CSC 352 / 452: Database Programming

assignment #3 (60 Points)

**CSC 352/452-501: Due on Sunday, 7/30/2017 at 11:59PM**

**CSC 352/452-510: Due on Monday, 7/31/2017 at 11:59PM**

Unless prior arrangements are made, homework turned in late will not be accepted. However, homework turned in within 24 hours late will be graded at 50% credit.

**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.
* **Do not try to use complicated joins or subqueries to get the results.** You can solve the questions quickly by using multiple PL/SQL statements.
* You are not allowed to create temporary tables, views, functions, or procedures.
* The EXCEPTION section is **NOT** allowed in your programs.
* Please review your file before submitting it to make sure you have the correct one. It is your responsibility to upload the correct file.

**1) (CSC 352 – 25 Points | CSC 452 – 20 Points)**

The table baby\_name consists of some popular given names for male and female babies born during the years 1915-2014. Rank 1 is the most popular, rank 2 is the next most popular, and so forth.

Create and populate the baby\_name table as described below.

CREATE TABLE baby\_name

( RANK NUMBER(3),

SEX CHAR CONSTRAINT ck\_sex CHECK (SEX in ('M', 'F')),

GIVEN\_NAME VARCHAR2(20) NOT NULL,

NUMBER\_OF\_OCCURRENCES NUMBER NOT NULL,

CONSTRAINT pk\_baby\_names PRIMARY KEY (RANK, SEX));

/

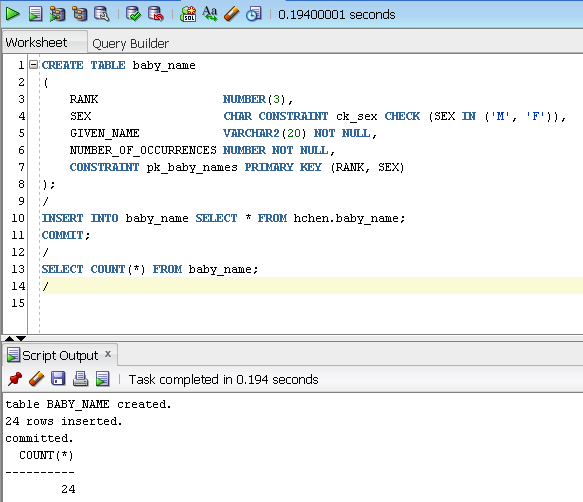
INSERT INTO baby\_name SELECT \* FROM **hchen**.baby\_name;

COMMIT;

/

SELECT COUNT(\*) FROM baby\_name;

/



Please make sure that there are 24 rows in your baby\_name table.

You cannot change the definition of the baby\_name table. You will receive 0 points if you use different tables.

Write a **PL/SQL anonymous block** that accepts a positive integer *n* from the user input and displays all rows (Sex, Rank, Given Name, and Number\_of\_Occurrences) having **rank ≤ *n***in the baby\_name table. Sort your output in **ascending order** by sex, and then rank.

The sex column (“M” or “F”) must be displayed as “Male” or “Female” in your output. You will lose 5 points if you fail to do so.

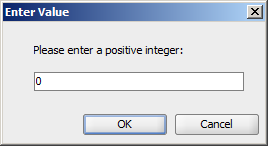
**CSC 452: your program displays “Male” and “Female” only once in the Sex column.**

To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9, plus (+) or minus (-) sign, and Enter. But, you need to check whether the user input is a positive number or not.

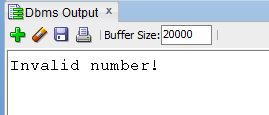
*(This question should take less than 40 minutes to solve.)*

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

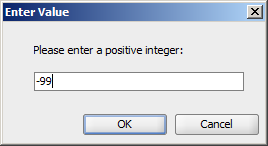
Case 1a)



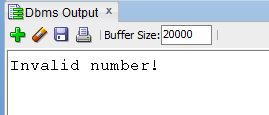
Output:



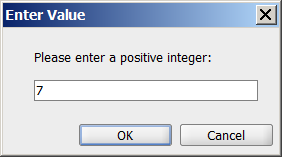
Case 1b)



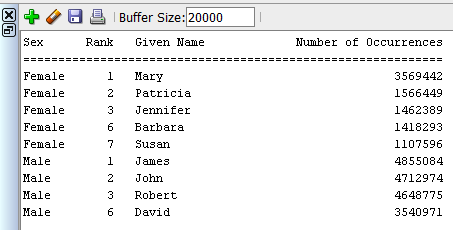
Output:



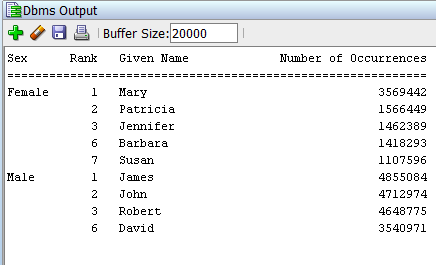
Case 2)



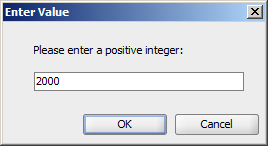
Output (CSC 352):



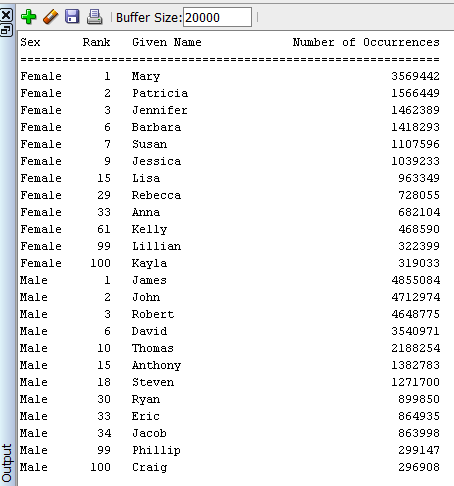
Output (CSC 452):



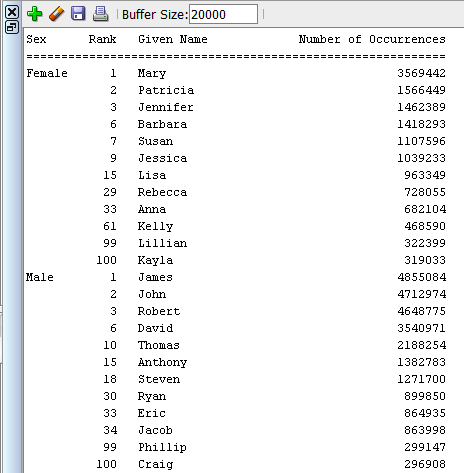
Case 3)



Output (CSC 352):



Output (CSC 452):



**2) (CSC 352 – 35 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 maximum/minimum/average total pay for the company.
* For each department, perform the following steps:

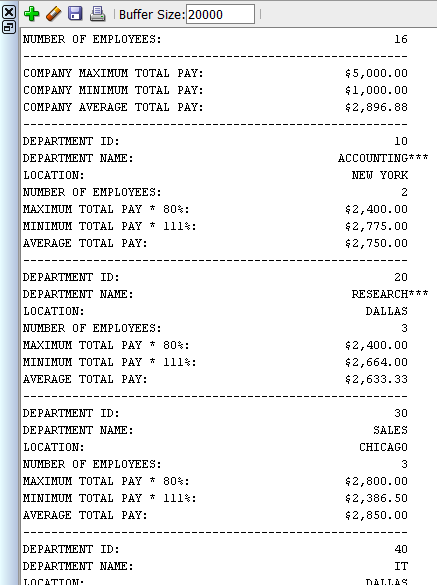
1. Display the department ID, name and location.
2. Display **80%** of the **maximum** total pay (maximum total pay \* 0.80) for the department.
3. Display **111%** of the **minimum** total pay (minimum total pay \* 1.11) for the department.
4. Display the average total pay for the department.
5. If the department does not have any employees, (b), (c) and (d) will be $0.00.
6. If the average total pay for the department is **greater than (b)** and **less than (c)**, the department name is marked with “\*\*\*”.
7. Sort your output in **ascending order** by department ID.

Total Pay = Salary + Commission.

You must display the maximum/minimum/average total pay with a dollar ($) sign, a comma, and two decimal places (e.g., $1,234.56). You will receive 0 points if you use hard-coded department names (e.g., Accounting, Research).

*(This question should take less than 50 minutes to solve.)*

**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 auto\_service table as described below.

CREATE TABLE auto\_service

(

SERVICE\_ID NUMBER(10) PRIMARY KEY,

VIN VARCHAR2(18) NOT NULL,

SERVICE\_DATE DATE NOT NULL,

SERVICE\_PRICE NUMBER(7, 2) NOT NULL,

NOTE VARCHAR2(200) NOT NULL

);

/

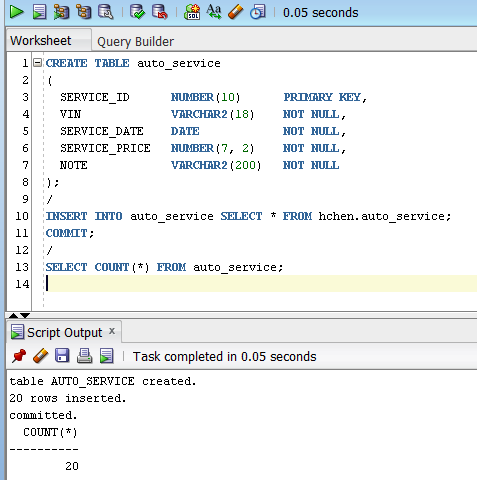
INSERT INTO auto\_service SELECT \* FROM **hchen**.auto\_service;

COMMIT;

/

SELECT COUNT(\*) FROM auto\_service;

Please make sure that there are 20 rows in your auto\_service table.



You cannot change the definition of the auto\_service table. You will receive 0 points if you use different tables.

*(A vehicle identification number, commonly abbreviated to VIN, is a unique code used by the automotive industry to identify individual motor vehicles.)*

To avoid complicating issues, 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 performs the following task:

* For **each VIN** in the auto\_service table, if 1) the **first** service date is in the period 02/01/2004 (01-FEB-04) to 03/31/2013 (31-MAR-13), 2) the **last** service date is in the period 03/01/2012 (01-MAR-12) to 03/31/2013 (31-MAR-13), and 3) the number of services is larger than 2, the following items along with the VIN are displayed:
* The total number of services
* The first service date and the service price for the first service
* The last service date and the service price for the last service
* The minimum service price and the service date for the minimum service price
* The maximum service price and the service date for the maximum service price

Hard coding the VIN being displayed (e.g., ABC123), the number of VIN being displayed (e.g., 2), or the number of services being displayed (e.g., 3, 4) will receive 0 points.

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.

*(This question should take less than 50 minutes to solve.)*

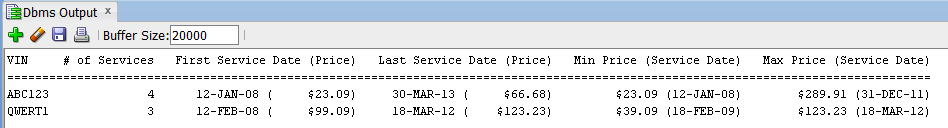
Hints: First service date – MIN(SERVICE\_DATE)

Last service date – MAX(SERVICE\_DATE)

Minimum service price– MIN(SERVICE\_PRICE)

Maximum service price– MAX(SERVICE\_PRICE)

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 by the 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.

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.