FALL 2023 CS 434-001

**HOMEWORK 4**

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 **December 11, 2023, 4:00 PM.**

Your file name must have the format:

**LASTNAME\_firstname\_HWK4.doc (or .docx)**

In this homework, we will continue to use health data from the government.

1. You are asked to create a JSON object using a COPD.csv file from Homework1.

The first step is to CLEAN the data. You only need state names, value, and Race Ethnicity attributes; delete the other columns. Replace “Suppressed” values with “null’ in the “values” attribute. Also, replace “White, not including Hispanic” with “white” and other Race Ethnicity strings accordingly. Load the cleaned data into an Oracle table COPD using SQL Developer.

Create a table CPOD\_JSON in Oracle. It should contain a single-column COPD. Then, write a PL/SQL code using an **implicit cursor** that reads from the COPD table and write into a COPD\_JSON file that contains all data. Each state should have 6 key-value pairs; \_id, state, white, black, hispanic, and other. The COPD table has 204 rows whereas the COPD\_JSON table should have 51 rows.

Copy and paste the complete PL/SQL code at the first red arrow below. (Edit the code so that it is NICELY indented.)

Do a SNIP or screen dump to show the first 10 rows of the COPD\_JSON file at the second arrow [5].

► set serveroutput on;

CREATE TABLE COPD\_JSON (

COPD CLOB

);

DECLARE

v\_json CLOB;

v\_last\_state VARCHAR2(100);

v\_counter NUMBER := 0;

FUNCTION GenerateStateJSON(p\_state VARCHAR2, p\_id NUMBER) RETURN CLOB IS

v\_state\_json CLOB;

BEGIN

v\_state\_json := '{';

v\_state\_json := v\_state\_json || '"\_id":"' || p\_id || '",';

v\_state\_json := v\_state\_json || '"state":"' || p\_state || '",';

SELECT

v\_state\_json || '"white":' || COALESCE(SUM(CASE WHEN Race = 'White' THEN Value END), 0) || ','

|| '"black":' || COALESCE(SUM(CASE WHEN Race = 'Black' THEN Value END), 0) || ','

|| '"hispanic":' || COALESCE(SUM(CASE WHEN Race = 'Hispanic' THEN Value END), 0) || ','

|| '"other":' || COALESCE(SUM(CASE WHEN Race = 'Other' THEN Value END), 0)

INTO v\_state\_json

FROM COPD

WHERE State = p\_state;

v\_state\_json := v\_state\_json || '}';

RETURN v\_state\_json;

END;

BEGIN

SELECT MAX(State) INTO v\_last\_state FROM COPD;

v\_json := '[';

FOR c\_rec IN (SELECT DISTINCT State FROM COPD ORDER BY State) LOOP

v\_counter := v\_counter + 1;

v\_json := v\_json || GenerateStateJSON(c\_rec.State, v\_counter);

IF c\_rec.State <> v\_last\_state THEN

v\_json := v\_json || ',';

END IF;

END LOOP;

v\_json := v\_json || ']';

INSERT INTO COPD\_JSON (COPD) VALUES (v\_json);

COMMIT;

END;

/

► A screenshot of a computer

Description automatically generated

**2-a)** Create a MongoDB collection COPD that contains exactly the same information as the COPD\_JSON objects.

Show the insertMany operation (10 lines) after the first red arrow.

Show the result of "the MongoDB version" of a **Select \* from COPD** at the second red arrow [5].

*Write a MongoDB query for the following (use the COPD collection).*

*Use “table view” in Studio3T to display the first 10 documents for each query.*

**2-b)** Write a MongoDB query to display state, white, and hispanic fields if the hispanic is more than or equal to 10 and the white is less than 75. Copy-paste the query, and show the result at the third red arrow [3].

**2-c)** Write a MongoDB query to display state, white, and hispanic fields if the hispanic is more than or equal to 10 or the white is less than 75. Copy-paste the query, and show the result at the fourth red arrow [3].

**2-d)** Write a MongoDB query to sort CPOD collection by the number of white COPD cases in ascending order for state names starting with A, C, M, W. Copy-paste the query, and show the result at the fifth red arrow [3].

**2-e)** Define a variable “**healtyStates**” in which hispanic, black, and other COPD values are null. Use the variable **healtyStates** to list all healthy states. Copy-paste the queries, and show the result at the sixth red arrow [3].

**2-f)** Insert a new field “**HealthInsurance**” and set it to “lower” for all **healtyStates**. Copy-paste the query, and show the result at the seventh red arrow. [3].

► db.COPD.insertMany([

{"\_id": 1, "state": "Alabama", "white": 46.8, "black": 125.8, "hispanic": 0.0, "other": 0.0},

{"\_id": 2, "state": "Alaska", "white": 42.0, "black": 43.1, "hispanic": 0.0, "other": 0.0},

{"\_id": 3, "state": "Arizona", "white": 16.7, "black": 18.6, "hispanic": 100.1, "other": 26.8},

{"\_id": 4, "state": "Arkansas", "white": 18.6, "black": 47.3, "hispanic": 0.0, "other": 140.9},

{"\_id": 5, "state": "California", "white": 14.1, "black": 19.0, "hispanic": 81.6, "other": 55.2},

{"\_id": 6, "state": "Colorado", "white": 20.7, "black": 24.1, "hispanic": 71.9, "other": 37.7},

{"\_id": 7, "state": "Connecticut", "white": 8.6, "black": 13.2, "hispanic": 60.4, "other": 22.3},

{"\_id": 8, "state": "Delaware", "white": 0.0, "black": 27.1, "hispanic": 0.0, "other": 87.5},

{"\_id": 9, "state": "District of Columbia", "white": 0.0, "black": 0.0, "hispanic": 51.5, "other": 10.4},

{"\_id": 10, "state": "Florida", "white": 12.1, "black": 27.6, "hispanic": 106.5, "other": 30.0}

]);

► db.COPD.find({});

A screenshot of a computer program

Description automatically generated

2-b ► db.COPD.find(

{

"hispanic": { "$gte": 10 }, // hispanic >= 10

"white": { "$lt": 75 } // white < 75

},

{

"state": 1, "white": 1, "hispanic": 1, "\_id": 0

}

);

► A screenshot of a computer

Description automatically generated

2-c ► db.COPD.find(

{

"$or": [

{ "hispanic": { "$gte": 10 } },

{ "white": { "$lt": 75 } }

]

},

{

"state": 1, "white": 1, "hispanic": 1, "\_id": 0

}

A screenshot of a computer

Description automatically generated);

►

2-d ► db.COPD.find(

{ "state": { "$regex": "^(A|C|M|W)" } }

).sort(

{ "white": 1 }

);

► (as only first 10 rows were inserted):

A screenshot of a computer

Description automatically generated

2-e ► var healthyStates = {

"$or": [

{ "hispanic": { "$eq": null } },

{ "hispanic": { "$eq": 0 } },

{ "black": { "$eq": null } },

{ "black": { "$eq": 0 } },

{ "other": { "$eq": null } },

{ "other": { "$eq": 0 } }

]

};

db.COPD.find(healthyStates, { "state": 1, "\_id": 0 });

► A screenshot of a computer

Description automatically generated

2-f ► db.COPD.updateMany(

healthyStates,

{ "$set": { "HealthInsurance": "lower" } }

);

**3-** Create a table POPULATION4\_JSON in Oracle. It should contain a single-column POPULATION. Then, write a PL/SQL code using an **implicit cursor** that reads from the **COPDobject** (from Homework3, question2a) table and write into a POPULATION4\_JSON file that contains all data but each state should be a JSON object as follows.

{"state":"Alabama",

"Year2017":{"white":131, "black":42.2, "multiR":null, "other":null},

"Year2018":{"white":134.6, "black":44.9, "multiR":10.2, "other":null},

"Year2019":{"white":131.2, "black":44.3, "multiR":null, "other":17.3},

"Year2020":{"white":125.8, "black":46.8, "multiR":null, "other":null}}

Both the **COPDobject** and POPULATION4\_JSON tables have 51 rows. All object values in the **COPDobject** table are stored as a nested JSON object in the POPULATION4\_JSON table.

Copy and paste the complete PL/SQL code at the first red arrow below. (Edit the code so that it is NICELY indented.)

Do a SNIP or screen dump to show the first 10 rows of the POPULATION4\_JSON file at the second arrow [5].

► DECLARE

v\_state\_json CLOB;

-- Function to convert race\_o object to JSON

FUNCTION raceToJson(p\_race race\_o) RETURN VARCHAR2 IS

BEGIN

RETURN '"white":' || COALESCE(TO\_CHAR(p\_race.White), 'null') || ','

|| '"black":' || COALESCE(TO\_CHAR(p\_race.Black), 'null') || ','

|| '"multiR":' || COALESCE(TO\_CHAR(p\_race.Hispanic), 'null') || ','

|| '"other":' || COALESCE(TO\_CHAR(p\_race.Others), 'null');

END;

BEGIN

FOR c\_rec IN (SELECT State, year2017, year2018, year2019, year2020 FROM COPDobject) LOOP

v\_state\_json := '{';

v\_state\_json := v\_state\_json || '"state":"' || c\_rec.State || '",';

v\_state\_json := v\_state\_json || '"Year2017":{' || raceToJson(c\_rec.year2017) || '},';

v\_state\_json := v\_state\_json || '"Year2018":{' || raceToJson(c\_rec.year2018) || '},';

v\_state\_json := v\_state\_json || '"Year2019":{' || raceToJson(c\_rec.year2019) || '},';

v\_state\_json := v\_state\_json || '"Year2020":{' || raceToJson(c\_rec.year2020) || '}';

v\_state\_json := v\_state\_json || '}';

-- Insert each state's JSON object into POPULATION4\_JSON table

INSERT INTO POPULATION4\_JSON (POPULATION) VALUES (v\_state\_json);

END LOOP;

COMMIT;

END;

/

► A screenshot of a computer

Description automatically generated

**4**- a) Create a MongoDB collection **population** that contains exactly the same information as the POPULATION4\_JSON objects.

Show the insertMany operation (10 lines) after the first red arrow.

Show the result of "the MongoDB version" of a **Select \* from population** at the second red arrow. [5]

*Write a MongoDB query for the following (use the* ***population*** *collection).*

*Use “table view” in Studio3T to display the first 10 documents for each query.*

**4-b)** Write a MongoDB query to display only the state names and Year2017 nested object if black, multiracial, and other value is null in Year2017 field. How many states did you list? Copy-paste the query, and show the result at the third red arrow [3].

**4-c)** Write a MongoDB query to find the summation values of all keys in Year2020.

Your query should return 4 sum values for Year2020; namely whiteSum, blackSum, otherSum, and multiSum. Copy-paste the query, and show the result at the forth red arrow [3].

**4-d)** Write a MongoDB query to find the summation values of all whites in all four years. Your query should return 4 sum values for whites; namely Sum2017, Sum2018, Sum2019, and Sum2020. Copy-paste the query, and show the result at the fifth red arrow [3].

**4-e)** Write a MongoDB query that has all data of Year2017 key-value pairs from the population collection and writes the result into a new collection named Year2017. The Year2017 collection should have only one document where the value is an array with 51 elements. Copy-paste the query, and show the result at the sixth red arrow [3].

**4-f)** Write a MongoDB query to display only the black values in Year2017 collection. Copy-paste the query, and show the result at the seventh red arrow [3].

4-a► db.population2.insertMany([

{"state":"Alabama","Year2017":{"white":131,"black":42.2,"multiR":0,"other":0},"Year2018":{"white":134.6,"black":44.9,"multiR":10.2,"other":0},"Year2019":{"white":131.2,"black":44.3,"multiR":0,"other":17.3},"Year2020":{"white":125.8,"black":46.8,"multiR":0,"other":0}},

{"state":"Alaska","Year2017":{"white":42.2,"black":0,"multiR":0,"other":50.3},"Year2018":{"white":45.8,"black":0,"multiR":0,"other":45},"Year2019":{"white":43.8,"black":0,"multiR":0,"other":38.9},"Year2020":{"white":42,"black":0,"multiR":0,"other":43.1}},

{"state":"Arizona","Year2017":{"white":111.2,"black":28,"multiR":16.5,"other":21},"Year2018":{"white":109.9,"black":35.1,"multiR":16.2,"other":19.9},"Year2019":{"white":102.7,"black":32.1,"multiR":17.2,"other":14.8},"Year2020":{"white":100.1,"black":26.8,"multiR":18.6,"other":16.7}},

{"state":"Arkansas","Year2017":{"white":146.5,"black":55.5,"multiR":0,"other":22.2},"Year2018":{"white":141,"black":42.6,"multiR":9.5,"other":0},"Year2019":{"white":139,"black":41.1,"multiR":0,"other":0},"Year2020":{"white":140.9,"black":47.3,"multiR":0,"other":18.6}},

{"state":"California","Year2017":{"white":88.8,"black":53.9,"multiR":14.2,"other":20.6},"Year2018":{"white":86.7,"black":53.1,"multiR":15.1,"other":19.7},"Year2019":{"white":84,"black":49.7,"multiR":13.7,"other":19.5},"Year2020":{"white":81.6,"black":55.2,"multiR":14.1,"other":19}},

{"state":"Colorado","Year2017":{"white":80.3,"black":38.6,"multiR":24.6,"other":22.3},"Year2018":{"white":79.4,"black":43.4,"multiR":23.6,"other":23.1},"Year2019":{"white":74.8,"black":34.5,"multiR":25.4,"other":14},"Year2020":{"white":71.9,"black":37.7,"multiR":24.1,"other":20.7}},

{"state":"Connecticut","Year2017":{"white":72.7,"black":29,"multiR":10.7,"other":11.5},"Year2018":{"white":69.7,"black":30,"multiR":12.7,"other":0},"Year2019":{"white":70.1,"black":24.3,"multiR":11.2,"other":0},"Year2020":{"white":60.4,"black":22.3,"multiR":13.2,"other":8.6}},

{"state":"Delaware","Year2017":{"white":96.8,"black":43.8,"multiR":0,"other":0},"Year2018":{"white":98.1,"black":39.1,"multiR":0,"other":0},"Year2019":{"white":87.8,"black":36.1,"multiR":0,"other":0},"Year2020":{"white":87.5,"black":27.1,"multiR":0,"other":0}},

{"state":"District of Columbia","Year2017":{"white":16.2,"black":42.6,"multiR":0,"other":0},"Year2018":{"white":17.4,"black":46.1,"multiR":0,"other":0},"Year2019":{"white":12.4,"black":44.3,"multiR":0,"other":0},"Year2020":{"white":10.4,"black":51.5,"multiR":0,"other":0}},

{"state":"Florida","Year2017":{"white":116.7,"black":33.2,"multiR":32.8,"other":19},"Year2018":{"white":114.4,"black":31.2,"multiR":29.3,"other":15.7},"Year2019":{"white":110.2,"black":29.9,"multiR":28.5,"other":13},"Year2020":{"white":106.5,"black":30,"multiR":27.6,"other":12.1}}

]);

db.population2.find({});

► A screenshot of a computer

Description automatically generated

4-b ► db.population2.find(

{

"Year2017.black": 0,

"Year2017.multiR": 0,

"Year2017.other": 0

},

{

"state": 1,

"Year2017": 1,

"\_id": 0

}

);

► 8 states:

A screenshot of a computer

Description automatically generated

4-c ► db.population2.aggregate([

{

$group: {

\_id: null,

whiteSum: { $sum: "$Year2020.white" },

blackSum: { $sum: "$Year2020.black" },

otherSum: { $sum: "$Year2020.other" },

multiSum: { $sum: "$Year2020.multiR" }

}

},

{

$project: {

\_id: 0,

whiteSum: 1,

blackSum: 1,

otherSum: 1,

multiSum: 1

}

}

]);

► A screen shot of a computer

Description automatically generated

4-d ► db.population2.aggregate([

{

$group: {

\_id: null,

Sum2017: { $sum: "$Year2017.white" },

Sum2018: { $sum: "$Year2018.white" },

Sum2019: { $sum: "$Year2019.white" },

Sum2020: { $sum: "$Year2020.white" }

}

},

{

$project: {

\_id: 0,

Sum2017: 1,

Sum2018: 1,

Sum2019: 1,

Sum2020: 1

}

}

]);

A screen shot of a computer

Description automatically generated►

4-e ► db.population2.aggregate([

{

$group: {

\_id: null,

Year2017Data: {

$push: "$Year2017"

}

}

},

{

$project: {

\_id: 0,

Year2017: "$Year2017Data"

}

},

{

$out: "Year2017"

}

]);

►A screenshot of a computer

Description automatically generated

4-f ► db.Year2017.aggregate([

{

$unwind: "$Year2017"

},

{

$project: {

\_id: 0,

black: "$Year2017.black"

}

}

]);

► A screenshot of a computer

Description automatically generated