**CS434 Fall 2023**

**HOMEWORK 2**

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) Save this file. All questions will be in boxes like this. Put your answer to every question into the space AFTER and OUTSIDE of the box. Then submit the whole file.

3) 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.

4) The whole homework is worth 50 points.

Points will appear in [ ] brackets.

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  **Friday, 20/October/2023 11:59 PM**

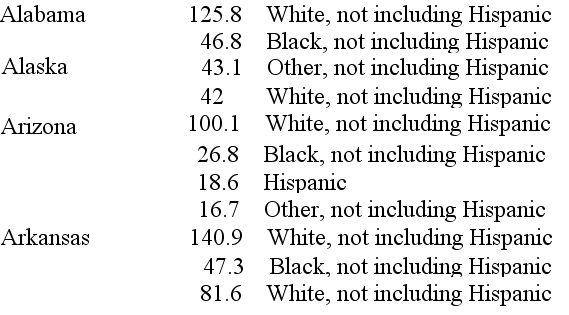
**Your file name must have the format:**

**LASTNAME\_firstname\_HWK2.doc (or .docx)**

This homework has three purposes:

* Continue writing "simple" PL/SQL programs.
* Start writing programs with cursors.
* Start writing programs with triggers

1. Write a PL/SQL program using an **implicit** cursor that displays each state name, value for each race, and race-ethnicity in the following format:



The state name should be written only once. You can only use the POPULATION\_COPD table. If you use any other table, 0 points.

Show the program at the first red arrow.

Show the first 10 rows of the result at the second red arrow by snipping them. [10 points]

►SET SERVEROUTPUT ON;

DECLARE

v\_CurrentState VARCHAR2(50) := NULL;

CURSOR population\_cursor IS

SELECT State, Value, RaceEthnicity

FROM Population\_copd

ORDER BY State, RaceEthnicity;

BEGIN

FOR r IN population\_cursor LOOP

IF v\_CurrentState IS NULL OR v\_CurrentState <> r.State THEN

DBMS\_OUTPUT.PUT\_LINE(r.State);

v\_CurrentState := r.State;

END IF;

DBMS\_OUTPUT.PUT\_LINE(

RPAD(' ', 15) || TO\_CHAR(r.Value, '9999.9') || ' ' || r.RaceEthnicity

);

END LOOP;

END;

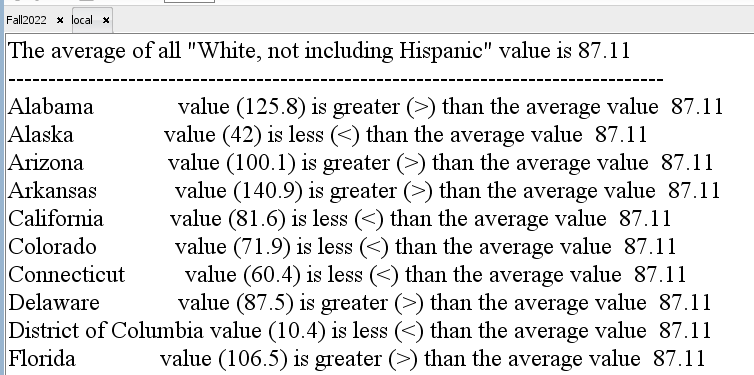
/

►A screenshot of a computer

Description automatically generated

**2-a)** Write a PL/SQL program using an **implicit** cursor that compares only White, not including Hispanic values to the average of White, not including Hispanic values.

Your output should look like the following:



You can only use the COPD1 table. If you use any other table, 0 points.

Show the program at the first red arrow.

Show the first 10 rows of the result at the second red arrow by snipping them. [5 points]

► SET SERVEROUTPUT ON;

DECLARE

v\_AverageWhite NUMBER;

BEGIN

-- Calculate the average value for "White, not including Hispanic"

SELECT AVG(Value)

INTO v\_AverageWhite

FROM COPD1

WHERE RaceEthnicity = 'White, not including Hispanic';

-- Display the average value

DBMS\_OUTPUT.PUT\_LINE('The average of all "White, not including Hispanic" value is ' || TO\_CHAR(v\_AverageWhite, '999.99'));

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

-- Compare each state's value with the average

FOR r IN (SELECT State, Value FROM COPD1 WHERE RaceEthnicity = 'White, not including Hispanic') LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(r.State, 20) || 'value(' || TO\_CHAR(r.Value, '999.9') || ') is ' ||

CASE

WHEN r.Value > v\_AverageWhite THEN 'greater (>) than'

WHEN r.Value < v\_AverageWhite THEN 'less (<) than'

ELSE 'equal to'

END ||

' average value ' || TO\_CHAR(v\_AverageWhite, '999.9')

);

END LOOP;

END;

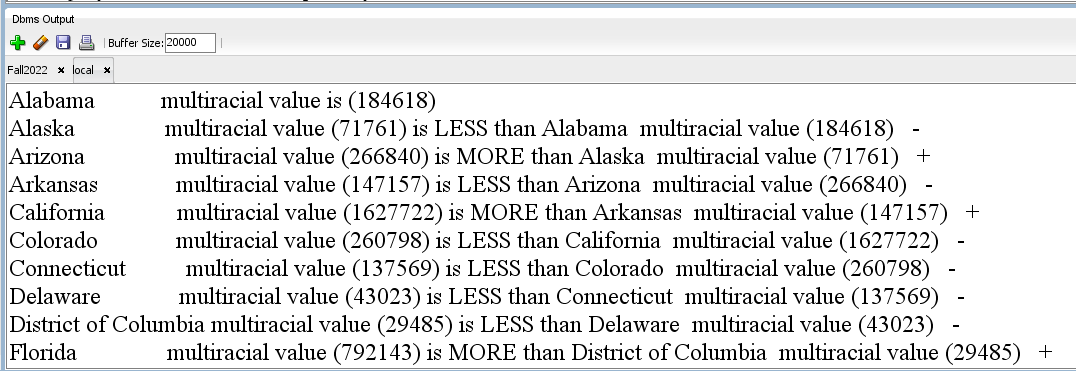
/

►A screenshot of a computer

Description automatically generated

**2-b)** Write a PL/SQL program using an **implicit** cursor that compares MULTIRACIAL values for all states in alphabetical order.

Your output should look like the following:



You can only use the POPULATION2 table. If you use any other table, 0 points.

Show the program at the first red arrow.

Show the first 10 rows of the result at the second red arrow by snipping them. [5 points]

►SET SERVEROUTPUT ON;

DECLARE

CURSOR population\_cursor IS

SELECT State, mixed\_race\_multi\_racial AS Multiracial

FROM POPULATION2

ORDER BY State;

v\_PrevState VARCHAR2(50) := NULL;

v\_PrevValue NUMBER;

v\_CurrentState VARCHAR2(50);

v\_CurrentValue NUMBER;

v\_FirstRow BOOLEAN := TRUE;

BEGIN

OPEN population\_cursor;

-- Fetch the first row

FETCH population\_cursor INTO v\_PrevState, v\_PrevValue;

-- Loop through the cursor

LOOP

-- Fetch the next row

FETCH population\_cursor INTO v\_CurrentState, v\_CurrentValue;

-- Exit the loop if there are no more rows

EXIT WHEN population\_cursor%NOTFOUND;

-- Skip the comparison for the first state (Alabama)

IF v\_FirstRow THEN

v\_FirstRow := FALSE;

CONTINUE;

END IF;

-- Compare the values and print the result

DBMS\_OUTPUT.PUT(v\_CurrentState || ' multiracial value is (' || v\_CurrentValue || ')');

IF v\_CurrentValue < v\_PrevValue THEN

DBMS\_OUTPUT.PUT\_LINE(' is LESS than ' || v\_PrevState || ' multiracial value (' || v\_PrevValue || ') -');

ELSIF v\_CurrentValue > v\_PrevValue THEN

DBMS\_OUTPUT.PUT\_LINE(' is MORE than ' || v\_PrevState || ' multiracial value (' || v\_PrevValue || ') +');

ELSE

DBMS\_OUTPUT.PUT\_LINE(' is EQUAL to ' || v\_PrevState || ' multiracial value (' || v\_PrevValue || ') =');

END IF;

-- Update the previous state and value

v\_PrevState := v\_CurrentState;

v\_PrevValue := v\_CurrentValue;

END LOOP;

CLOSE population\_cursor;

END;

/

►A screenshot of a computer

Description automatically generated

**3-** This question has two steps:

**STEP1:** Write a PL/SQL statement to add a new column (SUMMATION) into the table POPULATION2. [3 point]

Show the PL/SQL statement at the first red arrow.

**STEP2:** Write a PL/SQL program using an **implicit** cursor that adds integer values of POPULATION2 in a row and then saves them in a SUMMATION column. [7 points]

Show the program at the second red arrow.

Show the first 10 rows of the result at the third red arrow by snipping them.

Show the last 10 rows of the result at the fourth red arrow by snipping them.

STEP1

► ALTER TABLE POPULATION2

ADD SUMMATION NUMBER;

STEP2

► DECLARE

CURSOR population\_cursor IS

SELECT State, White, black\_or\_african\_american, mixed\_race\_multi\_racial,other2

FROM POPULATION2;

v\_State VARCHAR2(50);

v\_White NUMBER;

v\_Black NUMBER;

v\_Mixed NUMBER;

v\_other NUMBER;

v\_Summation NUMBER;

BEGIN

FOR r IN population\_cursor LOOP

v\_State := r.State;

v\_White := r.White;

v\_Black := r.black\_or\_african\_american;

v\_Mixed := r.mixed\_race\_multi\_racial;

v\_other := r.other2;

-- Calculate the sum of integer values

v\_Summation := TRUNC(v\_White) + TRUNC(v\_Black) + TRUNC(v\_Mixed);

-- Update the "SUMMATION" column

UPDATE POPULATION2

SET SUMMATION = v\_Summation

WHERE State = v\_State;

END LOOP;

COMMIT;

END;

/

► First 10 Rows:

A screenshot of a computer

Description automatically generated

► Last 10 Rows:

A screenshot of a computer

Description automatically generated

4- Create a trigger **NewPopulation** that will guarantee any time the table POPULATION1’s NATIVEAMERICAN or ASIAN or PACIFICISLANDER or OTHERRACE population status is updated, the trigger writes the following message to the output window.

|  |
| --- |
| STATE:  New Population Race:  Old Population Value:  New Population Value: |

|  |  |
| --- | --- |
| **Test it with:** | **Output should look like:** |
| update POPULATION1  set PACIFICISLANDER ++  where STATE = ‘New Jersey’;/ |  |
| update POPULATION1  set WHITE=WHITE+4  where state='New Jersey';/ |  |
| update POPULATION1  set NATIVEAMERICAN=NATIVEAMERICAN+1;/ |  |

++ Population may increase any number.

Show the program at the first red arrow.

Show the outputs of the trigger at the second red arrow by snipping them. [10]

► CREATE OR REPLACE TRIGGER NewPopulation

BEFORE UPDATE OF NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE, ASIAN, PACIFIC\_ISLANDER, SOME\_OTHER\_RACE

ON POPULATION1

FOR EACH ROW

BEGIN

-- Check if any of the specified columns are updated

IF :NEW.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE != :OLD.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE OR

:NEW.ASIAN != :OLD.ASIAN OR

:NEW.PACIFIC\_ISLANDER != :OLD.PACIFIC\_ISLANDER OR

:NEW.SOME\_OTHER\_RACE != :OLD.SOME\_OTHER\_RACE THEN

DBMS\_OUTPUT.PUT\_LINE('STATE: ' || :NEW.STATE);

DBMS\_OUTPUT.PUT\_LINE('New Population Race:');

IF :NEW.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE != :OLD.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE THEN

DBMS\_OUTPUT.PUT\_LINE('NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE - Old Population Value: ' || :OLD.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE);

DBMS\_OUTPUT.PUT\_LINE('NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE - New Population Value: ' || :NEW.NATIVE\_AMERICAN\_OR\_ALASKA\_NATIVE);

END IF;

IF :NEW.ASIAN != :OLD.ASIAN THEN

DBMS\_OUTPUT.PUT\_LINE('ASIAN - Old Population Value: ' || :OLD.ASIAN);

DBMS\_OUTPUT.PUT\_LINE('ASIAN - New Population Value: ' || :NEW.ASIAN);

END IF;

IF :NEW.PACIFIC\_ISLANDER != :OLD.PACIFIC\_ISLANDER THEN

DBMS\_OUTPUT.PUT\_LINE('PACIFIC\_ISLANDER - Old Population Value: ' || :OLD.PACIFIC\_ISLANDER);

DBMS\_OUTPUT.PUT\_LINE('PACIFIC\_ISLANDER - New Population Value: ' || :NEW.PACIFIC\_ISLANDER);

END IF;

IF :NEW.SOME\_OTHER\_RACE != :OLD.SOME\_OTHER\_RACE THEN

DBMS\_OUTPUT.PUT\_LINE('SOME\_OTHER\_RACE - Old Population Value: ' || :OLD.SOME\_OTHER\_RACE);

DBMS\_OUTPUT.PUT\_LINE('SOME\_OTHER\_RACE - New Population Value: ' || :NEW.SOME\_OTHER\_RACE);

END IF;

END IF;

END;

/

► Pacific\_Islander++ Output:

A screenshot of a computer

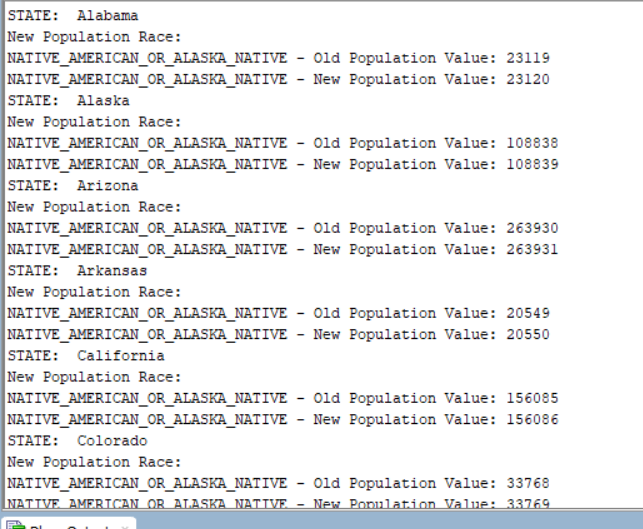
Description automatically generated

White+4 Output:

A computer screen shot of a program

Description automatically generated

Native\_American Output:



**5-** Create another trigger **NewPopulation2** that will guarantee any time the table POPULATION1’s PACIFICISLANDER or OTHERRACE population is updated, the trigger updates the table PopulationTrack with old and new values of Pacific Islanders and Other Race values.

**PopulationTrack**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STATE** | **OLD\_**  **PACIFICISLANDER** | **NEW\_**  **PACIFICISLANDER** | **OLD\_**  **OTHERRACE** | **NEW\_**  **OTHERRACE** |
|  |  |  |  |  |
|  |  |  |  |  |

**Test it with:**

update POPULATION1

set PACIFICISLANDER ++

where STATE = ‘New Jersey’;/

update POPULATION1

set OTHERRACE=OTHERRACE+5;

/

**PopulationTrack table should look like:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STATE** | **OLD\_**  **PACIFICISLANDER** | **NEW\_**  **PACIFICISLANDER** | **OLD\_**  **OTHERRACE** | **NEW\_**  **OTHERRACE** |
| New Jersey | 1947 | 1948 | null | null |
| Alabama | null | null | 14455 | 14460 |
| Alaska | null | null | 4575 | 4580 |
| Arizona | null | null | 31611 | 31616 |
| Arkansas | null | null | 8047 | 8052 |
| California | null | null | 223929 | 223934 |
| Many more rows… | | | | |

Show the PopulationTrack table creation SQL statement at the first red arrow.

Show the trigger code at the second red arrow.

Show the first 10 rows of the PopulationTrack table at the third red arrow by snipping it. [10]

►CREATE TABLE PopulationTrack (

STATE VARCHAR2(50),

OLD\_PACIFIC\_ISLANDER NUMBER,

NEW\_PACIFIC\_ISLANDER NUMBER,

OLD\_SOME\_OTHER\_RACE NUMBER,

NEW\_SOME\_OTHER\_RACE NUMBER

);

►CREATE OR REPLACE TRIGGER NewPopulation2

AFTER UPDATE OF PACIFIC\_ISLANDER, SOME\_OTHER\_RACE ON POPULATION1

FOR EACH ROW

BEGIN

-- Insert the old and new values into PopulationTrack

INSERT INTO PopulationTrack (STATE, OLD\_PACIFIC\_ISLANDER, NEW\_PACIFIC\_ISLANDER, OLD\_SOME\_OTHER\_RACE, NEW\_SOME\_OTHER\_RACE)

VALUES (:NEW.STATE, :OLD.PACIFIC\_ISLANDER, :NEW.PACIFIC\_ISLANDER, :OLD.SOME\_OTHER\_RACE, :NEW.SOME\_OTHER\_RACE);

END;

/

► NJ has +1 pacifc islander and +5 for some other race in output below

A screenshot of a computer

Description automatically generated