HR queries

This kata may be used to learn about the different type of cursors and to know *how to choose the right cursor* depending on your needs.

Setup

This kata relies on data from the HR sample schema.

The setup details vary depending on your situation:

- HR schema is installed and you have access : great, setup is done!
- HR schema is not installed and you have access to the SYS password, read the standard setup section.
- HR schema is not installed and you have access to a user able to create another user : read the new user setup section.
- HR schema is not installed and you can't create another user : read the existing schema setup section.

Standard setup

In this scenario the HR user will be dropped and recreated.

Connect to the database using sql*plus (user SYS or SYSTEM) and run script `setup/hr_main.sql`:

```
@hr_main.sql hr_user_password default_tablespace temporary_tablespace sys_pas
sword tns_alias
```

To perform cleanup, run script `setup/hr_drop.sql` or simply drop user HR.

Setup in an a new schema

In this scenario a new user will be created : it will be the owner of all RH objects.

Connect to the database using sql*plus (user able to create another user) and run script `setup/new_user_main.sql`.

```
@hr_main.sql new_user_name new_user_password default_tablespace temporary_tab
lespace super_user_name super_user_password tns_alias
```

To perform cleanup, run script `setup/no_user_drop.sql` or simply drop the new user.

Setup in an existing schema

In this scenario an existing user will will be the owner of all RH objects.

Connect to the database using sql*plus (existing user) and run script `setup/no_user_main.sql`.

To perform cleanup, run script `setup/no_user_drop.sql`.

Walkthrough

The kata will require to write several functions, the preliminary step is to declare a package that will regroup them all. Then follow the below steps in the order.

Step 1

Write a function that returns the first and last name of an employee given his id. If the id matches more than one name or if it doesn't match any the function should raise an error.

Step 2

Write a procedure that displays the following columns of the EMPLOYEES table :
FIRST_NAME, LAST_NAME, HIRE_DATE, SALARY

The procedure should display all rows for the employees of the "Shipping" departement (columns may be comma-separated).

Step 3

1 - Write a function that returns a result set composed of the following columns of the EMPLOYEES table :
FIRST_NAME, LAST_NAME, EMPLOYEE_ID

The query should return all rows for the employees of the "IT" departement.

2 - Write a procedure using the above function and displaying the result set.

Step 4

Write a function that returns a result set composed of the following columns of the EMPLOYEES table :
FIRST_NAME, LAST_NAME, SALARY

The contents of the result set depends on the value of an argument `salary_rank`:

- If the value is "ALL" then the query should return all rows for the "Sales Representative" employees.
- If the value is "AVG_ABOVE" then the query should return all rows for the "Sales Representative" employees earning more than the average salary of the "Sales" department.

Expected result for the "AVG_ABOVE" :

```
FIRST_NAME           LAST_NAME                 SALARY
-------------------- ------------------------- ----------
Peter                Tucker                    10000
David                Bernstein                 9500
Peter                Hall                      9000
Janette              King                      10000
Patrick              Sully                     9500
Allan                McEwen                    9000
Clara                Vishney                   10500
Danielle             Greene                    9500
Lisa                 Ozer                      11500
Harrison             Bloom                     10000
Tayler               Fox                       9600
Ellen                Abel                      11000

12 rows selected.
```

Step 5

Extending the function created in previous step, a new value "AVG_BELOW" may be assigned to the argument `salary_rank`. In this case the function should return a result set composed of the following columns of the EMPLOYEES table :
FIRST_NAME, LAST_NAME, PHONE_NUMBER, SALARY

In this case the query should return all rows for the "Sales Representative" employees earning less than the average salary of the "Sales" department.

Expected result for "AVG_BELOW" :

```
FIRST_NAME           LAST_NAME                 PHONE_NUMBER         SALARY
-------------------- ------------------------- -------------------- ---------
-
Christopher          Olsen                     011.44.1344.498718   8000
Nanette              Cambrault                 011.44.1344.987668   7500
Oliver               Tuvault                   011.44.1344.486508   7000
Lindsey              Smith                     011.44.1345.729268   8000
Louise               Doran                     011.44.1345.629268   7500
Sarath               Sewall                    011.44.1345.529268   7000
Mattea               Marvins                   011.44.1346.329268   7200
David                Lee                       011.44.1346.529268   6800
Sundar               Ande                      011.44.1346.629268   6400
Amit                 Banda                     011.44.1346.729268   6200
William              Smith                     011.44.1343.629268   7400
Elizabeth            Bates                     011.44.1343.529268   7300
Sundita              Kumar                     011.44.1343.329268   6100
Alyssa               Hutton                    011.44.1644.429266   8800
Jonathon             Taylor                    011.44.1644.429265   8600
Jack                 Livingston                011.44.1644.429264   8400
Charles              Johnson                   011.44.1644.429262   6200

17 rows selected.
```

Step 6

Write a function that returns a result set composed of the following columns of the EMP_DETAILS_VIEW view :
EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, DEPARTMENT_NAME, JOB_TITLE, COUNTRY_NAME

The function accepts three optional parameters `p_region_name`, `p_dept_name`, `p_max_salary`.

These parameters will be evaluated as predicates :
EMP_DETAILS_VIEWS.REGION_NAME like p_region_name
EMP_DETAILS_VIEWS.SALARY < p_max_salary
EMP_DETAILS_VIEWS.DEPARTMENT_NAME like p_dept_name

Expected results for p_region_name='meri', p_dept_name='IT', p_max_salary = 4800 :

```
EMPLOYEE_ID FIRST_NAME      LAST_NAME     SALARY  DEPARTMENT_NAME  JOB_TITLE
  COUNTRY_NAME
----------- --------------- ------------- ------- ---------------- ----------
```

```
- ------------------------
107         Diana           Lorentz       4200    IT                Programmer
  United States of America
```

Expected results for p_region_name='eur', p_dept_name='Public', p_max_salary not specified :

```
EMPLOYEE_ID FIRST_NAME LAST_NAME  SALARY  DEPARTMENT_NAME                JOB_
TITLE                          COUNTRY_NAME
----------- ---------- ---------- ------- ------------------------------ ----
------------------------------ -------------
204         Hermann    Baer         10000   Public Relations               Publ
ic Relations Representative    Germany
```

Expected results for p_region_name not specified, p_dept_name not specified, p_max_salary = 2500 :

```
EMPLOYEE_ID FIRST_NAME  LAST_NAME  SALARY DEPT_NAME JOB_TITLE   COUNTRY_NAME
----------- ----------- ---------- ------ --------- ----------- -------------
-----------
127         James       Landry     2400   Shipping  Stock Clerk United States
of America
128         Steven      Markle     2200   Shipping  Stock Clerk United States
of America
132         TJ          Olson      2100   Shipping  Stock Clerk United States
of America
135         Ki          Gee        2400   Shipping  Stock Clerk United States
of America
136         Hazel       Philtanker 2200   Shipping  Stock Clerk United States
of America
```