CSC 352/452: Database Programming

Mid-term Exam   
(300 Points)

**Late exams will not be accepted or graded.**

**CSC 352/452-501: Due on Sunday, 8/9/2015 at midnight.**

**CSC 352/452-510: Due on Monday, 8/10/2015 at midnight.**

* **The mid-term exam is a take-home exam.  You can use any of your class notes and readings to complete the exam.**
* **You may not consult in any form with any other person while doing this take-home exam.**
* **Please submit a text file containing all your answers to D2L before or on due date. All other file types (e.g., DOC, DOCX, RTF, PDF, JPG, or ZIP) will be rejected. In D2L, only the most recent submission is kept.**
* **Please review your assignment file before submitting it to make sure you have the correct one. It is your responsibility to ensure that you upload the correct assignment file. If you submit a blank/wrong file, you will simply receive a grade of zero.**

**Part I (CSC 352 and CSC 452 - 120 points)**

There are a total of 12 questions. Each question is worth 10 points. Please read each question carefully and choose the correct answer.

**Assume that the tab1 table exists in the underlying database.**

Q1. PL/SQL records of the same declared type can be compared for equality by using the equality operator (=).

DECLARE

TYPE t\_1 IS RECORD (col\_1 NUMBER, col\_2 NUMBER);

v\_1 t\_1;

v\_2 t\_1;

BEGIN

……

IF v\_1 = v\_2 THEN

DBMS\_OUTPUT.PUT\_LINE('v\_1 = v\_2');

ELSE

DBMS\_OUTPUT.PUT\_LINE('v\_1 != v\_2');

END IF;

……

END;

A. TRUE

B. FALSE

Q2. If you would like to code your PL/SQL block to retrieve multiple rows from a table, and then run through each row of output and perform some work, which of the following choices best identifies how you would do so?

A. SELECT statements

B. Implicit cursors

C. Explicit cursors

D. None of the above

Q3. Consider the following SELECT-INTO statement in a PL/SQL block. What happens if there are no rows satisfying the WHERE condition?

……

SELECT COUNT(\*)

INTO v\_n

FROM tab1

WHERE c1 > 0 AND c2 > 100;

……

A. A NO\_DATA\_FOUND exception is raised.

B. A TOO\_MANY\_ROWS exception is raised.

C. A ZERO\_DIVIDE exception is raised.

D. The SELECT-INTO statement executes successfully.

Q4. Evaluate the following PL/SQL block:

DECLARE

v\_stock BOOLEAN;

v\_quota BOOLEAN := TRUE;

v\_approval BOOLEAN := TRUE;

v\_option BOOLEAN := v\_approval;

BEGIN

v\_approval := v\_option AND v\_stock AND v\_quota;

END;

Which value is assigned to v\_approval?

A. TRUE

B. FALSE

C. NULL

D. None of the above

Q5. Evaluate the following CURSOR statement:

DECLARE

CURSOR c\_1 (p\_1 VARCHAR2(50)) IS

SELECT \*

FROM tab1

WHERE c9 = p\_1;

Why will this statement cause a syntax error?

A. The UPPER or LOWER function is missing in the WHERE clause.

B. The SELECT statement is missing the INTO clause.

C. You cannot define a cursor with a parameter in PL/SQL.

D. The size of the p\_1 parameter cannot be specified.

Q6. What is the purpose of the FETCH command in using an explicit cursor?

A. To close a cursor.

B. To define a cursor to be used later

C. To retrieve rows (records) from the active set into local variables

D. To execute the query and identify the active set.

Q7. In which section of a PL/SQL block is a WHEN NO\_DATA\_FOUND THEN clause allowed?

A. DECLARATION

B. EXECUTION

C. EXCEPTION

D. All of the above

Q8. Which guideline relates to a CURSOR FOR Loop?

FOR cursor\_idx IN cursor\_name LOOP

statement1;

statement2;

......

END LOOP;

A. The user must explicitly declare the cursor\_idx in the DECLARATION section.

B. It requires an OPEN cursor\_name statement.

C. It requires a CLOSE cursor\_name statement.

D. It does not require a FETCH statement.

E. The cursor must return at least one row.

F. All of the above

Q9. Evaluate the following PL/SQL block:

DECLARE

CURSOR c\_1 IS SELECT \* FROM tab1 ORDER BY c2;

v\_1 c\_1%ROWTYPE;

BEGIN

OPEN c\_1;

LOOP

SELECT c\_1 INTO v\_1;

EXIT WHEN

(c\_1%NOTFOUND = TRUE OR c\_1%ROWCOUNT = 2);

DBMS\_OUTPUT.PUT\_LINE(v\_1.c2);

END LOOP;

CLOSE c\_1;

END;

Why will the above block cause a syntax error?

A. The ROWTYPE% attribute can only be used in reference to actual tables.

B. The variable v\_1 must be declared of the cursor’s %TYPE rather than %ROWTYPE.

C. The EXIT WHEN statement is illegal.

D. The SELECT-INTO statement is illegal.

Q10. What is the value of v\_flag when the following PL/SQL block is executed successfully?

DECLARE

v\_flag BOOLEAN;

v\_n NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_n

FROM tab1;

v\_flag := SQL%NOTFOUND;

END;

A. The value is always TRUE.

B. The value is always FALSE.

C. The value is always NULL.

D. The value is TRUE if and only if the tab1 table is empty.

E. The value is FALSE if and only if the tab1 table is empty.

F. The value is NULL if and only if the tab1 table is empty.

Q11. To which of the following will an exception raised in the DECLARATIONsection of the block B\_3 propagate?

<<B\_4>>

DECLARE

v\_4 NUMBER;

v\_5 NUMBER;

BEGIN

v\_4 := 10;

v\_5 := v\_4 + 80;

<<B\_3>>

DECLARE

v\_3 NUMBER := 90;

v\_4 NUMBER := v\_3 / (v\_3 - v\_5);

-- Run-time error, propagate to?

BEGIN

v\_3 := v\_3 \* v\_4 + 1;

v\_4 := v\_3 + v\_4;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (B\_3)!');

END B\_3;

v\_4 := v\_4 \* v\_4;

<<B\_2>>

DECLARE

v\_2 NUMBER := 0;

v\_5 NUMBER := v\_4;

BEGIN

……

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (B\_2)!');

END B\_2;

<<B\_1>>

DECLARE

v\_1 NUMBER := 3;

BEGIN

……

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (B\_1)!');

END B\_1;

……

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (B\_4)!');

END B\_4;

A1. The B\_1 block’s DECLARATION section

A2. The B\_1 block’s EXECUTION section

A3. The B\_1 block’s EXCEPTION section

B1. The B\_2 block’s DECLARATION section

B2. The B\_2 block’s EXECUTION section

B3. The B\_2 block’s EXCEPTION section

C1. The B\_3 block’s DECLARATION section

C2. The B\_3 block’s EXECUTION section

C3. The B\_3 block’s EXCEPTION section

D1. The B\_4 block’s DECLARATION section

D2. The B\_4 block’s EXECUTION section

D3. The B\_4 block’s EXCEPTION section

E. None of the above

Q12. How many rows will be inserted into the tab1 table after the following PL/SQL block has been executed successfully (no runtime error)?

DECLARE

v\_count NUMBER := 1;

v\_1 NUMBER := 1;

BEGIN

DELETE FROM tab1;

COMMIT;

FOR i IN REVERSE 5..8 LOOP

INSERT INTO tab1 VALUES (i, i\*2, i+3);

END LOOP;

SELECT COUNT(\*)

INTO v\_count

FROM tab1;

FOR i IN 2..v\_count + 10 LOOP

INSERT INTO tab1 VALUES (i, i+10, i+20);

END LOOP;

INSERT INTO tab1 VALUES (99, 205, 306);

SELECT COUNT(\*)

INTO v\_1

FROM tab1;

WHILE v\_1 >= 7 LOOP

IF v\_1 = 10 OR v\_1 = 25 OR v\_1 = 35 OR v\_1 = 45 THEN

INSERT INTO tab1 VALUES (v\_1\*20, v\_1\*30, v\_1\*40);

ELSE

v\_1 := v\_1 - 1;

END IF;

v\_1 := v\_1 - 2;

END LOOP;

FOR i IN 1..50 LOOP

IF i = 4 OR i = 16 OR i = 47 THEN

INSERT INTO tab1 VALUES (i\*20, i\*30, i\*40);

END IF;

INSERT INTO tab1 VALUES (i\*21, i\*31, i\*41);

END LOOP;

INSERT INTO tab1 VALUES (616, 222, 243);

INSERT INTO tab1 VALUES (77, 88, 99);

COMMIT;

END;

A. 72

B. 73

C. 74

D. 75

E. 76

F. None of the above

**Part II (CSC 352 and CSC 452 - 180 points)**

* You are not allowed to create/use temporary tables/views/functions/procedures/triggers.
* If you modified the DEPARTMENT and EMPLOYEE tables created in Assignment #1, you need to delete and re-populate them.
* The Exception section in your program is optional.
* PL/SQL collections are not required.

**1) (CSC 352 and CSC 452 - 60 points)**

The BIRTHDAY\_DISTRIBUTION table consists of every day of the year, from January 1 to December 31, along with a ranking based on how many babies were born in the United States on that date between 1973 and 1999. Rank 1 is the most popular, rank 2 is the next most popular, and so forth.

Create and populate the BIRTHDAY\_DISTRIBUTION table by using the following SQL statements. (You have to connect to CDM’s Oracle server to populate the birthday\_ distribution table.)

CREATE TABLE birthday\_distribution

(MONTH NUMBER,

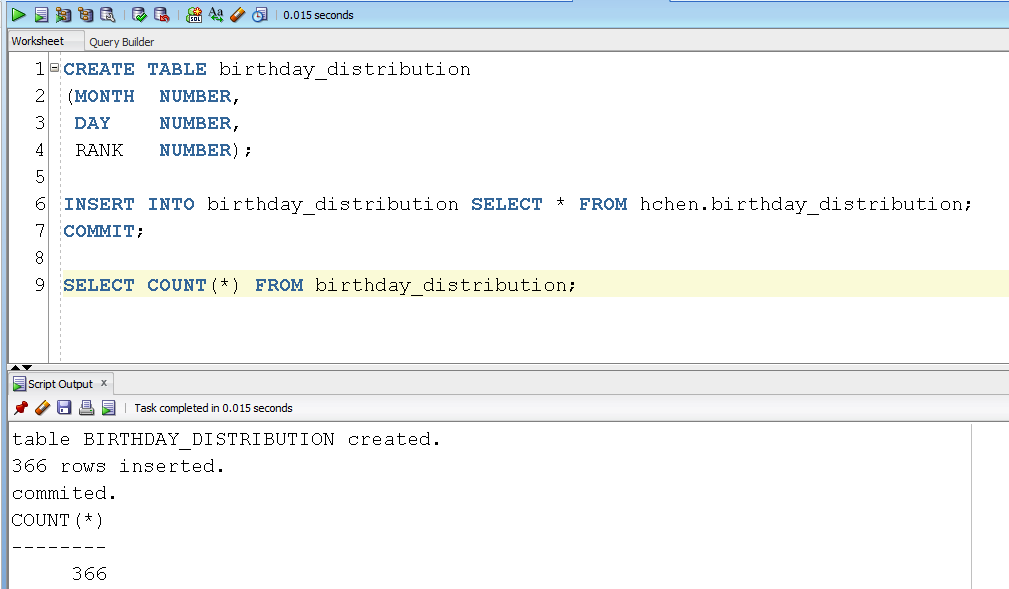
DAY NUMBER,

RANK NUMBER);

INSERT INTO birthday\_distribution SELECT \* FROM **hchen**.birthday\_distribution;

COMMIT;

SELECT COUNT(\*) FROM birthday\_distribution;



Please make sure that there are 366 rows in your BIRTHDAY\_DISTRIBUTION table.

In the BIRTHDAY\_DISTRIBUTION table, you can find that September 16 is the most popular birthday (rank = 1) and February 29 is the least popular birthday (rank = 366). Excluding leap years, December 25 is the least popular birthday (rank = 365).

**========================= Begin (1a) CSC 352 only ========================**

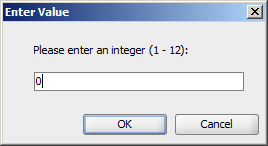
**1a) (CSC 352 only)**

Write a PL/SQL anonymous block that accepts an integer for a month (1 ≤ *m* ≤ 12) from the user input and displays all days along with the ranks of the month. The most popular birthday of the month is marked with “\*\*\* Most Popular Day \*\*\*”. The least popular birthday of the month is marked with “\*\*\* Least Popular Day \*\*\*”. Sort your output in ascending order by days.

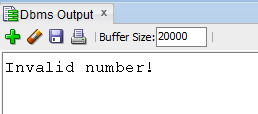
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output has an incorrect format. For example, you must display the month, day, rank and “\*\*\* Most Popular Day \*\*\*” in one line.
* To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter.
* Submitting more than **one** PL/SQL program will receive 0 points.

Test your program. You must ensure that the output of your program matches the following output:

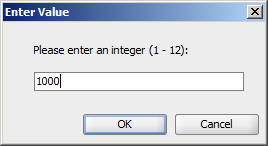
Case 1)



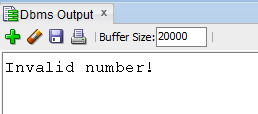
Output:



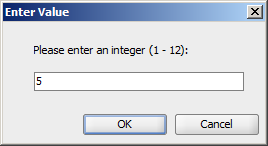
Case 2)



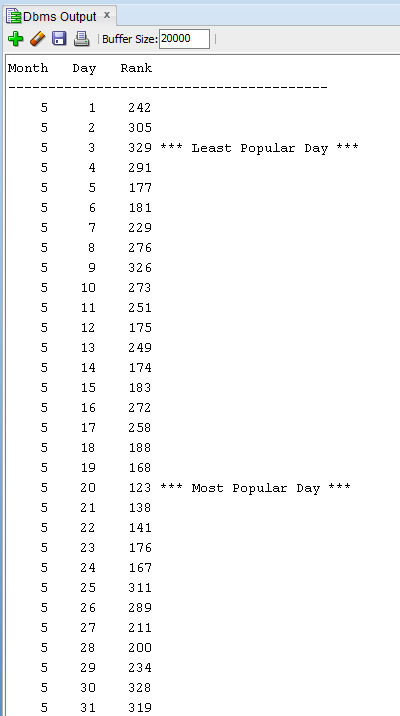
Output:



Case 3)



Output:



**========================== End (1a) CSC 352 only ========================**

**========================== Begin (1b) CSC 452 only =======================**

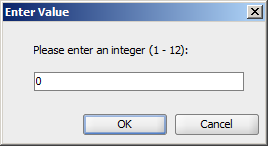
**1b) (CSC 452 only)**

Write a PL/SQL anonymous block that accepts an integer for a month (1 ≤ *m* ≤ 12) from the user input and displays all days along with the ranks of the month. The two (2) most popular birthdays of the month are marked with “\*\*\* 1st Most Popular Day” and “\*\*\* 2nd Most Popular Day”, respectively. The two (2) least popular birthdays of the month are marked with “\*\*\* 1st Least Popular Day” and “\*\*\* 2nd Least Popular Day”, respectively. Sort your output in ascending order by days.

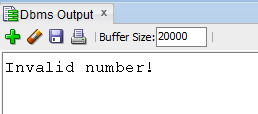
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output has an incorrect format. For example, you must display the day, rank and “(1st Most Popular Day)” in one line.
* To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter. But, you need to check whether the user input is between 1 and 12.
* Submitting more than **one** PL/SQL program will receive 0 points.

Test your program. You must ensure that the output of your program matches the following output:

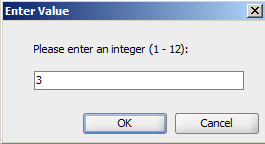
Case 1



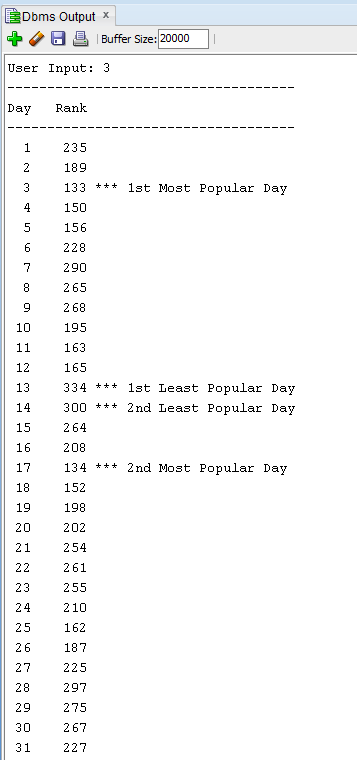
Output:



Case 2)



Output:



**========================== End (1b) CSC 452 only =======================**

**2) (CSC 352 and CSC 452 - 60 points)**

ABC Airlines Inc. keeps track of its employees in its Human Resources database. The HR\_PERSON table contains basic employee information. The JOB\_TYPE field indicates whether a person is a full-time employee (FT) or a part-time employee (PT). The JOB\_STATUS field indicates whether a person is an active (A) or inactive (I) employee. The structure of the table is shown below along with some sample records:

**HR\_PERSON**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PERSON\_ID** | **LAST\_NAME** | **FIRST\_NAME** | **HIRE\_DATE** | **JOB\_TYPE** | **JOB\_STATUS** |
| **NUMBER PK** | **VARCHAR2(30)** | **VARCHAR2(30)** | **DATE** | **VARCHAR2(10)** | **CHAR** |
| 1000 | Smith | Ryan | 04-MAY-90 | FT | I |
| 1170 | Brown | Dean | 01-DEC-92 | PT | A |
| 2010 | Fisher | Jane | 12-FEB-95 | FT | I |
| 2080 | Brewster | Andre | 28-JUL-98 | FT | A |
| 3190 | Clark | Dan | 04-APR-01 | PT | A |
| 3500 | Jackson | Tyler | 01-NOV-05 | FT | A |
| 4000 | Miller | Mary | 11-JAN-08 | FT | A |
| 4100 | Jackson | Peter | 08-AUG-11 | PT | I |
| 4200 | Smith | Ryan | 08-DEC-12 | FT | A |

An employee (full-time employee as well as part-time employee) can be a pilot. In this case, the information related to pilots is kept inside a separate table HR\_PILOT as shown below:

**HR\_PILOT**

|  |  |
| --- | --- |
| **PERSON\_ID** | **PILOT\_TYPE** |
| **NUMBER PK, FK** | **VARCHAR2(100)** |
| 1170 | Commercial pilot |
| 2010 | Airline transport pilot |
| 3500 | Airline transport pilot |

For example, Brewster Andre is a full-time employee and is not a pilot while Tyler Jackson is a full-time employee and is also a pilot. On the other hand, Dean Brown is a part-time employee and is also a pilot. However, Dan Clark is a part-time employee but is not a pilot.

Create and populate the HR\_PERSON and HR\_PILOT tables by using the following SQL statements.

CREATE TABLE hr\_person

(

person\_id NUMBER PRIMARY KEY,

last\_name VARCHAR2(30) NOT NULL,

first\_name VARCHAR2(30) NOT NULL,

hire\_date VARCHAR2(30) NOT NULL,

job\_type varchar2(10) NOT NULL,

job\_status CHAR NOT NULL

);

/

INSERT INTO hr\_person VALUES (1000, 'Smith', 'Ryan', '04-MAY-90','FT', 'I');

INSERT INTO hr\_person VALUES (1170, 'Brown', 'Dean', '01-DEC-92','PT', 'A');

INSERT INTO hr\_person VALUES (2010, 'Fisher', 'Jane', '12-FEB-95','FT', 'I');

INSERT INTO hr\_person VALUES (2080, 'Brewster', 'Andre', '28-JUL-98', 'FT', 'A');

INSERT INTO hr\_person VALUES (3190, 'Clark', 'Dan', '04-APR-01','PT', 'A');

INSERT INTO hr\_person VALUES (3500, 'Jackson', 'Tyler', '01-NOV-05', 'FT', 'A');

INSERT INTO hr\_person VALUES (4000, 'Miller', 'Mary', '11-JAN-08', 'FT', 'A');

INSERT INTO hr\_person VALUES (4100, 'Jackson', 'Peter', '08-AUG-11', 'PT','I');

INSERT INTO hr\_person VALUES (4200, 'Smith', 'Ryan', '08-DEC-12', 'FT','A');

COMMIT;

/

CREATE TABLE hr\_pilot

(

person\_id NUMBER PRIMARY KEY,

pilot\_type VARCHAR2(100) NOT NULL,

CONSTRAINT fk\_hr\_person\_pilot FOREIGN KEY (person\_id)

REFERENCES hr\_person(person\_id)

);

/

INSERT INTO hr\_pilot VALUES (1170, 'Commercial pilot');

INSERT INTO hr\_pilot VALUES (2010, 'Airline transport pilot');

INSERT INTO hr\_pilot VALUES (3500, 'Airline transport pilot');

COMMIT;

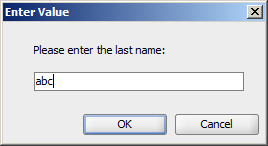
/

Write a **PL/SQL anonymous block** that accepts a **last name** (LAST\_NAME) from the user input and displays employee’s information (job status, employee name, job type, hire date, and pilot type). Sort your output in ascending order by the job status (Active, Inactive), employee name (Last Name, First Name), job type (Full-Time, Part-Time), and hire date.

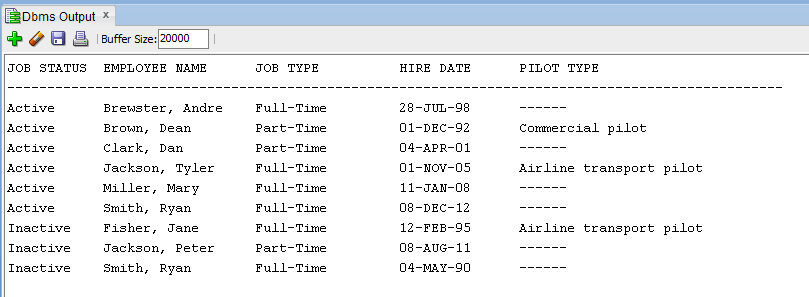
* If the last name is **NOT** in the HR\_PERSON table (LAST\_NAME), your program displays information about **ALL** employees.
* If the last name is in the HR\_PERSON table (LAST\_NAME), your program displays the corresponding employee’s information. We have duplicate names in the HR\_PERSON table.
* The last name is not case sensitive (e.g., Jackson = JACKSON). You will lose 10 points if you do not use the UPPER (or LOWER) function in your program.
* The job type (“FT” or “PT”) must be displayed as “Full-Time” or “Part-Time” in your output. You will lose 10 points if you fail to do so. (Hint: you can use IF…THEN…ELSE/CASE/DECODE statement/function to convert one string to another.)
* The status (“A” or “I”) must be displayed as “Active” or “Inactive” in your output. You will lose 10 points if you fail to do so. (Hint: you can use IF…THEN… ELSE/CASE/DECODE statement/function to convert one string to another.)
* If an employee is not a pilot, the pilot type is shown as “------” in your output.
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output has an incorrect format.
* Using different table/column names will receive 0 points.
* Submitting more than **one** PL/SQL program will receive 0 points.

Test your program. You must ensure that the output of your program matches the following sample output:

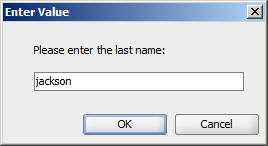
Case 1)



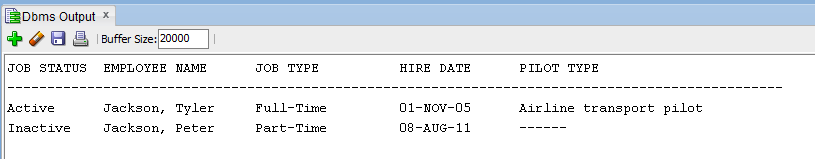
Output:



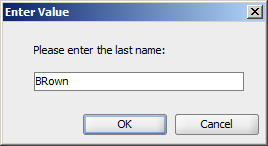
Case 2)



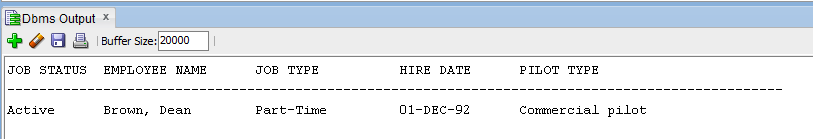
Output:



Case 3)



Output:



**3) (CSC 352 and CSC 452 - 60 points)**

**========================= Begin (3a) CSC 352 only ========================**

**3a) (CSC 352 only)**

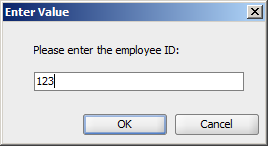
Based on the tables created in Assignment #1, write a **PL/SQL program** that accepts an employee ID from the user input and displays 1) employee name, hire date, salary, commission, total pay (salary + commission), and his/her department name (If the given employee does not belong to any department, the department name is shown as “\*\*\*\*\*\*” in your output.), and 2) all employees (alone with their hire dates) who work in the same department as the given employee **and** were hired **before** the given employee (or “NO OUTPUT”). Sort your output by the employee name.

* You must display salary/commission/total pay with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56). If the employee does not have a commission, the commission must be shown as $0.00 in your output.
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output has an incorrect format.
* Hard coding (e.g., IF v\_emp\_id = 7596 THEN v\_1 := ...) will receive a zero grade.
* Submitting more than **one** PL/SQL program will receive 0 points.

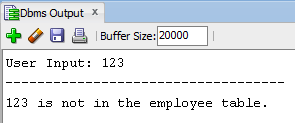
To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter.

Test your program. You must ensure that the output of your program matches the following sample output:

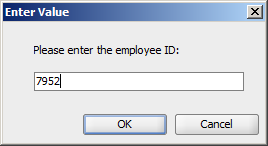
Case 1)



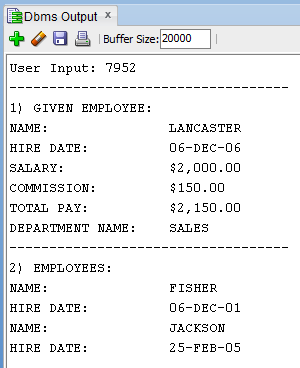
Output:



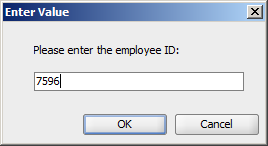
Case 2)



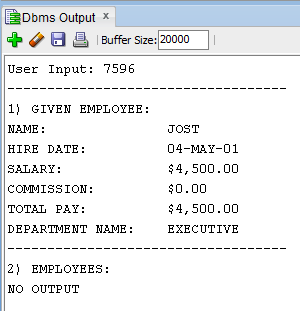
Output:



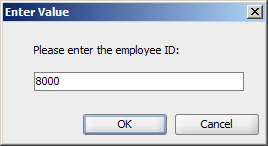
Case 3)



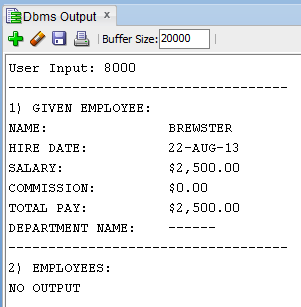
Output:



Case 4)



Output:



**========================= End (3a) CSC 352 only =========================**

**========================= Begin (3b) CSC 452 only ========================**

**3b) (CSC 452 only)**

Based on the tables created in Assignment #1, write a **PL/SQL anonymous block** that displays all employees who were hired on the days of the week on which the **highest** number of employees were hired. The output of the program must contain all the hire dates, employee names, job, their corresponding department names (If an employee does not belong to any department, the department name is shown as “------” in your output.), and their corresponding manager names (If an employee does not have a manager, the manager name is shown as “------” in your output.). Sort your output by days of the week (Monday, Tuesday, …, Friday) and the hire date.

* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output has an incorrect format.
* Hard coding (e.g., IF v\_day = 'Thursday' OR v\_day = 'Friday' OR v\_max\_num = 4 THEN …) will receive a zero grade.
* Submitting more than **one** PL/SQL program will receive 0 points.

Hints:

(1) TO\_CHAR(hire\_date, 'Day')

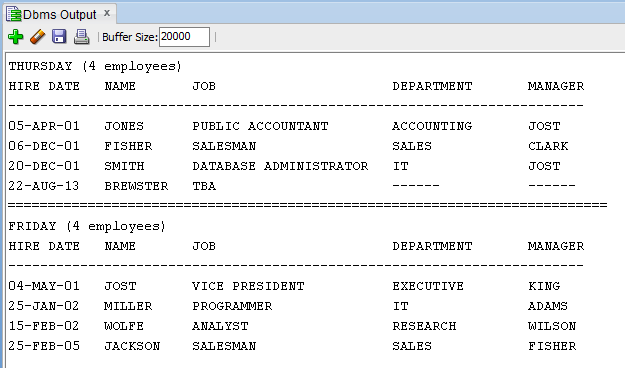
(2) TRIM(TO\_CHAR(hire\_date, 'Day'))

(3) TRIM(TO\_CHAR(hire\_date, 'D')

(4) GROUP BY TO\_CHAR(hire\_date, 'Day')

(5) HW1 (Part III)

**The output of your program must match the following:**



**========================= End (3b) CSC 452 only ========================**