CPRG 307
Lab 5: Exception Handling

Student: _____________________

Mark: ___ / 2 (*one mark for each deliverable*)

Lab Objectives

Here is what you will be able to do when you complete each objective:

- Describe the database vendor's error-handling environment within the programming block.
- Demonstrate the integration of exception handlers into a program.
- Use predefined exceptions provided by the database vendor.
- Use user-defined exceptions.

Lab Instructions

To complete this lab, follow the steps below. This lab is due on the day and time indicated by your instructor.

Steps:

☐ 1. ATTEND the lecture on the material that will be performed in the lab exercise.

☐ 2. COMPLETE the out-of-class learning activities as indicated by your instructor.

☐ 3. COMPLETE the prelab tasks identified in the lab document before the lab class, making sure to submit solutions to the appropriate forum and thread in the D2L discussion board.

☐ 4. COMPLETE the tasks identified in the lab document, making sure to submit solutions to the appropriate forum and thread in the D2L discussion board.

☐ 5. COMPLETE the post lab tasks identified in the lab document after the lab has been completed.

Deliverables

☐ 1. SUBMIT the complete and tested prelab code by the date and time indicated by your instructor to the appropriate forum and topic in the D2L discussion board.

☐ 2. SUBMIT the complete and tested lab code by the date and time indicated by your instructor to the appropriate forum and topic in the D2L discussion board.

For this lab, all code should be placed in the body of the discussion board posts – not as an attachment.

Unless stated otherwise, code from all tasks should be included in discussion board posts.

Prelab Tasks

The following questions use the Astra Talent Agency (ATA) table set.

☐ 1. Write the PL/SQL code for the following problem:

Retrieve the client id for contract number 8 (make this a constant). This will cause an exception in your code. Capture this exception using a predefined exception handler and print an appropriate error message to the screen.

☐ 2. Write the PL/SQL code for the following problem:

Retrieve the contract fee for client id 0000020 (make this a constant). This will cause an exception in your code. Capture this exception using a predefined exception handler and print an appropriate error message to the screen. *Hint: CLIENT_ID is a VARCHAR2 datatype, this means the value above, because of the leading zeros, must be placed between single quotes.*

☐ 3. Write the PL/SQL code for the following problem:

Add $500.00 dollars (make this a constant) to the contract fee if the event type is *Retirement Party* (make this a constant). If no rows are updated, create a user defined exception that will print an appropriate error message to the screen. Solve using ***RAISE***.

☐ 4. Write the PL/SQL code for the following problem:

Add $500.00 dollars (make this a constant) to the contract fee if the event type is *Retirement Party* (make this a constant). If no rows are updated, create a user defined exception that will print an appropriate error message to the screen. Solve using ***RAISE_APPLICATION_ERROR***.

Lab Tasks

The following questions use the EMP table set.

Although you do not have to submit a flowchart or test plan for this lab problem, it is recommended you still create them as this will help you to breakdown and solve the problem. When your instructor posts the solutions for the lab to the discussion board, they will also include the flowchart, test plan, and test code to provide you with examples of the entire problem lifecycle.

☐ 1. CREATE **<u>and</u>** thoroughly test a PL/SQL coded solution for the following problem:

The president of the company wants to ensure that everyone is receiving a "fair" wage. He has asked you to modify salaries of everyone, except him, using the following guidelines (each point uses the salary from the previous step unless otherwise specified):

- If anyone makes more than the president does, they should have their salaries reduced by 50% or 25% less than the president makes (whichever one would be less).
- If anyone makes less than $100, their salary should be increased by 10%, but only if the **original** average salary for the <u>entire</u> company is still more than their new raised salary.
  - The company has decided that they only want to look at the average salary **<u>before</u>** the changes from the first step are performed rather than after.
- If an employee's commission is more than 22% of their salary, the commission should be changed to the lowest commission in their department.
  - If an employee does not have a commission (i.e. NULL or 0 value), no changes should be made to their commission.
- If an employee does not have a *manager* for their department, no changes should be made to this employee and an error message should be written to the screen (this should not prevent other employees from being processed).

Restrictions:

- Must use one looping structure (with cursor)
- Can only use a SELECT…INTO to get: (1) the average salary of the company, (2) the president's salary, (3) the lowest commission in a department, and (4) determine if there is a manager assigned to the employee's department
  - The first two queries identified above must be performed outside of the looping structure
  - All DML commands must be inside the loop and utilize the cursor data
  - The last two queries mentioned above must be inside the loop and utilize the cursor data
  - Only **<u>one</u>** DML command can be used in the coded solution

- Can only hard code information that was provided in the problem
    - ‘PRESIDENT’
    - ‘MANAGER’
    - 50%
    - 25%
    - $100
    - 10%
    - 22%
- Hard coded values should be defined as constants and then the constants used in the body of the code
- Can assume the company has only one president
- Can assume a department will have only one manager
- Where transactional control statements are placed is important. For this coded solution can use only **one** COMMIT, but can have multiple ROLLBACKs depending on how exceptions are handled
- Checking to see if an employee has a department manager **must** be done after the DML task is performed (per loop iteration)
- A department manager will be assigned a department number and have the job title *MANAGER*. This is the only way to identify a department manager. The MGR value is not always a department manager.

Extra Questions to Practice

Below are extra questions to practice for the material in this module (these are extra to the lab and are for practice only):

☐ 1. Complete the following textbook problem: *Assignment 4-4: Using exception handling* on page 169-170 of your textbook. If you have questions on the results of this problem, please call your instructor over.

Post Lab Tasks

☐ 1. COMPARE your posted solutions to those posted by your instructor. If you are unsure why there are differences between the solutions, make sure to talk to your instructor.