    CLOSE c;
    return total;
end;
/

DECLARE
   total number(30);
BEGIN
   total := total_sales_value();
   dbms_output.put_line('Total value of sales: ' || total);
END;
/
```

8. **Requirement**

Create a function named the_best_employees that returns a cursor which shows the employees with the highest grades from the NICHES table.

Solution

```
create or replace function the_best_employees
return SYS_REFCURSOR is
c  SYS_REFCURSOR;
begin
   OPEN c FOR
      select e.first_name, e.last_name from g_d_employees e
      join niches n on e.niche_id=n.niche_id
      where n.grades>8;
   RETURN c;
end;
/
```

9. **Requirement**

Create a function find_the_best_customer which will return the customer with the biggest number of orders and uses a cursor.

Solution

```
create function find_the_best_customer
return varchar2 is
   v_full_name varchar2(30);
   v_count_orders number;
   v_max_count_orders number := 0;
   v_best_customer varchar2(30);
   cursor c is
      SELECT c.first_name || ' ' || c.last_name AS full_name, COUNT(*) AS count_orders
      FROM g_d_customers c
      JOIN g_d_orders o ON c.cnp = o.cnp
      GROUP BY c.first_name, c.last_name;

begin
   for r in c loop
      v_count_orders := r.count_orders;
      if v_count_orders > v_max_count_orders then
         v_max_count_orders := v_count_orders;
         v_best_customer := r.full_name;
      end if;
   end loop;
   return v_best_customer;

end;
/
```

```
DECLARE
   best_cust varchar2(30);
BEGIN
   best_cust := find_the_best_customer();
   dbms_output.put_line('The best customer is ' || best_cust);
END;
/
```

10. Requirement

Create a package called my_package which contains a function that already exists (the function from 9) and the procedure called count_orders. Execute the procedure from the package.

Solution

```
create or replace package my_package as
    function f_the_best_customer return varchar2;
    procedure count_orders;
end my_package;
/

CREATE OR REPLACE PACKAGE BODY my_package AS
  FUNCTION f_the_best_customer return varchar2 IS
    v_result varchar2(100);
  BEGIN
    v_result := find_the_best_customer();
    RETURN v_result;
  END;

   PROCEDURE count_orders IS
    cursor c is
        select e.employee_id, count(o.total_value) val
        from g_d_employees e
        join g_d_orders o on e.employee_id=o.employee_id
        group by o.employee_id;
    no_orders number(30);
  BEGIN
    for r in c loop
      dbms_output.put_line('Employee with id '||r.employee_id||' has '||r.val||' orders');
    end loop;
  EXCEPTION
    WHEN others THEN
      dbms_output.put_line('Cannot find the orders number');
  END count_orders;
END my_package;
/

EXECUTE my_package.count_orders;
```

11. Requirement

Create a function which will raise the salary of the best employee. The function will receive the salary of the employee and the raise that will be added as parameters. Create a declare block in which you will test the function for the employee with id 3.

Solution

```
create or replace function salary_raise_best_employee(a in number, b in number)
return number is
c number(30);
begin
c:=a+b;
   return c;
end;
/

declare
v_salary number(30);
v_raise number(30);
v_new_salary number(30);
begin
select e.salary into v_salary
from g_d_employees e
where e.employee_id=3;
v_raise:=1000;
v_new_salary:=salary_raise_best_employee(v_salary,v_raise);
dbms_output.put_line('The new salary is '||v_new_salary);

end;
/
```

12. Requirement

Create a trigger niches_update_row_trigger which will activate before the UPDATE command will be used on NICHES table. The trigger will write a message.

Solution

```
CREATE OR REPLACE TRIGGER niches_update_row_trigger
before update on niches
for each row
declare
begin
   dbms_output.put_line('A row has been updated in the NICHES table.');
end;
/
```

13. Requirement

Create a trigger which will fill in the G_D_JOBS_HISTORY table with the new values when the user tries to insert or update the data about a new/old employee from the G_D_EMPLOYEES table.

Solution

```
CREATE OR REPLACE TRIGGER update_the_jobs_history
AFTER INSERT OR UPDATE ON g_d_employees
FOR EACH ROW
BEGIN
  IF inserting THEN
    INSERT INTO g_d_jobs_history(employee_id, end_date)
    VALUES (:NEW.employee_id, SYSDATE);
  ELSE
    UPDATE g_d_jobs_history
    SET end_date = SYSDATE
    WHERE employee_id = :OLD.employee_id AND end_date IS NULL;
  END IF;
END;
/
```

14. Requirement

Create a trigger at statement which will write a message to the user every time an action is performed on G_D_CUSTOMERS table. Name the trigger g_d_customers_statement.

Solution

```
CREATE OR REPLACE TRIGGER g_d_customers_statement
AFTER INSERT OR DELETE OR UPDATE ON g_d_customers
DECLARE
   v_action VARCHAR2(30);
BEGIN
   IF INSERTING THEN
      v_action := 'inserted';
   ELSIF DELETING THEN
      v_action := 'deleted';
   ELSIF UPDATING THEN
      v_action := 'updated';
   END IF;

   DBMS_OUTPUT.PUT_LINE('The G_D_CUSTOMERS table was ' || v_action);
END;
/
```

15. Requirement

Create a trigger at statement which will be activated whenever more that 20 rows from the G_D_EMPLOYEES tables will be affected by INSERT, UPDATE or DELETE. Name it g_d_employees_statement.

Solution

```
CREATE OR REPLACE TRIGGER g_d_employees_statement
AFTER INSERT OR UPDATE OR DELETE ON g_d_employees
DECLARE
  number_of_rows NUMBER;
BEGIN
  SELECT COUNT(*) INTO number_of_rows FROM g_d_employees;
  IF number_of_rows > 20 THEN
    DBMS_OUTPUT.PUT_LINE('The trigger was activated because more than 20 rows were
affected.');
  END IF;
END;
/
```

16. Requirement

Create a declare statement which will use an implicit cursor SQL%ROWCOUNT to count the number of the row updated after modifying the REASON field in G_D_JOBS_HISTORY table to ‘Retired’ if there are more than 700 days since the END_DATE of job.

Solution

```
DECLARE
  v_rowcount NUMBER(30);
BEGIN
  UPDATE g_d_jobs_history
  SET reason = 'Retired'
  WHERE end_date < SYSDATE - 700;

  v_rowcount := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('The number of people retired from job is: ' || v_rowcount);
END;
/
```