CSC 352 / 452: Database Programming

assignment #3 (60 Points)

**CSC 352/452-501: Due on Sunday, 8/2/2015 at 11:59PM**

**CSC 352/452-510: Due on Monday, 8/3/2015 at 11:59PM**

Unless prior arrangements are made, homework turned in late but within 24 hours of the due time will be graded at 75% credit, homework turned in between 24 and 48 hours will be graded at 50% credit, and homework turned in later than 48 hours will not be accepted.

**Please note that only TEXT files will be accepted. 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.**

* If you modified the DEPARTMENT and EMPLOYEE tables created in Assignment #1, you need to delete and re-populate them.
* You cannot use hard-coded values (e.g., IF department\_name = 'MARKETING' THEN ……) in your programs.
* You are not allowed to create temporary tables, views, functions, or procedures.
* The EXCEPTION section is NOT allowed in your programs.
* 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.

**1) (CSC 352 – 30 Points | CSC 452 – 20 Points)**

The table popular\_names consists of the 20 most popular given names for male and female babies born during the years 1912-2011. For each rank and sex, the table has the name and the number of occurrences of that name. Rank 1 is the most popular, rank 2 is the next most popular, and so forth.

Create and populate the table popular\_names as described below.

CREATE TABLE popular\_names

( RANK NUMBER(3) PRIMARY KEY,

MALE\_GIVEN\_NAME VARCHAR2(20),

MALE\_NUMBER NUMBER,

FEMALE\_GIVEN\_NAME VARCHAR2(20),

FEMALE\_NUMBER NUMBER);

/

INSERT INTO popular\_names VALUES (1, 'James', 4877368, 'Mary', 3675303);

INSERT INTO popular\_names VALUES (10, 'Thomas', 2189104, 'Jessica', 1033915);

INSERT INTO popular\_names VALUES (8, 'Joseph', 2430380, 'Margaret', 1099230);

INSERT INTO popular\_names VALUES (4, 'Michael', 4246425, 'Jennifer', 1457441);

INSERT INTO popular\_names VALUES (7, 'Richard', 2517162, 'Susan', 1108187);

INSERT INTO popular\_names VALUES (5, 'William', 3758373, 'Linda', 1446686);

INSERT INTO popular\_names VALUES (19, 'Steven', 1263143, 'Ashley', 822403);

INSERT INTO popular\_names VALUES (20, 'Kenneth', 1251782, 'Kimberly', 818749);

INSERT INTO popular\_names VALUES (15, 'Anthony', 1354398, 'Lisa', 962672);

INSERT INTO popular\_names VALUES (16, 'Paul', 1345297, 'Helen', 872494);

INSERT INTO popular\_names VALUES (17, 'Mark', 1333807, 'Sandra', 871149);

INSERT INTO popular\_names VALUES (6, 'David', 3513288, 'Barbara', 1421809);

INSERT INTO popular\_names VALUES (2, 'John', 4771740, 'Patricia', 1566852);

INSERT INTO popular\_names VALUES (3, 'Robert', 4677163, 'Elizabeth',1459403);

INSERT INTO popular\_names VALUES (9, 'Charles', 2212351, 'Dorothy', 1036364);

INSERT INTO popular\_names VALUES (11, 'Christopher', 1959155, 'Sarah', 992164);

INSERT INTO popular\_names VALUES (12, 'Daniel', 1808296, 'Betty', 982783);

INSERT INTO popular\_names VALUES (13, 'Matthew', 1508790, 'Nancy', 982107);

INSERT INTO popular\_names VALUES (14, 'Donald', 1397379, 'Karen', 981212);

INSERT INTO popular\_names VALUES (18, 'George', 1299705, 'Donna', 827929);

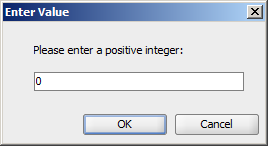
COMMIT;

Write a **PL/SQL anonymous block** that accepts a positive integer *n* from the user input and displays the *n* most popular given names for female and male and the number of occurrences of the corresponding name. Sort your output in **descending order** by ranks.

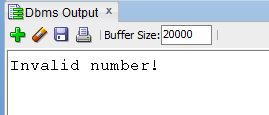
* Submitting more than **one** PL/SQL program will receive 0 points.
* To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter. But, you need to check whether the user input is a positive number or not.

Please test your program. The output of your program must match the following (ONE rank per line):

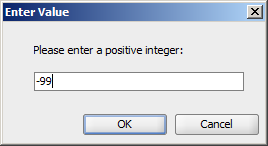
Case 1a)



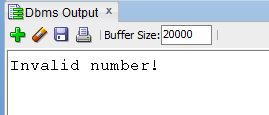
Output:



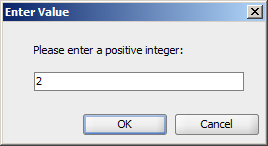
Case 1b)



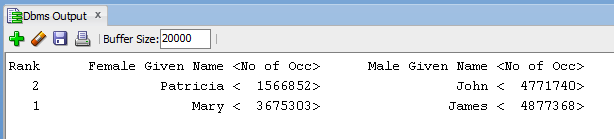
Output:



Case 2)

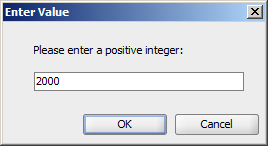


Output:

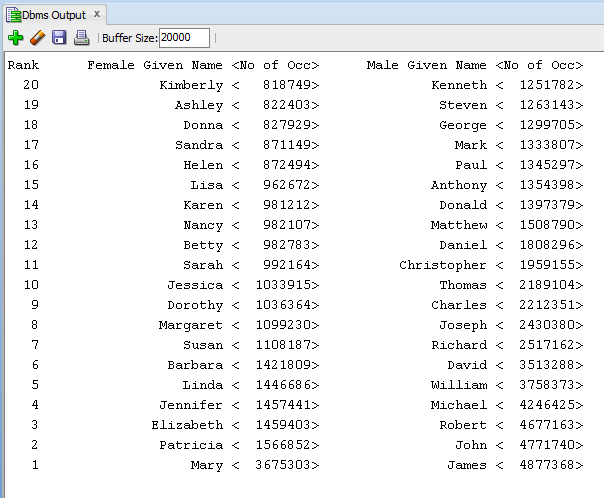


Case 3)

If the user input (*n*) is larger than the total number of rows in the table, your program display all rows in the table.



Output:



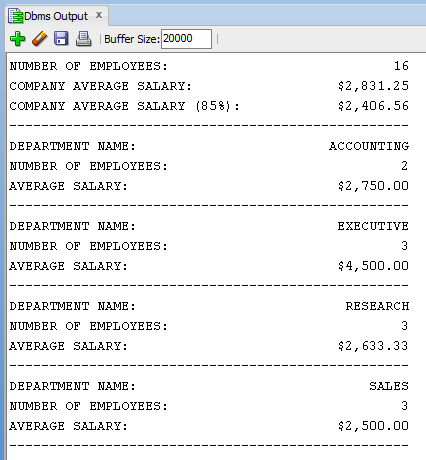
**2) (CSC 352 – 30 Points | CSC 452 – 20 Points)**

Based on the tables created in Assignment #1, write a **PL/SQL anonymous block** to perform the following tasks:

* Display the number of employees in the company;
* Display the average salary for the company;
* Display 85% of the average salary for the company;
* For each department whose average salary is **greater than 85%** of the average salary for the company, display the name of the department, the number of employees in the department, and the average salary for the department.
* You must display the average salary with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56).
* Sort your output in **ascending order** by department name.
* Submitting more than **one** PL/SQL program will receive 0 points.

Hint: IF v\_dept\_avg\_sal > 0.85 \* v\_comp\_avg\_sal THEN ……

**Please test your program. The output of your program must match the following:**



**3) (CSC 452 only – 20 points)**

The table AUTO\_SERVICE consists of auto repair and maintenance information. Create and populate the table AUTO\_SERVICE by using the following SQL statements.

CREATE TABLE auto\_service

( VIN VARCHAR2(18),

SERVICE\_ID VARCHAR2(10),

SERVICE\_DATE DATE,

DESCRIPTION VARCHAR2(50),

SERVICE\_PRICE NUMBER(7, 2),

CONSTRAINT pk\_auto\_service PRIMARY KEY (VIN, SERVICE\_ID));

/

INSERT INTO auto\_service VALUES('ZZZZ21', 'NAP0010189', '13-OCT-03', '---', 33.98);

INSERT INTO auto\_service VALUES('ZZZZ21', 'CHI0010123', '31-DEC-04', '---', 289.92);

INSERT INTO auto\_service VALUES('XYZ111', 'JOL0200080', '20-OCT-06', '---', 1230.23);

INSERT INTO auto\_service VALUES('ABC123', 'NAP1000890', '12-JAN-08', '---', 23.09);

INSERT INTO auto\_service VALUES('MMM789', 'NAP1000891', '12-JAN-08', '---', 44.01);

INSERT INTO auto\_service VALUES('ZZZZ21', 'AUR0002456', '12-FEB-08', '---', 1020.97);

INSERT INTO auto\_service VALUES('QWERT1', 'NAP1000990', '12-FEB-08', '---', 99.09);

INSERT INTO auto\_service VALUES('QWERT1', 'NAP1001890', '18-FEB-09', '---', 39.09);

INSERT INTO auto\_service VALUES('ABC123', 'CHI0018089', '12-OCT-09', '---', 46.00);

INSERT INTO auto\_service VALUES('ZXCV88', 'CHI0018189', '13-OCT-09', '---', 46.98);

INSERT INTO auto\_service VALUES('ZXCV88', 'JOL0300080', '20-MAR-11', '---', 123.25);

INSERT INTO auto\_service VALUES('ABC123', 'NAP1011123', '31-DEC-11', '---', 289.91);

INSERT INTO auto\_service VALUES('QWERT1', 'JOL0400090', '18-MAR-12', '---', 123.23);

INSERT INTO auto\_service VALUES('ZXCV88', 'NAP1014123', '31-DEC-12', '---', 289.90);

INSERT INTO auto\_service VALUES('ABC123', 'CHI0031199', '31-MAR-13', '---', 66.68);

INSERT INTO auto\_service VALUES('TAX999', 'CHI0031208', '31-MAR-13', '---', 20.91);

INSERT INTO auto\_service VALUES('WWW000', 'CHI0031298', '04-APR-13', '---', 1000.01);

INSERT INTO auto\_service VALUES('ZXCV88', 'AUR0700979', '04-APR-13', '---', 66.67);

INSERT INTO auto\_service VALUES('XYZ111', 'JOL0400080', '20-SEP-14', '---', 2200.10);

INSERT INTO auto\_service VALUES('WWW000', 'CHI0041299', '01-SEP-14', '---', 40.01);

COMMIT;

To avoid complicating issues, we assume that:

* For each VIN, there can be at most one service on any day.
* For each VIN, the service price is unique.

Write a **PL/SQL anonymous block** that reads **each VIN** in the AUTO\_SERVICE table and displays

1) the first service date and the service price for the first service,

2) the last service date and the service price for the last service,

3) the total number of services,

4) the total service prices for all services,

5) the average service prices for all services,

6) the minimum service price and the service date for the minimum service price, and

7) the maximum service price and the service date for the maximum service price.

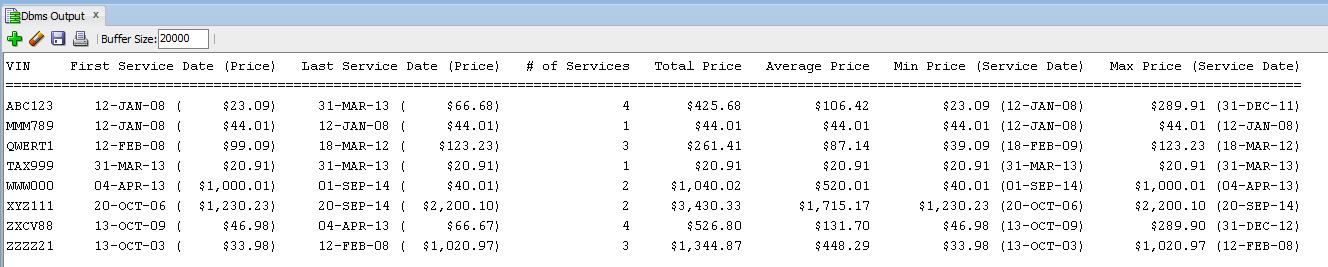
* You must display the service price with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56).
* Sort your output in **ascending order** by VIN.
* Submitting more than **one** PL/SQL program will receive 0 points.

Hints: First service date – MIN(SERVICE\_DATE)

Last service date – MAX(SERVICE\_DATE)

ORDER BY VIN

The output of your program must match the following (**one VIN per line**):



**Please submit a text file containing all the source codes to D2L before or on due date.**

**Optional Question**

**Just for fun (no credit, no extra credit, no need to submit, just for if you are a curious person and like database programming).**

In a bug tracking database, there is a table called BUGS. The table has several columns: BUG\_ID, REPORTED\_DATE, DESCRIPTION, PRIORITY, ASSIGNED\_TO, CLOSED\_DATE, and NOTE.

Create and populate the BUGS table.

CREATE TABLE bugs

(

BUG\_ID NUMBER PRIMARY KEY,

REPORTED\_DATE DATE NOT NULL,

DESCRIPTION VARCHAR2(20),

PRIORITY NUMBER(2),

ASSIGNED\_TO VARCHAR2(10),

CLOSED\_DATE DATE,

NOTE VARCHAR2(20)

);

INSERT INTO BUGS VALUES (1230, '25-APR-14', NULL, 3, 'Team 3', '28-APR-14', NULL);

INSERT INTO BUGS VALUES (1231, '29-APR-14', NULL, 1, 'Team 1', '29-APR-14', NULL);

INSERT INTO BUGS VALUES (1232, '03-MAY-14', NULL, 1, 'Team 1', '03-MAY-14', NULL);

INSERT INTO BUGS VALUES (1233, '03-MAY-14', NULL, 1, 'Team 3', '08-MAY-14', NULL);

INSERT INTO BUGS VALUES (1234, '04-MAY-14', NULL, 2, 'Team 5', '15-MAY-14', NULL);

INSERT INTO BUGS VALUES (1235, '04-MAY-14', NULL, 2, 'Team 1', NULL, NULL);

INSERT INTO BUGS VALUES (1236, '05-MAY-14', NULL, 1, 'Team 2', '06-MAY-14', NULL);

INSERT INTO BUGS VALUES (1237, '05-MAY-14', NULL, 3, 'Team 3', '10-MAY-14', NULL);

INSERT INTO BUGS VALUES (1238, '09-MAY-14', NULL, 4, 'Team 5', '16-MAY-14', NULL);

INSERT INTO BUGS VALUES (1239, '09-MAY-14', NULL, 5, 'Team 6', NULL, NULL);

INSERT INTO BUGS VALUES (1240, '12-MAY-14', NULL, 5, 'Team 2', '30-MAY-14', NULL);

INSERT INTO BUGS VALUES (1241, '12-MAY-14', NULL, 1, 'Team 1', '20-MAY-14', NULL);

INSERT INTO BUGS VALUES (1242, '13-MAY-14', NULL, 4, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1243, '14-MAY-14', NULL, 4, 'Team 3', '01-JUN-14', NULL);

INSERT INTO BUGS VALUES (1244, '14-MAY-14', NULL, 2, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1245, '20-MAY-14', NULL, 2, 'Team 4', NULL, NULL);

INSERT INTO BUGS VALUES (1246, '22-MAY-14', NULL, 2, 'Team 4', '25-MAY-14', NULL);

INSERT INTO BUGS VALUES (1247, '25-MAY-14', NULL, 2, 'Team 1', '29-MAY-14', NULL);

INSERT INTO BUGS VALUES (1248, '30-MAY-14', NULL, 1, 'Team 1', '01-JUN-14', NULL);

INSERT INTO BUGS VALUES (1249, '05-JUN-14', NULL, 1, 'Team 2', '07-JUN-14', NULL);

COMMIT;

“Open Bugs” - A bug is considered open on a given day if (1) its “REPORTED\_DATE” is on or before that day, and (2) its “CLOSED\_DATE” is on or after that day (or is unknown (NULL)). For example, we have 5 open bugs on 5/5/2014.

Write a **PL/SQL anonymous block** that generates a report to show the number of open bugs from 5/1/2014 through 5/31/2014. Sort your output by the number of open bugs in descending order and then by the date in ascending order. At the end of the report, the maximum number of open bugs on a single day is displayed. Assume that there were no open bugs on 4/30/2014-14.

The output of your program should match the following:

Number of Open Bugs Date

9 14-MAY-14

9 15-MAY-14

9 25-MAY-14

8 16-MAY-14

8 20-MAY-14

8 22-MAY-14

8 23-MAY-14

8 24-MAY-14

7 13-MAY-14

7 17-MAY-14

7 18-MAY-14

7 19-MAY-14

7 21-MAY-14

6 12-MAY-14

6 26-MAY-14

6 27-MAY-14

6 28-MAY-14

6 29-MAY-14

6 30-MAY-14

5 05-MAY-14

5 06-MAY-14

5 09-MAY-14

5 10-MAY-14

5 31-MAY-14

4 07-MAY-14

4 08-MAY-14

4 11-MAY-14

3 04-MAY-14

2 03-MAY-14

0 01-MAY-14

0 02-MAY-14

---------------------------------------

The maximum number of open bugs on a single day is 9.

There were 9 open bugs on 14-MAY-14.

There were 9 open bugs on 15-MAY-14.

There were 9 open bugs on 25-MAY-14.