CSC 352/452: Database Programming

Mid-term Exam   
(300 Points)

**CSC 352/452-501: Due on Sunday, 8/6/2017 at 11:59PM**

**CSC 352/452-510: Due on Monday, 8/7/2017 at 11:59PM**

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

**(No lecture and office hours on Thursday, 8/3/2017)**

* **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 by the 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 do not include the original questions in your file.**
* **Please review your assignment file before submitting it to make sure you have the correct one. It is your responsibility to upload the correct assignment file.**
* **Do not wait until the last day to submit your file! Submit ahead in case of unforeseen circumstances such as a computer crash or illness. There are no exceptions to the deadline.**

**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.
* Each question has one correct answer. You will receive 0 points if you submit more than one answer in any format (e.g., Q5: A, False).
* Please submit your answers in the following format:

Part I: Q1: A, Q2: B, Q3: C, …

**Please do not include the original questions.**

*(This part should take less than 20 minutes to finish.)*

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

Q1. Consider the following SELECT-INTO statement in a PL/SQL block. What happens if there is **NO** row satisfying the WHERE condition?

......

SELECT COUNT(\*)

INTO v\_n

FROM tab1

WHERE col\_15 > 10 OR col\_20 > 10;

......

A. The SELECT-INTO statement executes successfully.

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

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

D. A ZERO\_DIVIDE exception is raised.

Q2. In which section of a PL/SQL block is a WHEN TOO\_MANY\_ROWS THEN statement allowed (The TOO\_MANY\_ROWS is a predefined exception for ORA-01422)?

A. DECLARATION

B. EXECUTION

C. EXCEPTION

D. All of the above

Q3. Evaluate the following CASE statement:

CASE v\_input

WHEN 10 THEN v\_out := 100; v\_input := 30;

WHEN 20 THEN v\_out := 202; v\_input := 40;

WHEN 30 THEN v\_out := 606; v\_input := 50;

WHEN 40 THEN v\_out := 410; v\_input := 60;

WHEN 50 THEN v\_out := 881;

ELSE v\_out := 991;

END CASE;

If v\_input is 20, which value would be assigned to v\_out?

A. 100

B. 202

C. 410

D. 606

E. 881

F. 991

G. None of the above

Q4. In the DECLARATION section of a PL/SQL block, you declare these variables:

v\_room\_100, v\_room\_200 VARCHAR2(15);

Why does this statement cause an error?

A. The NOT NULL keyword is missing.

B. The size (15) of the data type VARCHAR2 cannot be specified.

C. Multiple variables cannot be declared in the same statement.

D. A default value must be assigned to each variable.

E. All of the above.

Q5. PL/SQL records of the same declared type can be compared for equality by using the equality operator (=).

DECLARE

TYPE t\_type IS RECORD (p\_id NUMBER, p\_age NUMBER);

v\_a t\_type;

v\_b t\_type;

BEGIN

......

IF v\_a = v\_b THEN

DBMS\_OUTPUT.PUT\_LINE('v\_a = v\_b');

ELSE

DBMS\_OUTPUT.PUT\_LINE('v\_a != v\_b');

END IF;

END;

A. TRUE

B. FALSE

Q6. What is the value of v\_found\_flag when the following PL/SQL block is executed successfully?

DECLARE

v\_n NUMBER;

v\_found\_flag BOOLEAN;

BEGIN

SELECT COUNT(\*)

INTO v\_n

FROM tab1

WHERE col\_99 > 0;

v\_found\_flag := SQL%FOUND;

END;

A. The value is always NULL.

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

C. The value is NULL if and only if the tab1 table is not empty.

D. The value is always FALSE.

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

F. The value is FALSE if and only if the tab1 table is not empty.

G. The value is always TRUE.

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

I. The value is TRUE if and only if the tab1 table is not empty.

Q7. Which guideline relates to a CURSOR FOR Loop?

FOR v\_idx IN cursor\_name LOOP

statement1;

statement2;

......

END LOOP;

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

B. The cursor must return at least one row.

C. It does not require a FETCH statement.

D. All of the above

Q8. Evaluate the following CURSOR statement:

DECLARE

CURSOR c\_1 (p\_max\_num NUMBER(3) := 500) IS

SELECT col\_1, col\_2, col\_3

FROM tab1

WHERE col1\_5 = p\_max\_num;

Why will this statement cause an error?

A. The default value (500) cannot be assigned to the p\_max\_num parameter.

B. The size (3) of the p\_max\_num parameter cannot be specified.

C. The SELECT statement is missing the INTO clause.

D. All of the above.

Q9. In a PL/SQL block, when a variable is declared as NOT NULL, you must initialize the variable when it is declared.

A. TRUE

B. FALSE

Q10. Evaluate the following PL/SQL block:

DECLARE

CURSOR c\_1 IS

SELECT col\_1, col\_3

FROM tab1

ORDER BY c1;

v\_rec c\_1%ROWTYPE;

BEGIN

OPEN c\_1;

LOOP

FETCH c\_1 INTO v\_rec;

EXIT WHEN v\_rec.col\_1 >= 100 OR

v\_rec.col\_1 <= -50 OR

c\_1%NOTFOUND;

......

DBMS\_OUTPUT.PUT\_LINE(v\_rec%ROWCOUNT);

DBMS\_OUTPUT.PUT\_LINE(v\_rec.col\_3);

......

END LOOP;

CLOSE c\_1;

END;

Why will the above block cause a syntax error?

A. The DBMS\_OUTPUT.PUT\_LINE(v\_rec.col\_3)statement is illegal.

B. The DBMS\_OUTPUT.PUT\_LINE(v\_rec%ROWCOUNT)statement is illegal.

C. The EXIT-WHEN statement is illegal.

D. The FETCH statement is illegal.

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

Q11. An exception will be raised in the DECLARATIONsection of the block BL\_20. To which of the following section will the exception propagate?

<<BL\_10>>

DECLARE

v\_4 NUMBER;

v\_5 NUMBER;

BEGIN

v\_4 := 10;

v\_5 := v\_4 + 80;

<<BL\_20>>

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 (BL\_20)!');

END BL\_20;

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

<<BL\_30>>

DECLARE

v\_2 NUMBER := 0;

v\_5 NUMBER := v\_4;

BEGIN

......

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (BL\_30)!');

END BL\_30;

<<BL\_40>>

DECLARE

v\_1 NUMBER := 3;

BEGIN

......

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (BL\_40)!');

END BL\_40;

......

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR (BL\_10)!');

END BL\_10;

A1. The BL\_10 block’s DECLARATION section

A2. The BL\_10 block’s EXECUTION section

A3. The BL\_10 block’s EXCEPTION section

B1. The BL\_20 block’s DECLARATION section

B2. The BL\_20 block’s EXECUTION section

B3. The BL\_20 block’s EXCEPTION section

C1. The BL\_30 block’s DECLARATION section

C2. The BL\_30 block’s EXECUTION section

C3. The BL\_30 block’s EXCEPTION section

D1. The BL\_40 block’s DECLARATION section

D2. The BL\_40 block’s EXECUTION section

D3. The BL\_40 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 6..11 LOOP

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

END LOOP;

SELECT COUNT(\*)

INTO v\_count

FROM tab1;

FOR i IN 3..v\_count + 11 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 2..48 LOOP

IF i = 4 OR i = 16 OR i= 32 OR i = 48 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);

INSERT INTO tab1 VALUES (77, 88, 55);

INSERT INTO tab1 VALUES (66, 88, 99);

INSERT INTO tab1 VALUES (66, 88, 91);

COMMIT;

END;

A. 75

B. 76

C. 77

D. 78

E. 79

F 80

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

**If there is a syntax error anywhere in your program, you will receive 0 points for the program.**

* 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.

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

XYZ Airlines Inc. keeps track of its employees in its Human Resources database. The TAB\_EMPLOYEE table contains basic employee information. The EMPLOYEE\_TYPE field indicates whether an employee is a full-time (FT) or part-time (PT) employee. The EMPLOYEE\_STATUS field indicates whether an employee is in an active (AC) or inactive (IN) work status. The structure of the table is shown below along with some sample records:

**TAB\_EMPLOYEE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **LAST\_NAME** | **FIRST\_NAME** | **HIRE\_DATE** | **EMPLOYEE\_TYPE** | **EMPLOYEE\_STATUS** |
| **VARCHAR2(7) PK** | **VARCHAR2(30)** | **VARCHAR2(30)** | **DATE** | **CHAR(2)** | **CHAR(2)** |
| 0A11000 | Smith | Ryan | 04-MAY-90 | FT | IN |
| 0A11170 | Brown | Dean | 01-DEC-92 | PT | AC |
| 0A12010 | Fisher | Jane | 12-FEB-95 | FT | IN |
| 0A12080 | Brewster | Andre | 28-JUL-98 | FT | AC |
| 0A13190 | Clark | Dan | 04-APR-01 | PT | AC |
| 0A13500 | Jackson | Tyler | 01-NOV-05 | FT | AC |
| 0A14000 | Miller | Mary | 11-JAN-08 | FT | AC |
| 0A14100 | Jackson | Peter | 08-AUG-11 | PT | IN |
| 0A14200 | Smith | Ryan | 08-DEC-12 | FT | AC |

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 TAB\_PILOT as shown below:

**TAB\_PILOT**

|  |  |
| --- | --- |
| **EMPLOYEE\_ID** | **PILOT\_TYPE** |
| **VARCHAR2(7) PK, FK** | **VARCHAR2(100)** |
| 0A11170 | Commercial pilot |
| 0A12010 | Airline transport pilot |
| 0A13500 | Airline transport pilot |

For example, Andre Brewster 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 TAB\_EMPLOYEE and TAB\_PILOT tables by using the following SQL statements.

CREATE TABLE TAB\_EMPLOYEE

(

employee\_id VARCHAR2(7) PRIMARY KEY,

last\_name VARCHAR2(30) NOT NULL,

first\_name VARCHAR2(30) NOT NULL,

hire\_date VARCHAR2(30) NOT NULL,

employee\_type CHAR(2) NOT NULL,

employee\_status CHAR(2) NOT NULL

);

/

INSERT INTO TAB\_EMPLOYEE VALUES ('0A11000', 'Smith', 'Ryan', '04-MAY-90','FT', 'IN');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A11170', 'Brown', 'Dean', '01-DEC-92','PT', 'AC');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A12010', 'Fisher', 'Jane', '12-FEB-95','FT', 'IN');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A12080', 'Brewster', 'Andre', '28-JUL-98','FT', 'AC');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A13190', 'Clark', 'Dan', '04-APR-01','PT', 'AC');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A13500', 'Jackson', 'Tyler', '01-NOV-05','FT', 'AC');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A14000', 'Miller', 'Mary', '11-JAN-08','FT', 'AC');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A14100', 'Jackson', 'Peter', '08-AUG-11','PT','IN');

INSERT INTO TAB\_EMPLOYEE VALUES ('0A14200', 'Smith', 'Ryan', '08-DEC-12','FT','AC');

COMMIT;

/

CREATE TABLE TAB\_PILOT

(

employee\_id VARCHAR2(7) PRIMARY KEY,

pilot\_type VARCHAR2(100) NOT NULL,

CONSTRAINT fk\_employee\_pilot FOREIGN KEY (employee\_id)

REFERENCES TAB\_EMPLOYEE(employee\_id)

);

/

INSERT INTO TAB\_PILOT VALUES ('0A11170', 'Commercial pilot');

INSERT INTO TAB\_PILOT VALUES ('0A12010', 'Airline transport pilot');

INSERT INTO TAB\_PILOT VALUES ('0A13500', 'Airline transport pilot');

COMMIT;

/

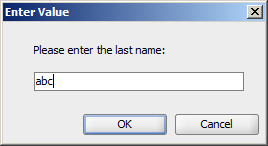
Write a **PL/SQL anonymous block** that accepts a **last name** (LAST\_NAME) from the user input and displays employee’s information (employee\_id, last\_name, first\_name, hire\_date, employee\_status, employee\_type, and pilot\_type). Sort your output in ascending order by the name (last\_name, first\_name) and hire\_date.

* If the last name is **NOT** in the TAB\_EMPLOYEE table (LAST\_NAME), your program displays information about **ALL** employees.
* If the last name is in the TAB\_EMPLOYEE table (LAST\_NAME), your program displays the corresponding employee’s information. We have duplicate names in the TAB\_EMPLOYEE table.
* The 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 employee\_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.
* The employee\_status (“AC” or “IN”) must be displayed as “Active” or “Inactive” in your output. You will lose 10 points if you fail to do so.
* If an employee is not a pilot, the pilot type is shown as “N/A” 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 is not in the correct format (e.g., wrong order of columns).
* You can only use the TAB\_EMPLOYEE and TAB\_PILOT tables. You will receive 0 points if you use other tables.
* You will receive 0 points if you submit more than **one** PL/SQL program.

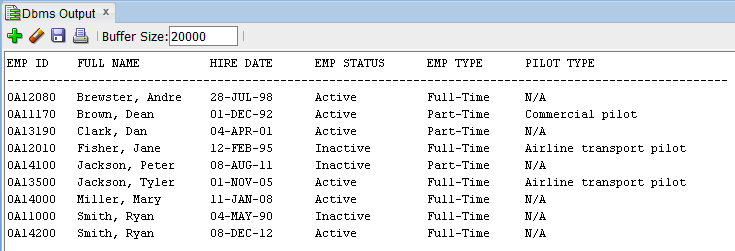
*(This question should take less than 45 minutes to solve.)*

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

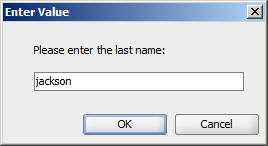
Case 1)



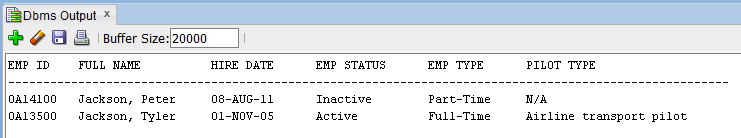
Output:



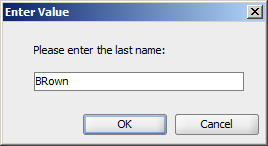
Case 2)



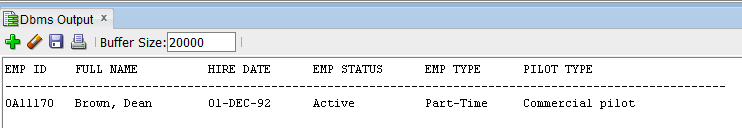
Output:



Case 3)



Output:



Case 4)

……

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

The TAB\_BIRTHDAY 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 TAB\_BIRTHDAY table by using the following SQL statements.

CREATE TABLE tab\_birthday

(BIRTHDAY\_MONTH NUMBER,

BIRTHDAY\_DAY NUMBER,

BIRTHDAY\_RANK NUMBER);

/

INSERT INTO tab\_birthday

SELECT \* FROM **hchen**.tab\_birthday;

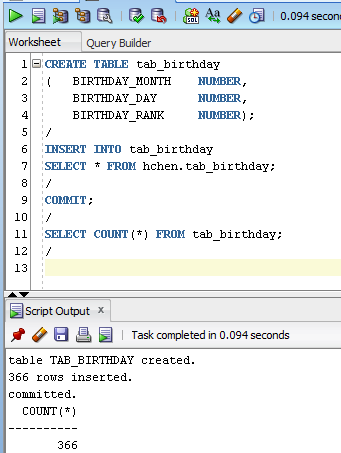
/

COMMIT;

/

SELECT COUNT(\*) FROM tab\_birthday;

/



Please make sure that there are 366 rows in your TAB\_BIRTHDAY table.

You cannot change the definition of the TAB\_BIRTHDAY table. You will receive 0 points if you use different table names, column names, or data types.

In the TAB\_BIRTHDAY 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 (2a) CSC 352 only ========================**

**2a) (CSC 352 only)**

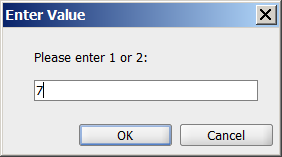
Write a PL/SQL anonymous block that accepts an integer ***n*** (***n = 1 or n = 2***) from the user input and 1) displays the most popular birthdays along with the ranks for each month if the user input is 1 (*n = 1*), or 2) displays the least popular birthdays along with the ranks for each month if the user input is 2 (*n = 2*). Sort your output in ascending order by months.

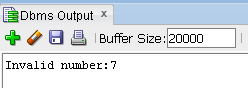
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output is not in the correct format. For example, you must display the birthdays and ranks for the same month in one line.
* Each day-rank pair must be displayed in the DD/RRR format (2-digit day and 3-digit rank) (e.g., 08/012).
* You may hard-corded values of months (e.g., FOR idx IN 1..12 LOOP).
* If you have hard coded the birthdays or ranks (e.g., DBMS\_OUPT.PUT\_LINE('1 20/ 240')) in your PL/SQL block, you 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.
* **This question can be solved without using cursors.**
* You will receive 0 points if you submit more than **one** PL/SQL program.

*(This question should take less than 45 minutes to solve.)*

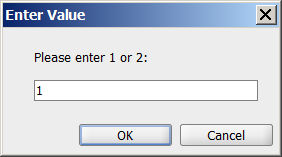
Test your program. You must ensure that the output of your program matches the following output (one month per line):

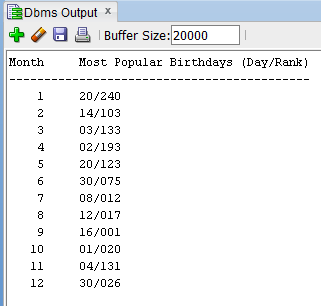
Case 1)



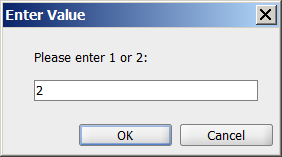


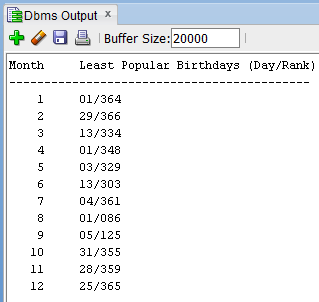
Case 2)





Case 3)





**========================== End (2a) CSC 352 only ========================**

**========================== Begin (2b) CSC 452 only =======================**

**2b) (CSC 452 only)**

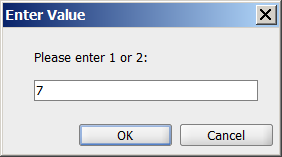
Write a PL/SQL anonymous block that accepts an integer ***n*** (***n = 1 or n = 2***) from the user input and 1) displays five (5) most popular birthdays along with the ranks for each month if the user input is 1 (*n = 1*), or 2) displays five (5) least popular birthdays along with the ranks for each month if the user input is 2 (*n = 2*). Sort your output in ascending order by months, and then most/least popular birthdays.

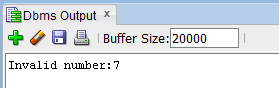
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output is not in the correct format. For example, you must display the birthdays and ranks for the same month in one line.
* Each day-rank pair must be displayed in the DD/RRR format (2-digit day and 3-digit rank) (e.g., 03/088).
* You may hard-corded values of months (e.g., FOR idx IN 1..12 LOOP).
* If you have hard coded the birthdays or ranks (e.g., DBMS\_OUPT.PUT\_LINE('1 <20/240> <14/260>')) in your PL/SQL block, you 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.
* **This question can be solved without using cursors.**
* You will receive 0 points if you submit more than **one** PL/SQL program.

*(This question should take less than 60 minutes to solve.)*

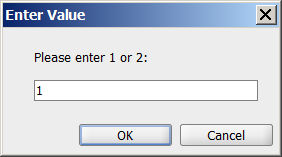
Test your program. You must ensure that the output of your program matches the following output (one month per line):

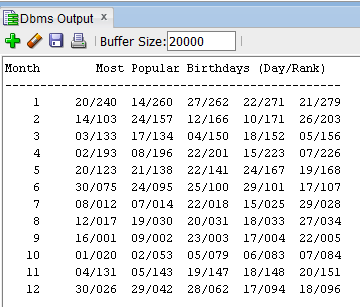
Case 1)



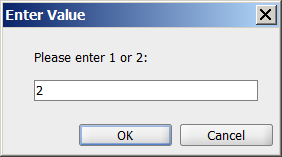


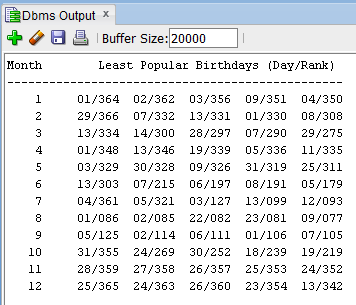
Case 2)





Case 3)





**========================== End (2b) CSC 452 only =======================**

**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) the employee's name, job title, hire date, salary, manager name (If the given employee does not have a manger, the manager name is shown as “------” in your output.), and 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 job titles, hire dates, and salary) 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.

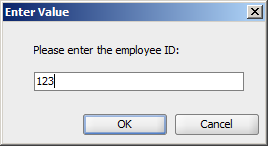
* Hard coding (e.g., IF v\_emp\_id = 7596 THEN v\_1 := ...) will receive 0 points.
* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output is not in the correct format.
* Your program must display the salary with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56).
* You will receive 0 points if you submit more than **one** PL/SQL program.

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.

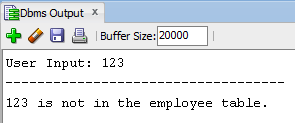
*(This question should take less than 60 minutes to solve.)*

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

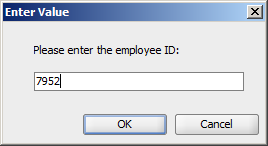
Case 1)



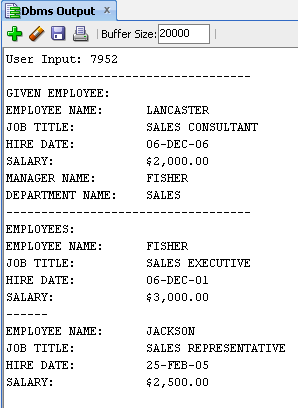
Output:



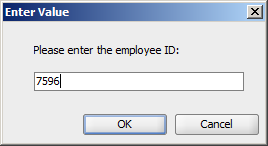
Case 2)



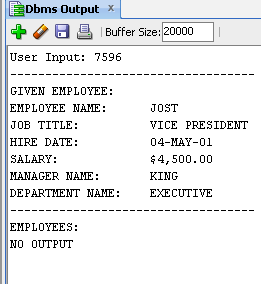
Output:



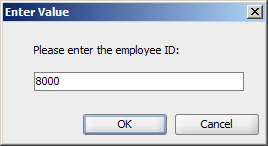
Case 3)



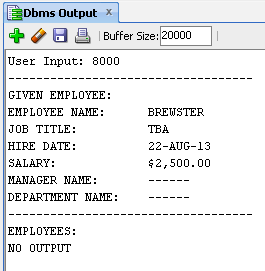
Output:



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 performs the following tasks.

a) Find all employees who were hired on the days of the week on which the **highest** number of employees were hired.

b) For each employee found in the previous step, display the following items in the output of your program:

* His/her hire date, name, job title, and salary.
* The total number of employees who report to him/her directly. (For example, you can find that there are 3 employees who report to Jost directly, Jones, Smith, and Wilson.)
* The department name that he/she works. If the employee does not belong to any department, the department name is shown as “------” in your output.
* His/her manager name and salary. If the employee does not have a manager, the manager name and salary are shown as “------” in your output.

Sort your output by days of the week (Monday, Tuesday, …, Friday), the hire date, and employee name.

* You will lose 10 points if the title lines are missing in your output.
* You will lose 10 points if your output is not in the correct format.
* Your program must display the salary with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56).
* Hard coding (e.g., IF v\_day = 'Thursday' OR v\_day = 'Friday' OR v\_max\_num = 4 THEN …) will receive 0 points.
* You will receive 0 points if you submit more than **one** PL/SQL program.

Hints:

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

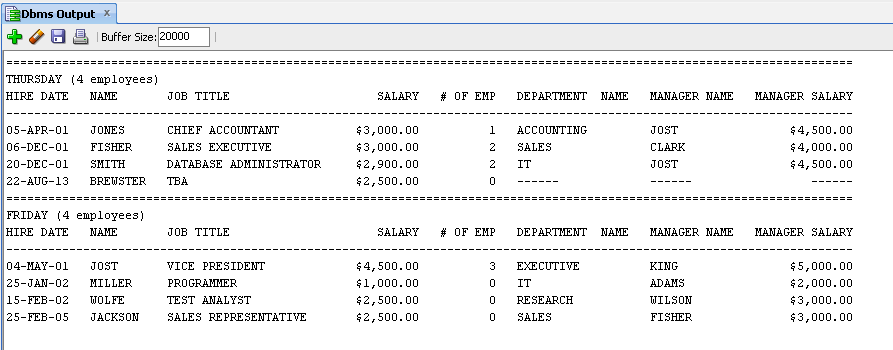
(2) TRIM(TO\_CHAR(hire\_date, 'Day'))

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

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

*(This question should take less than 60 minutes to solve.)*

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



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