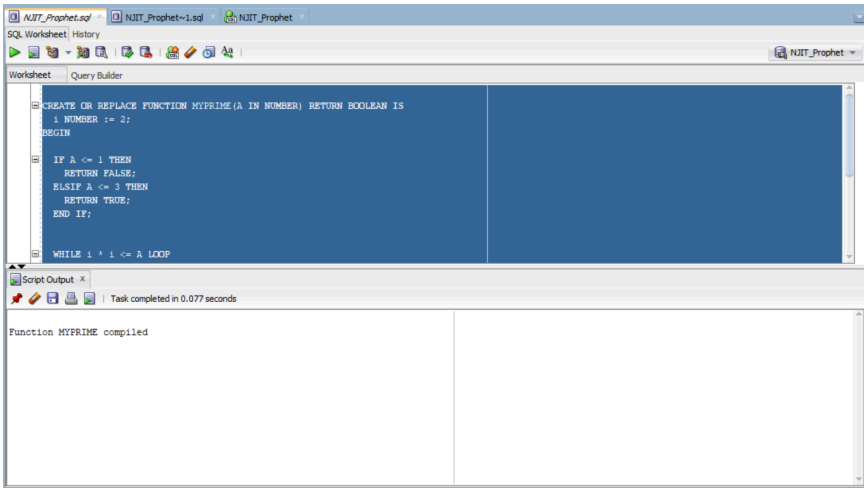
1) **ALL PL/SQL work needs to be done with SQL Developer. All database work needs to be done in Oracle.**

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 take one 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
```



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:

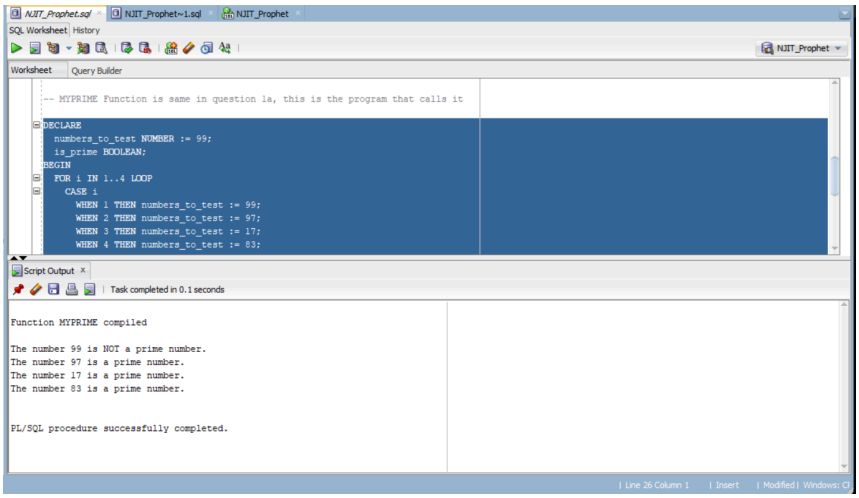
The number 99 is NOT a prime number.

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
```

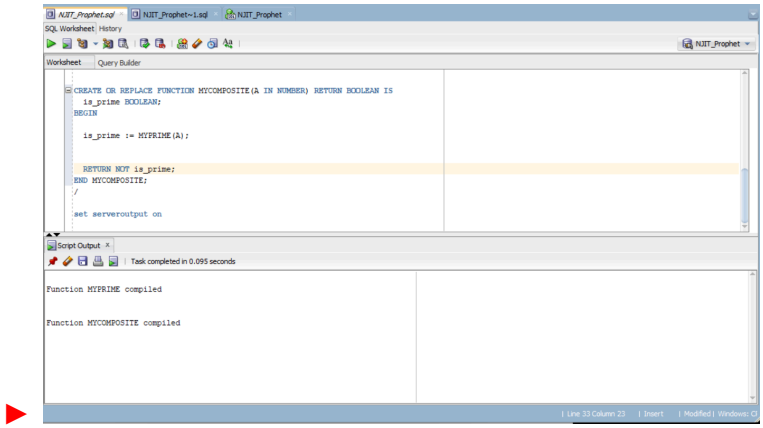


c) A **composite number** is a positive integer that can be formed by multiplying two smaller positive integers. Equivalently, it is a positive integer that has at least one **divisor** other than 1 and itself.

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
```



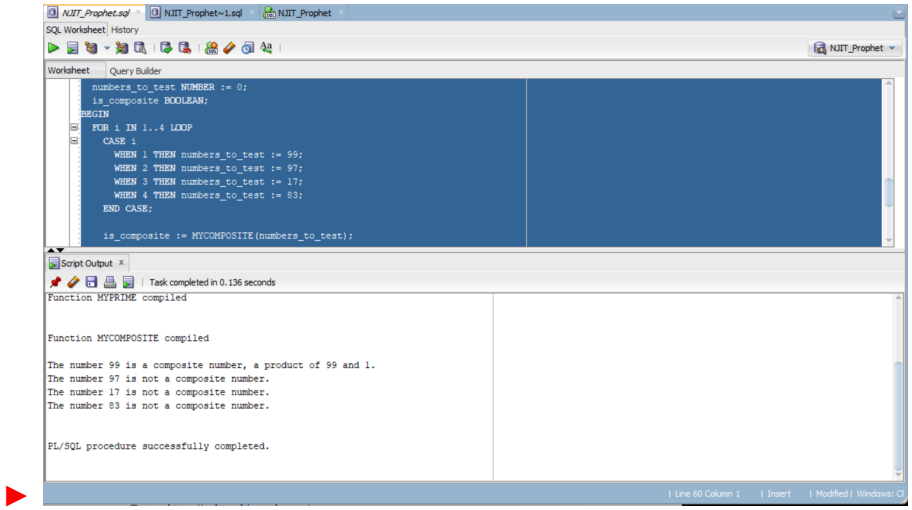
d) Call MYCOMPOSITE from a main program with n=99, n=97, n=17, and n=83. The main program should send the results to the screen. The results should be self-documenting.

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
```



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]

2) 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.

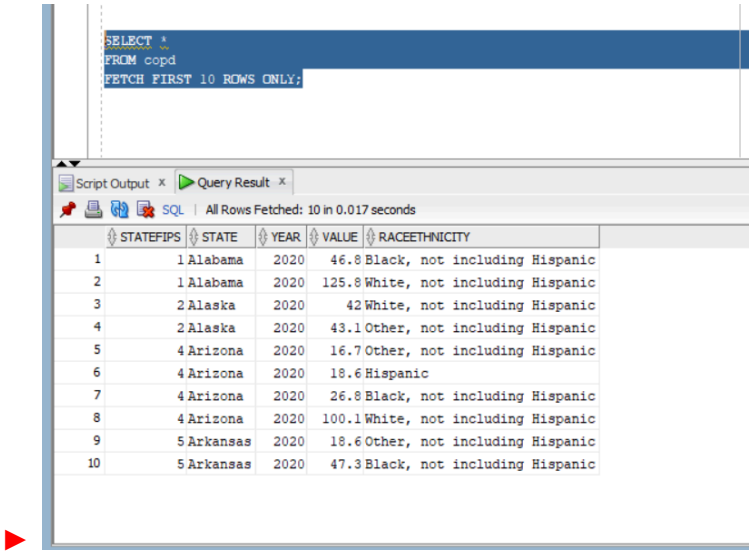
Now Clean the Data as suggested in class.

3) 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;
```



```
SELECT *
FROM copd
FETCH FIRST 10 ROWS ONLY;
```

Script Output | Query Result

SQL | All Rows Fetched: 10 in 0.017 seconds

	STATEFIPS	STATE	YEAR	VALUE	RACEETHNICITY
1	1	Alabama	2020	46.8	Black, not including Hispanic
2	1	Alabama	2020	125.8	White, not including Hispanic
3	2	Alaska	2020	42	White, not including Hispanic
4	2	Alaska	2020	43.1	Other, not including Hispanic
5	4	Arizona	2020	16.7	Other, not including Hispanic
6	4	Arizona	2020	18.6	Hispanic
7	4	Arizona	2020	26.8	Black, not including Hispanic
8	4	Arizona	2020	100.1	White, not including Hispanic
9	5	Arkansas	2020	18.6	Other, not including Hispanic
10	5	Arkansas	2020	47.3	Black, not including Hispanic

4) 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.

5) 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).

6) 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.

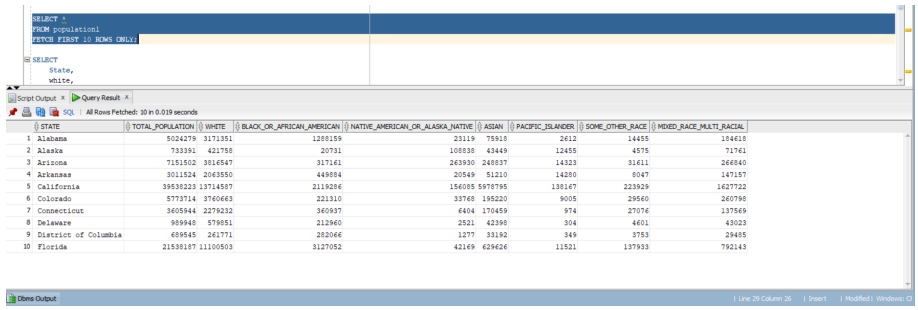
   This is called a problem of different grain size. POPULATION1 is more fine-grained.

   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;
```



	STATE	TOTAL_POPULATION	WHITE	BLACK_OR_AFRICAN_AMERICAN	NATIVE_AMERICAN_OR_ALASKA_NATIVE	ASIAN	PACIFIC_ISLANDER	SOME_OTHER_RACE	MIXED_RACE_MULTI_RACIAL
1	Alabama	5024279	3171351	1288159	23119	75918	2612	14455	184618
2	Alaska	733391	421758	20731	108838	43449	12455	4575	71761
3	Arizona	7151502	3816547	317161	263930	248837	14323	31611	266840
4	Arkansas	3011524	2063550	449854	20549	51210	14280	8047	147157
5	California	39538223	13714587	2119286	156085	5978795	138167	223929	1627722
6	Colorado	5773714	3760663	221310	33768	195220	9005	29560	260798
7	Connecticut	3605944	2279232	360937	6404	170459	974	27076	137569
8	Delaware	989948	579851	212960	2521	42398	304	4601	43023
9	District of Columbia	689545	261771	282066	1277	33192	349	3753	29485
10	Florida	21538187	11100503	3127052	42169	629626	11521	137933	792143

▶

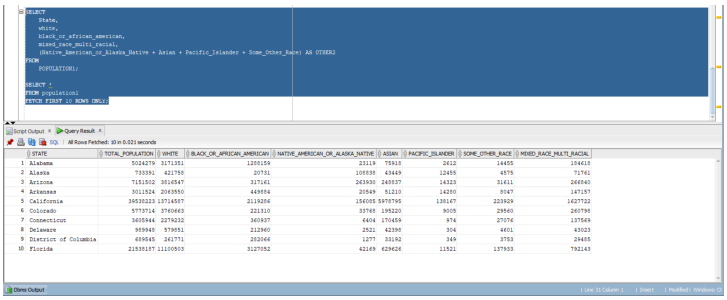
7) Write a SELECT statement against POPULATION1 that returns the columns State, White, Black, Mixed, and a new computed column OTHER2 that contains the SUM of Native…, Asian, Pacific…, Some Other… So, there will be 5 columns in the answer. [5 points]

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]

▶

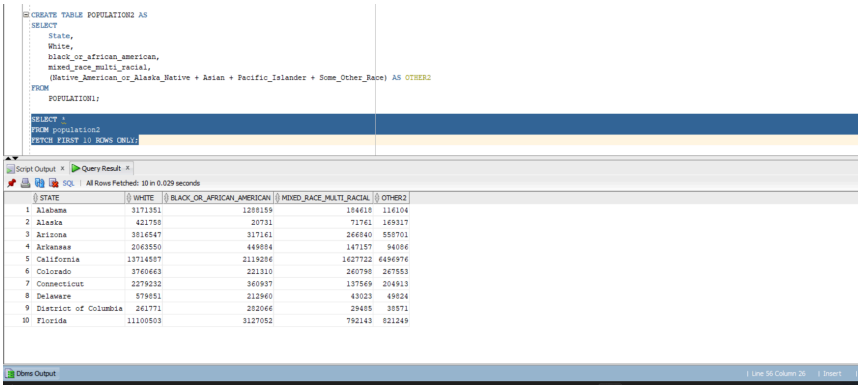
```
SELECT
    State,
    white,
    black_or_african_american,
    mixed_race_multi_racial,
    (Native_American_or_Alaska_Native + Asian + Pacific_Islander + Some_Other_Race) AS OTHER2
FROM
    POPULATION1;

SELECT *
FROM population1
FETCH FIRST 10 ROWS ONLY;
```



8) 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;
```



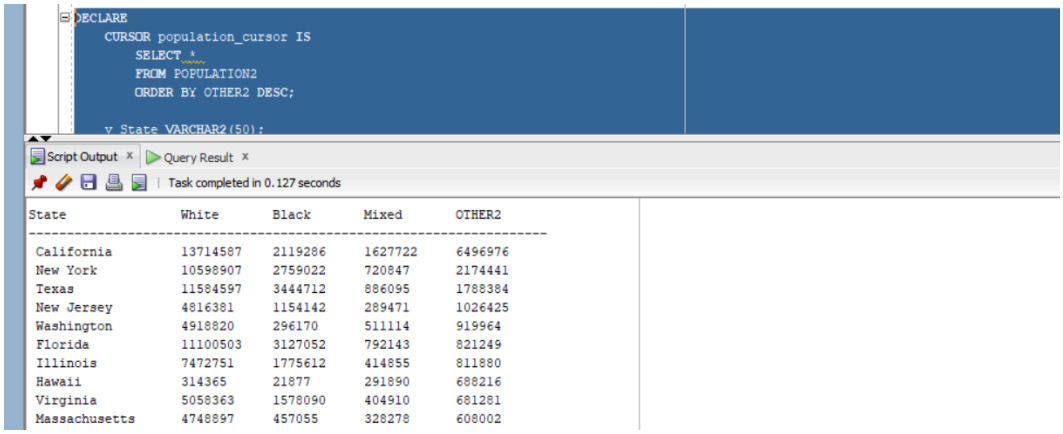
9) Write a PL/SQL program using an **explicit** cursor that displays all columns sorted by OTHER2 in descending order in the table POPULATION2.

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;
/
```

▶ First 10:



```
DECLARE
    CURSOR population_cursor IS
        SELECT *
        FROM POPULATION2
        ORDER BY OTHER2 DESC;

    v_State VARCHAR2(50);
```

Script Output × | Query Result ×

Task completed in 0.127 seconds

State	White	Black	Mixed	OTHER2
California	13714587	2119286	1627722	6496976
New York	10598907	2759022	720847	2174441
Texas	11584597	3444712	886095	1788384
New Jersey	4816381	1154142	289471	1026425
Washington	4918820	296170	511114	919964
Florida	11100503	3127052	792143	821249
Illinois	7472751	1775612	414855	811880
Hawaii	314365	21877	291890	688216
Virginia	5058363	1578090	404910	681281
Massachusetts	4748897	457055	328278	608002

Last 10:

Mississippi	1639077	1079001	83446	54535
Rhode Island	754050	55386	52250	53592
North Dakota	636160	26152	30248	53122
Delaware	579851	212960	43023	49824
New Hampshire	1200649	18655	54564	44207
District of Columbi	261771	282066	29485	38571
Maine	1228264	25115	53573	28798
West Virginia	1598834	64749	72135	23171
Wyoming	469664	4735	23674	19732
Vermont	573201	8649	29549	16174

10) 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;
```

▶

	STATE	VALUE	YEAR	RACEETHNICITY	WHITE	BLACK_OR_AFRICAN_AMERICAN	MIXED_RACE_MULTI_RACIAL	OTHER2
1	Alabama	46.8	2020	Black, not including Hispanic	3171351	1288159	184618	116104
2	Alabama	125.8	2020	White, not including Hispanic	3171351	1288159	184618	116104
3	Alaska	42	2020	White, not including Hispanic	(null)	(null)	(null)	(null)
4	Alaska	43.1	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
5	Arizona	16.7	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
6	Arizona	18.6	2020	Hispanic	(null)	(null)	(null)	(null)
7	Arizona	26.8	2020	Black, not including Hispanic	(null)	(null)	(null)	(null)
8	Arizona	100.1	2020	White, not including Hispanic	(null)	(null)	(null)	(null)
9	Arkansas	18.6	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
10	Arkansas	47.3	2020	Black, not including Hispanic	(null)	(null)	(null)	(null)

11) 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);
```

▶

	STATE	VALUE	YEAR	RACEETHNICITY	WHITE	BLACK_OR_AFRICAN_AMERICAN	MIXED_RACE_MULTI_RACIAL	OTHER2
1	Alabama	46.8	2020	Black, not including Hispanic	3171351	1288159	184618	116104
2	Alabama	125.8	2020	White, not including Hispanic	3171351	1288159	184618	116104
3	Alaska	42	2020	White, not including Hispanic	(null)	(null)	(null)	(null)
4	Alaska	43.1	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
5	Arizona	16.7	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
6	Arizona	18.6	2020	Hispanic	(null)	(null)	(null)	(null)
7	Arizona	26.8	2020	Black, not including Hispanic	(null)	(null)	(null)	(null)
8	Arizona	100.1	2020	White, not including Hispanic	(null)	(null)	(null)	(null)
9	Arkansas	18.6	2020	Other, not including Hispanic	(null)	(null)	(null)	(null)
10	Arkansas	47.3	2020	Black, not including Hispanic	(null)	(null)	(null)	(null)

12) 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

```
State | Total Value
------------------
Alabama               172.60
Alaska                 85.10
Arizona               162.20
Arkansas              206.80
California            169.90
Colorado              154.40
Connecticut           104.50
Delaware              114.60
District of Columbia   61.90
Florida               176.20
```

Last 10

```
South Dakota          110.50
Tennessee             158.90
Texas                 150.50
Utah                   78.90
Vermont                77.40
Virginia              123.70
Washington            107.00
West Virginia         185.00
Wisconsin             149.70
Wyoming               106.90
```