**Assignment 2a CSC 401**

**Name:**

**Date:**

For this assignment, you will write an anonymous PL/SQL block that processes multiple rows from the database you created for the More Movie Rentals in Assignment 1a: Lab Exercise. In the second part of this assignment, you will modify a block to add code that handles exceptional situations.

**Part 1**

Now that business is becoming strong and the movie stock is growing for More Movie Rentals, the manager wants to do more inventory evaluations. One item of interest concerns any movie for which the company is holding $75 or more in value. The manager wants to focus on these movies in regards to their revenue generation to ensure the stock level is warranted. To make these stock queries more efficient, the application team decides that a column should be added to the MM\_MOVIE table named STK\_FLAG that will hold a value '\*' if stock is $75 or more. Otherwise the value should be NULL. Add the needed column and create an anonymous block that contains a CURSOR FOR loop to accomplish the task. The company will run this program monthly to update the STK\_FLAG column before the inventory evaluations.

Make sure that you display the table structure of MM\_MOVIE using DESC, and SELECT all data from this table before and after you do this exercise to show the effect of your block.

Submit all code and Oracle responses. Before submitting, edit your document to remove erroneous attempts.

**Part 2**

Here is a block that retrieves the movie title and rental count based on a movie id provided via a host variable.

SET SERVEROUTPUT ON

VARIABLE g\_movie\_id NUMBER

BEGIN

 :g\_movie\_id := 4;

END;

/

DECLARE

 v\_count NUMBER;

 v\_title mm\_movie.movie\_title%TYPE;

BEGIN

 SELECT m.movie\_title, COUNT(r.rental\_id)

 INTO v\_title, v\_count

 FROM mm\_movie m, mm\_rental r

 WHERE m.movie\_id = r.movie\_id

 AND m.movie\_id = :g\_movie\_id

 GROUP BY m.movie\_title;

 DBMS\_OUTPUT.PUT\_LINE(v\_title || ': ' || v\_count);

END;

/

Add exception handlers for errors that you can and cannot anticipate. Test the block by running it with various values of the host variable.