**CS 434 Fall 2023**

**HOMEWORK 1**

General Instructions.

Read this carefully.

This is more important than the individual questions.

There are NO GROUP HOMEWORKS IN THIS CLASS. YOU NEED TO WORK ALONE.

1) Do not copy code from any other person. You can ask questions and get answers, but NEVER copy code. Also, do not copy and paste English text unless I specifically permit it. NEVER.

2) SHOW EVERYTHING. **Anything you don't show will be automatically assumed as not done.** Don't logically argue that "of course, you must have done it." If we don't see it, then it is not done.

3) There will be no half points. Questions worth 1 point will be graded "all or nothing."

Points will appear in [ ] brackets.

The whole homework is worth 50 points.

If you miss the due date by up to one week, there will be a late penalty of 8 points subtracted.

If you miss the due date by MORE than a week you will get ZERO points.

The due date is  **Thursday October 5, 4:00 PM.**

**Your file name must have the format:**

**LASTNAME\_firstname\_HWK1.doc (or .docx)**

1. **ALL PL/SQL work needs to be done with SQL Developer. All database work needs to be done in Oracle.**

**a)** Write a function in PL/SQL called **MYPRIME** to check whether the number A is prime or not. **NO recursive, must be iterative.** The function should takeone IN parameter named A. The function does not send anything to the screen.

Copy/paste your PL/SQL code at the first red arrow and SNIP the screen output of the main program at the second red arrow. [3]

►

CREATE OR REPLACE FUNCTION MYPRIME(A IN NUMBER) RETURN BOOLEAN IS

i NUMBER := 2;

BEGIN

IF A <= 1 THEN

RETURN FALSE;

ELSIF A <= 3 THEN

RETURN TRUE;

END IF;

WHILE i \* i <= A LOOP

IF A MOD i = 0 THEN

RETURN FALSE;

END IF;

i := i + 1;

END LOOP;

RETURN TRUE;

END MYPRIME;

/

set serveroutput on

► A screenshot of a computer

Description automatically generated

**b)** Write **one** main program that calls your function MYPRIME with the following parameters:

A=99

A=97

A=17

A=83

The main program should send the results to the screen. The results should be self-documenting. It should **not** be just Yes/No.

The output should say something like this:

The number 99 is NOT a prime number.

The number 97 is a prime number.

Copy/paste your PL/SQL code at the first red arrow and SNIP the screen output of the main program at the second red arrow. [2]

►

DECLARE

numbers\_to\_test NUMBER := 99;

is\_prime BOOLEAN;

BEGIN

FOR i IN 1..4 LOOP

CASE i

WHEN 1 THEN numbers\_to\_test := 99;

WHEN 2 THEN numbers\_to\_test := 97;

WHEN 3 THEN numbers\_to\_test := 17;

WHEN 4 THEN numbers\_to\_test := 83;

END CASE;

is\_prime := MYPRIME(numbers\_to\_test);

IF is\_prime THEN

DBMS\_OUTPUT.PUT\_LINE('The number ' || numbers\_to\_test || ' is a prime number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The number ' || numbers\_to\_test || ' is NOT a prime number.');

END IF;

END LOOP;

END;

/

set serveroutput on

► A screenshot of a computer

Description automatically generated

**c)** A **composite number** is a positive integer that can be formed by multiplying two smaller positive integers. Equivalently, it is a positive integer that has at least one **divisor** other than 1 and itself.

Write a function in PL/SQL called **MYCOMPOSITE** to check if the number is a composite number or not. Your program should call MYPRIME.

Copy/paste your PL/SQL code at the first red arrow and SNIP the screen output of the main program at the second red arrow. [3]

►

CREATE OR REPLACE FUNCTION MYCOMPOSITE(A IN NUMBER) RETURN BOOLEAN IS

is\_prime BOOLEAN;

BEGIN

is\_prime := MYPRIME(A);

RETURN NOT is\_prime;

END MYCOMPOSITE;

/

set serveroutput on

► A screenshot of a computer

Description automatically generated

**d)** Call MYCOMPOSITE from a main program with n=99, n=97, n=17, and n=83. The main program should send the results to the screen. The results should be self-documenting.

The number 99 is a composite number, a product of 11 and 9.

The number 97 is not a composite number.

Copy/paste your PL/SQL code at the third red arrow and SNIP the screen output of the main program at the fourth red arrow. [2]

► DECLARE

numbers\_to\_test NUMBER := 0;

is\_composite BOOLEAN;

BEGIN

FOR i IN 1..4 LOOP

CASE i

WHEN 1 THEN numbers\_to\_test := 99;

WHEN 2 THEN numbers\_to\_test := 97;

WHEN 3 THEN numbers\_to\_test := 17;

WHEN 4 THEN numbers\_to\_test := 83;

END CASE;

is\_composite := MYCOMPOSITE(numbers\_to\_test);

IF is\_composite THEN

DBMS\_OUTPUT.PUT\_LINE('The number ' || numbers\_to\_test || ' is a composite number, a product of ' || numbers\_to\_test || ' and 1.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The number ' || numbers\_to\_test || ' is not a composite number.');

END IF;

END LOOP;

END;

/

set serveroutput on

► A screenshot of a computer

Description automatically generated

1. The goal of this problem is to load some health data from the government into our Oracle Database and use it to answer a few questions. (This will be continued in the next homework.)  
     
   Go to: <https://ephtracking.cdc.gov/>

Click on Explore Data

**STEP 1: CONTENT**

Click on “Select Content Area” so that you get “Chronic Obstructive Pulmonary Disease (COPD)”.

Choose “Mortality from COPD” on the second drop-down menu.

Choose “Crude Death Rate from COPD among people >= 25 years of age per 100,000 population” on the third drop-down menu.

**STEP 2: GEOGRAPHY TYPE**

National by State

**STEP 3: GEOGRAPHY**

All States.

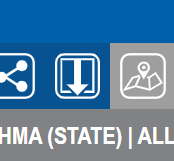
**STEP 4: TIME**

2020

**STEP 5: ADVANCED OPTIONS**

Race Ethnicity

All 4 Choices for Race

Then download the data and save it as a COPD.CSV file.   


Look at the Map. Note the menu that lets you choose between the four race choices.

Ask yourself these NON-CREDIT questions: What state has the highest value? What state has the lowest value? What does the number mean? Why do you think the state(s) with the highest value(s) have the highest value? Do you notice differences between the Race Choices? Why do you think those happen?

Even though these are non-credit questions, I would like you to attempt to write answers. If they are wrong, you don’t lose anything.

[5 points]

1. One of the most important steps when handling data (here and in Machine Learning) is to CLEAN the data. Look at the spreadsheet now.  
     
   We will do computation on the data. Is there anything in the data that will make computation impossible? Do you notice a pattern? NO CREDIT. But write the answer.  
     
   **Now Clean the Data as suggested in class.**
2. Load the Cleaned Data into an Oracle Table COPD1 using SQL Developer.   
   Write an SQL statement to display the COPD1 table.

Copy/paste your SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show only the first 10 rows. [5 points]

►

SELECT \*

FROM copd

FETCH FIRST 10 ROWS ONLY;

► A screenshot of a computer

Description automatically generated

1. Unfortunately, the values in Column D are percentages. If we want real numbers of COPD cases, then we need a second table that contains states and populations by state and by race.
2. I found such a table:  
     
   <https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_race/ethnicity>  
     
   Copy and paste the table into an EXCEL spreadsheet, starting from Alabama and ending with Wyoming.  
   Save it as a file POPULATION.CSV.   
     
   Delete all percentage columns.  
   Delete the Hispanic column (they are double-counted as race).

Check the first column's first character (make sure it is not a space character).

1. Load the resulting POPULATION.CSV file into Oracle using SQL/Developer. Call the table POPULATION1.   
     
   Now we have a problem. COPD1 has White, Black, Other, and Multi.  
   POPULATION1 has additional columns Native…, Asian, Pacific…, Some Other…  
   So the definition of OTHER is different in the two data sets. A common problem.  
     
   This is called a problem of different grain size. POPULATION1 is more fine-grained.

Write an SQL statement to display the POPULATION1 table.

Copy/paste your SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show only the first 10 rows. [5 points]

► SELECT \*

FROM population1

FETCH FIRST 10 ROWS ONLY;

► A screenshot of a computer

Description automatically generated

1. Write a SELECT statement against POPULATION1 that returns the columns State, White, Black, Mixed, and a new computed column OTHER2 that contains the SUM of Native…, Asian, Pacific…, Some Other… So, there will be 5 columns in the answer.   
   [5 points]  
     
   Hand-check the first two rows to make sure the result makes sense. NO CREDIT on this.

Copy/paste your SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show only the first 10 rows. [5 points]

► SELECT

State,

white,

black\_or\_african\_american,

mixed\_race\_multi\_racial,

(Native\_American\_or\_Alaska\_Native + Asian + Pacific\_Islander + Some\_Other\_Race) AS OTHER2

FROM

POPULATION1;

SELECT \*

FROM population1

FETCH FIRST 10 ROWS ONLY;

► A screenshot of a computer

Description automatically generated

1. Write a combined CREATE/SELECT that captures the result of Question 8) into a new table POPULATION2.

Copy/paste your SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show only the first 10 rows. [3 points]

► CREATE TABLE POPULATION2 AS

SELECT

State,

White,

black\_or\_african\_american,

mixed\_race\_multi\_racial,

(Native\_American\_or\_Alaska\_Native + Asian + Pacific\_Islander + Some\_Other\_Race) AS OTHER2

FROM

POPULATION1;

SELECT \*

FROM population2

FETCH FIRST 10 ROWS ONLY;

► A screenshot of a computer

Description automatically generated

1. Write a PL/SQL program using an **explicit** cursor that displays all columns sorted by OTHER2 in descending order in the table POPULATION2.

Copy/paste your PL/SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show the first 10 and the last 10 rows. [5 points]  
Your screen outputs should look like this:

► DECLARE

CURSOR population\_cursor IS

SELECT \*

FROM POPULATION2

ORDER BY OTHER2 DESC;

v\_State VARCHAR2(50);

v\_White NUMBER;

v\_Black NUMBER;

v\_Mixed NUMBER;

v\_OTHER2 NUMBER;

BEGIN

OPEN population\_cursor;

DBMS\_OUTPUT.PUT\_LINE(

RPAD('State', 20) || RPAD('White', 12) || RPAD('Black', 12) ||

RPAD('Mixed', 12) || RPAD('OTHER2', 12)

);

DBMS\_OUTPUT.PUT\_LINE(

RPAD('--------------------', 20, '-') || RPAD('------------', 12, '-') ||

RPAD('------------', 12, '-') || RPAD('------------', 12, '-') ||

RPAD('------------', 12, '-')

);

LOOP

FETCH population\_cursor INTO

v\_State, v\_White, v\_Black, v\_Mixed, v\_OTHER2;

EXIT WHEN population\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(

RPAD(v\_State, 20) || RPAD(TO\_CHAR(v\_White), 12) ||

RPAD(TO\_CHAR(v\_Black), 12) || RPAD(TO\_CHAR(v\_Mixed), 12) ||

RPAD(TO\_CHAR(v\_OTHER2), 12)

);

END LOOP;

CLOSE population\_cursor;

END;

/

set serveroutput on

► First 10:

A screenshot of a computer

Description automatically generated

Last 10:

A number on a white background

Description automatically generated

1. Now we are finally ready to start the real work. Do a JOIN between COPD1 and POPULATION2 so that we get a table that contains every State multiple time, with all race information from POPULATION2.

Copy/paste your SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show only the first 10 rows. [5 points]

► SELECT

C.State,

C.Value,

C.Year,

C.RaceEthnicity,

P.White,

P.Black\_or\_African\_American,

P.Mixed\_Race\_Multi\_Racial,

P.OTHER2

FROM

COPD C

LEFT JOIN

POPULATION2 P

ON

TRIM(BOTH ' ' FROM C.State) = TRIM(BOTH ' ' FROM P.State)

WHERE

ROWNUM <= 10;

► A screenshot of a computer

Description automatically generated

1. Write an SQL statement to capture the result of 11) in a new table POPULATION\_COPD.

Copy/paste your SQL code at the first red arrow.

In the NEXT Homework, you will write a PL/SQL program that computes the absolute number of Whites with COPD, Blacks with COPD, Others with COPD, and Mixed with COPD, based on the table POPULATION\_COPD.

Write an SQL statement to display the POPULATION\_COPD table.

Copy/paste your SQL code at the second red arrow and SNIP the screen output at the third red arrow. Show only the first 10 rows. [2 points]

► CREATE TABLE POPULATION\_COPD AS

SELECT

C.State,

C.Value,

C.Year,

C.RaceEthnicity,

P.White,

P.Black\_or\_African\_American,

P.Mixed\_Race\_Multi\_Racial,

P.OTHER2

FROM

(SELECT State, SUM(Value) AS OTHER2

FROM COPD1

GROUP BY State) C

LEFT JOIN

POPULATION2 P

ON

TRIM(BOTH ' ' FROM C.State) = TRIM(BOTH ' ' FROM P.State);

►

A screenshot of a computer

Description automatically generated

1. Write a PL/SQL program using an **explicit** cursor that displays states and summation of values for each state, sorted by the state in ascending order in the table POPULATION\_COPD. [5 points]

Copy/paste your PL/SQL code at the first red arrow and SNIP the screen output at the second red arrow. Show the first 10 and the last 10 rows. [5 points]  
Your screen outputs should look like this:

► DECLARE

CURSOR copd\_cursor IS

SELECT State, SUM(Value) AS Total\_Value

FROM population\_copd

GROUP BY State

ORDER BY State ASC;

v\_State VARCHAR2(50);

v\_Total\_Value NUMBER;

BEGIN

OPEN copd\_cursor;

DBMS\_OUTPUT.PUT\_LINE('State | Total Value');

DBMS\_OUTPUT.PUT\_LINE('------------------');

LOOP

FETCH copd\_cursor INTO

v\_State, v\_Total\_Value;

EXIT WHEN copd\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(

RPAD(v\_State, 20) || TO\_CHAR(v\_Total\_Value, '999,999.99')

);

END LOOP;

CLOSE copd\_cursor;

END;

/

First 10 Last 10

► A screenshot of a document

Description automatically generatedA screenshot of a computer screen

Description automatically generated