**CS434 Fall 2023**

**HOMEWORK 3**

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  **Tuesday, November 20 at 4:00 PM.**

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

**LASTNAME\_firstname\_HWK3.doc (or .docx)**

1-a) 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**

Choose all these 4 years:

2017, 2018, 2019, 2020

**STEP 5: ADVANCED OPTIONS**

Race Ethnicity

All 4 Choices for Race

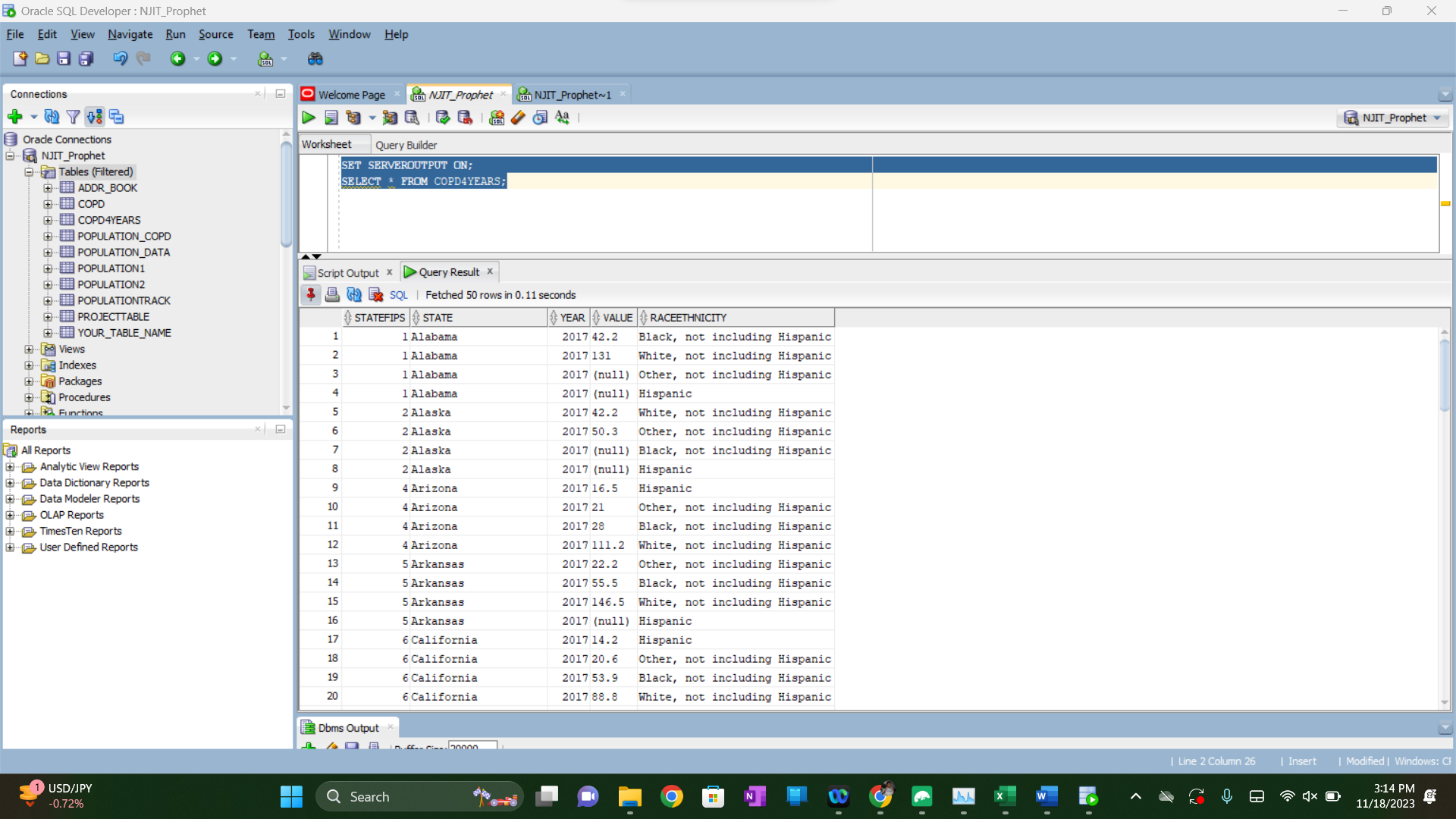
Then download the data and save it as a COPD4YEARS.CSV file [1 point].

Replace all “Suppressed” values in the VALUE column to NULL.

**b)** Load the Cleaned Data into an Oracle Table COPD4YEARS using SQL Developer.   
Write an SQL statement to display the COPD4YEARS table.

► SELECT \* FROM COPD4YEARS

After the first red arrow below, show a screen dump that shows the select statement and at least the first 20 lines for the table. [2]

► 

**c)** Create an Oracle Class **race\_o** from the COPD4YEARS table with the following attributes; White, Black, Hispanic, and Others

Show the code for the types after the second red arrow. [1]

► CREATE OR REPLACE TYPE race\_o AS OBJECT (

White NUMBER,

Black NUMBER,

Hispanic NUMBER,

Others NUMBER

);

A screenshot of a computer

Description automatically generated►

**d)** Create a table **COPDobject** with **State** attribute data type of VARCHAR2(26) and four attributes with (data type of race\_o) named Year2017, Year2018, year2019, and Year2020 respectively.

Show the code for the table after the third red arrow [1].

► CREATE TABLE COPDobject (

State VARCHAR2(26),

Year2017 race\_o,

Year2018 race\_o,

Year2019 race\_o,

Year2020 race\_o

);

► A screenshot of a computer

Description automatically generated

**2) a)** Write a PL/SQL program using an **implicit** cursor that inserts into a **COPDobject** table all states’ names and each year all race data.

Show the program after the first red arrow.

► DECLARE

v\_year\_2017 race\_o;

v\_year\_2018 race\_o;

v\_year\_2019 race\_o;

v\_year\_2020 race\_o;

v\_current\_state VARCHAR2(100);

v\_previous\_state VARCHAR2(100) := NULL;

FUNCTION reset\_race\_o RETURN race\_o IS

BEGIN

RETURN race\_o(0, 0, 0, 0);

END reset\_race\_o;

BEGIN

FOR rec IN (SELECT State, Year, Value, RACEETHNICITY FROM COPD4YEARS ORDER BY State, Year, RACEETHNICITY) LOOP

v\_current\_state := rec.State;

IF v\_current\_state != v\_previous\_state AND v\_previous\_state IS NOT NULL THEN

INSERT INTO COPDobject (State, Year2017, Year2018, Year2019, Year2020)

VALUES (v\_previous\_state, v\_year\_2017, v\_year\_2018, v\_year\_2019, v\_year\_2020);

v\_year\_2017 := reset\_race\_o;

v\_year\_2018 := reset\_race\_o;

v\_year\_2019 := reset\_race\_o;

v\_year\_2020 := reset\_race\_o;

END IF;

IF rec.Year = 2017 THEN

v\_year\_2017 := race\_o(

CASE WHEN rec.RACEETHNICITY = 'White, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2017.White END,

CASE WHEN rec.RACEETHNICITY = 'Black, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2017.Black END,

CASE WHEN rec.RACEETHNICITY = 'Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2017.Hispanic END,

CASE WHEN rec.RACEETHNICITY = 'Other, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2017.Others END

);

ELSIF rec.Year = 2018 THEN

v\_year\_2018 := race\_o(

CASE WHEN rec.RACEETHNICITY = 'White, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2018.White END,

CASE WHEN rec.RACEETHNICITY = 'Black, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2018.Black END,

CASE WHEN rec.RACEETHNICITY = 'Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2018.Hispanic END,

CASE WHEN rec.RACEETHNICITY = 'Other, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2018.Others END

);

ELSIF rec.Year = 2019 THEN

v\_year\_2019 := race\_o(

CASE WHEN rec.RACEETHNICITY = 'White, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2019.White END,

CASE WHEN rec.RACEETHNICITY = 'Black, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2019.Black END,

CASE WHEN rec.RACEETHNICITY = 'Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2019.Hispanic END,

CASE WHEN rec.RACEETHNICITY = 'Other, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2019.Others END

);

ELSIF rec.Year = 2020 THEN

v\_year\_2020 := race\_o(

CASE WHEN rec.RACEETHNICITY = 'White, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2020.White END,

CASE WHEN rec.RACEETHNICITY = 'Black, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2020.Black END,

CASE WHEN rec.RACEETHNICITY = 'Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2020.Hispanic END,

CASE WHEN rec.RACEETHNICITY = 'Other, not including Hispanic' THEN NVL(rec.Value, 0) ELSE v\_year\_2020.Others END

);

END IF;

v\_previous\_state := v\_current\_state;

END LOOP;

IF v\_previous\_state IS NOT NULL THEN

INSERT INTO COPDobject (State, Year2017, Year2018, Year2019, Year2020)

VALUES (v\_previous\_state, v\_year\_2017, v\_year\_2018, v\_year\_2019, v\_year\_2020);

END IF;

END;

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

► Output after iterating each race\_o object:  
 A screenshot of a computer

Description automatically generated

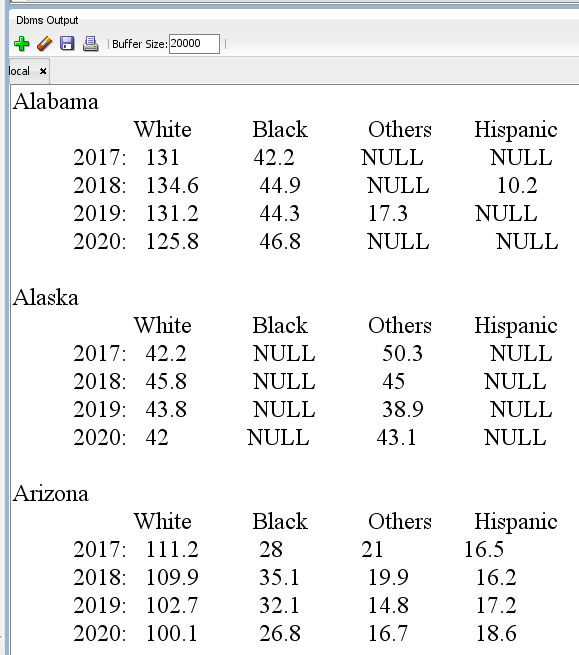
And ouput as table itself (but sqldeveloper didn’t let me display race\_o object in its format. Rather it used my accountName.race\_o as:

A screenshot of a computer

Description automatically generated

b) Write a PL/SQL program using an **implicit** cursor that displays information about all states in the table **COPDobject** in a nice format. Use rpad().

Your output should look like this:



Show the program at the third red arrow.

Show the output of the first three state data **COPDobject** table at the fourth red arrow, the last three state data at the fifth red arrow by snipping them [10].

► DECLARE

CURSOR copd\_cursor IS

SELECT State,

Year2017, Year2018, Year2019, Year2020

FROM COPDobject;

rec copd\_cursor%ROWTYPE;

FUNCTION zero\_to\_null(p\_value IN NUMBER) RETURN VARCHAR2 IS

BEGIN

IF p\_value = 0 THEN

RETURN 'NULL';

ELSE

RETURN TO\_CHAR(p\_value);

END IF;

END zero\_to\_null;

BEGIN

FOR rec IN copd\_cursor LOOP

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

DBMS\_OUTPUT.PUT\_LINE(RPAD(' ', 5) || 'White' || RPAD(' ', 4) || 'Black' || RPAD(' ', 4) || 'Others' || RPAD(' ', 4) || 'Hispanic');

DBMS\_OUTPUT.PUT\_LINE('2017: ' || RPAD(zero\_to\_null(rec.Year2017.White), 10) || RPAD(zero\_to\_null(rec.Year2017.Black), 10) || RPAD(zero\_to\_null(rec.Year2017.Others), 10) || RPAD(zero\_to\_null(rec.Year2017.Hispanic), 10));

DBMS\_OUTPUT.PUT\_LINE('2018: ' || RPAD(zero\_to\_null(rec.Year2018.White), 10) || RPAD(zero\_to\_null(rec.Year2018.Black), 10) || RPAD(zero\_to\_null(rec.Year2018.Others), 10) || RPAD(zero\_to\_null(rec.Year2018.Hispanic), 10));

DBMS\_OUTPUT.PUT\_LINE('2019: ' || RPAD(zero\_to\_null(rec.Year2019.White), 10) || RPAD(zero\_to\_null(rec.Year2019.Black), 10) || RPAD(zero\_to\_null(rec.Year2019.Others), 10) || RPAD(zero\_to\_null(rec.Year2019.Hispanic), 10));

DBMS\_OUTPUT.PUT\_LINE('2020: ' || RPAD(zero\_to\_null(rec.Year2020.White), 10) || RPAD(zero\_to\_null(rec.Year2020.Black), 10) || RPAD(zero\_to\_null(rec.Year2020.Others), 10) || RPAD(zero\_to\_null(rec.Year2020.Hispanic), 10));

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

END LOOP;

END;

SET SERVEROUTPUT ON;

► A screenshot of a computer

Description automatically generated

► A screenshot of a computer screen

Description automatically generated

**3**- Write a PL/SQL program using an **implicit cursor** that loads all the data from the POPULATION2 (from Homework1) table into the POPULATIONXML table.

To clarify: POPULATIONXML has to have all rows with the following information from POPULATION2, but in a single column, as XML expressions.

STATE, WHITE, AFRICANAMERICAN, MULTIRACIAL, OTHER for

WHITE>1,000,000 and AFRICANAMERICAN >10,000 and MULTIRACIAL>50,000

HINTS: Students usually have a really hard time with this.

You need to construct the XML expression from strings with lots of concatenation (||) operators.

'<population>

<STATE>' || variablefromcursorwithSTATEinit || '</STATE>

<WHITE>' || variablefromcursorwithWHITEinit || </WHITE>'

Etc. etc.

Show the POPULATIONXML table creation statement at the first red arrow below [1 point].

Show the INDENTED program to insert data with cursor at the second red arrow below [6].

► A screenshot of a computer

Description automatically generated

► DECLARE

CURSOR pop\_cursor IS

SELECT STATE,

WHITE,

BLACK\_OR\_AFRICAN\_AMERICAN,

MIXED\_RACE\_MULTI\_RACIAL,

OTHER2

FROM POPULATION2

WHERE WHITE > 1000000

AND BLACK\_OR\_AFRICAN\_AMERICAN > 10000

AND MIXED\_RACE\_MULTI\_RACIAL > 50000;

v\_xml\_expression VARCHAR2(4000);

BEGIN

FOR rec IN pop\_cursor LOOP

v\_xml\_expression :=

'<population>' ||

'<STATE>' || rec.STATE || '</STATE>' ||

'<WHITE>' || rec.WHITE || '</WHITE>' ||

'<AFRICANAMERICAN>' || rec.BLACK\_OR\_AFRICAN\_AMERICAN || '</AFRICANAMERICAN>' ||

'<MULTIRACIAL>' || rec.MIXED\_RACE\_MULTI\_RACIAL || '</MULTIRACIAL>' ||

'<OTHER>' || rec.OTHER2 || '</OTHER>' ||

'</population>';

INSERT INTO POPULATIONXML (PopulationData)

VALUES (v\_xml\_expression);

END LOOP;

END;

**4**- Write a PL/SQL program that uses ONLY the table POPULATIONXML and does NOT use the table POPULATION2 or any other table or any other view. Zero points if you do not follow these rules.

This program should send the complete table POPULATIONXML to the screen in the following format:

STATE : Alabama

White : 3171351

African American : 1288159

Multiracial : 184618

Others : 116104

STATE : Arizona

White : 3816547

African American : 317161

Multiracial : 266840

Others : 558701

STATE : Arkansas

White : 2063550

African American : 449884

Multiracial : 147157

Others : 94086

STATE : California

Etc. etc.

Note the empty line. There has to be an empty line after every new table row. Every table row becomes 5 lines + 1 empty line after.

Note that all the ":" are lined up in the same column. It has to be that way.

Show the program at the first red arrow below.

Show the first 12 lines of the result at the second red arrow below [6 points].

► DECLARE

CURSOR xml\_cursor IS

SELECT PopulationData

FROM POPULATIONXML;

v\_xml CLOB;

v\_state VARCHAR2(100);

v\_white NUMBER;

v\_african\_american NUMBER;

v\_multiracial NUMBER;

v\_others NUMBER;

BEGIN

FOR rec IN xml\_cursor LOOP

v\_xml := rec.PopulationData;

SELECT EXTRACTVALUE(XMLTYPE(v\_xml), '/population/STATE') INTO v\_state FROM DUAL;

SELECT EXTRACTVALUE(XMLTYPE(v\_xml), '/population/WHITE') INTO v\_white FROM DUAL;

SELECT EXTRACTVALUE(XMLTYPE(v\_xml), '/population/AFRICANAMERICAN') INTO v\_african\_american FROM DUAL;

SELECT EXTRACTVALUE(XMLTYPE(v\_xml), '/population/MULTIRACIAL') INTO v\_multiracial FROM DUAL;

SELECT EXTRACTVALUE(XMLTYPE(v\_xml), '/population/OTHER') INTO v\_others FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE('STATE : ' || RPAD(v\_state, 20));

DBMS\_OUTPUT.PUT\_LINE(RPAD('White', 25) || ': ' || v\_white);

DBMS\_OUTPUT.PUT\_LINE(RPAD('African American', 25) || ': ' || v\_african\_american);

DBMS\_OUTPUT.PUT\_LINE(RPAD('Multiracial', 25) || ': ' || v\_multiracial);

DBMS\_OUTPUT.PUT\_LINE(RPAD('Others', 25) || ': ' || v\_others);

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

END LOOP;

END;

► A screenshot of a computer

Description automatically generated

**5-** Create a table POPULATION\_JSON in Oracle. It should contain a single column POPULATION that contains STATE, WHITE, AFRICANAMERICAN, MULTIRACIAL, OTHER attributes of POPULATION2.

Then, write a PL/SQL program using an **implicit cursor** that loads all the data from the POPULATION2 table into the POPULATION\_JSON.

Show the POPULATION\_JSON table creation statement at the first red arrow below [1 point].

Show the INDENTED program to insert data with cursor at the second red arrow below [5].

► A screenshot of a computer

Description automatically generated

► DECLARE

CURSOR pop\_cursor IS

SELECT STATE,

WHITE,

BLACK\_OR\_AFRICAN\_AMERICAN,

MIXED\_RACE\_MULTI\_RACIAL,

OTHER2

FROM POPULATION2;

v\_json CLOB;

BEGIN

FOR rec IN pop\_cursor LOOP

v\_json :=

'{' ||

'"state": "' || rec.STATE || '",' ||

'"white": ' || rec.WHITE || ',' ||

'"africanAmerican": ' || rec.BLACK\_OR\_AFRICAN\_AMERICAN || ',' ||

'"multiracial": ' || rec.MIXED\_RACE\_MULTI\_RACIAL || ',' ||

'"other": ' || rec.OTHER2 ||

'}';

INSERT INTO POPULATION\_JSON (Population)

VALUES (v\_json);

END LOOP;

END;

**6-** Write a program **using a cursor** that displays the complete table POPULATION\_JSON on the screen in the following format. NOTE THE NEWLINES.

STATE : Alabama

White : 3171351

African American : 1288159

Multiracial : 184618

Others : 116104

STATE : Arizona

White : 3816547

African American : 317161

Multiracial : 266840

Others : 558701

Etc…etc….

Every table row (i.e., JSON expression) becomes five rows followed by an empty row.

Show the program at the first red arrow below.

Show the first 12 lines of the result at the second red arrow below [6 points]

► DECLARE

CURSOR json\_cursor IS

SELECT Population

FROM POPULATION\_JSON;

v\_state VARCHAR2(100);

v\_white VARCHAR2(100);

v\_african\_american VARCHAR2(100);

v\_multiracial VARCHAR2(100);

v\_others VARCHAR2(100);

FUNCTION get\_json\_value(json\_data IN CLOB, key IN VARCHAR2) RETURN VARCHAR2 IS

v\_start\_pos PLS\_INTEGER;

v\_end\_pos PLS\_INTEGER;

BEGIN

v\_start\_pos := INSTR(json\_data, '"' || key || '": ') + LENGTH(key) + 4;

IF v\_start\_pos > LENGTH(key) + 4 THEN

v\_end\_pos := INSTR(json\_data, ',', v\_start\_pos);

IF v\_end\_pos = 0 THEN

v\_end\_pos := INSTR(json\_data, '}', v\_start\_pos);

END IF;

RETURN TRIM(BOTH '"' FROM SUBSTR(json\_data, v\_start\_pos, v\_end\_pos - v\_start\_pos));

ELSE

RETURN NULL;

END IF;

END get\_json\_value;

BEGIN

FOR rec IN json\_cursor LOOP

v\_state := get\_json\_value(rec.Population, 'state');

v\_white := get\_json\_value(rec.Population, 'white');

v\_african\_american := get\_json\_value(rec.Population, 'africanAmerican');

v\_multiracial := get\_json\_value(rec.Population, 'multiracial');

v\_others := get\_json\_value(rec.Population, 'other');

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

DBMS\_OUTPUT.PUT\_LINE(RPAD('White', 25) || ': ' || v\_white);

DBMS\_OUTPUT.PUT\_LINE(RPAD('African American', 25) || ': ' || v\_african\_american);

DBMS\_OUTPUT.PUT\_LINE(RPAD('Multiracial', 25) || ': ' || v\_multiracial);

DBMS\_OUTPUT.PUT\_LINE(RPAD('Others', 25) || ': ' || v\_others);

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

END LOOP;

A screenshot of a computer

Description automatically generatedEND;

►