# Overview of the Assignment:

Part 1 of this assignment has you working with Analytical functions in SQL and gives you a bit of a taste in what you can do to transform data in SQL. Part 2 goes into an introduction to Python. Make sure to download all of the Python-related files into the same folder.

# Part 1 – Analytical/Windowed Functions

You will be working with the us\_national\_statistics database that you restored in assignment 1-A. The primary table is person\_economic\_info. Each row describes a person sampled from that state. It has the following columns:

|  |  |
| --- | --- |
| **Column\_name** | **Column\_description** |
| age | Person’s age in years |
| marital\_status | Indicates whether the person is married |
| address\_state | Numeric code refers to ‘states’ table |
| income | Annual income in USD |
| income\_category | Categorized income |
| car\_price | Price of car in USD |
| car\_price\_category | Categorized car price |
| education | Numeric code refers to education\_codes table |
| years\_employed | Years of continuous employment for person |
| retired | Indicates person is retired |
| employment\_category | Categorizes type of most recent employment for person per employment\_categories table |
| gender | ‘m’ or ‘f’ for male or female |
| length\_at\_current\_residence | Years person has lived at current residence |
| wireless | Indicates person owns a wireless phone |
| multiple\_lines | Indicates person has multiple voice phone lines |
| voice\_mail | Indicates person has voice mail |
| pager | Indicates person has a pager |
| internet | Indicates person has a dedicated residential internet connection rather than cellular or dial-up |
| caller\_id | Indicates person has caller id service on voice line |
| call\_waiting | Indicates person has call waiting service on voice line |
| own\_tv | Indicates person owns a television set |
| own\_dvd\_player | Indicates person owns a DVD player |
| own\_smartphone | Indicates person owns a smartphone |
| own\_computer | Indicates person owns a personal computer |
| own\_fax | Indicates person has a fax send/receive device on a phone line |
| read\_newspapers | Indicates person reads physical newspapers |

Description tables augmenting this table are:

|  |  |
| --- | --- |
| **Table name** | **Usage** |
| states | Translates numeric state codes to actual states |
| employment\_categories | Translates numeric employment categories to descriptions |
| education\_codes | Translates numeric education codes to descriptions |

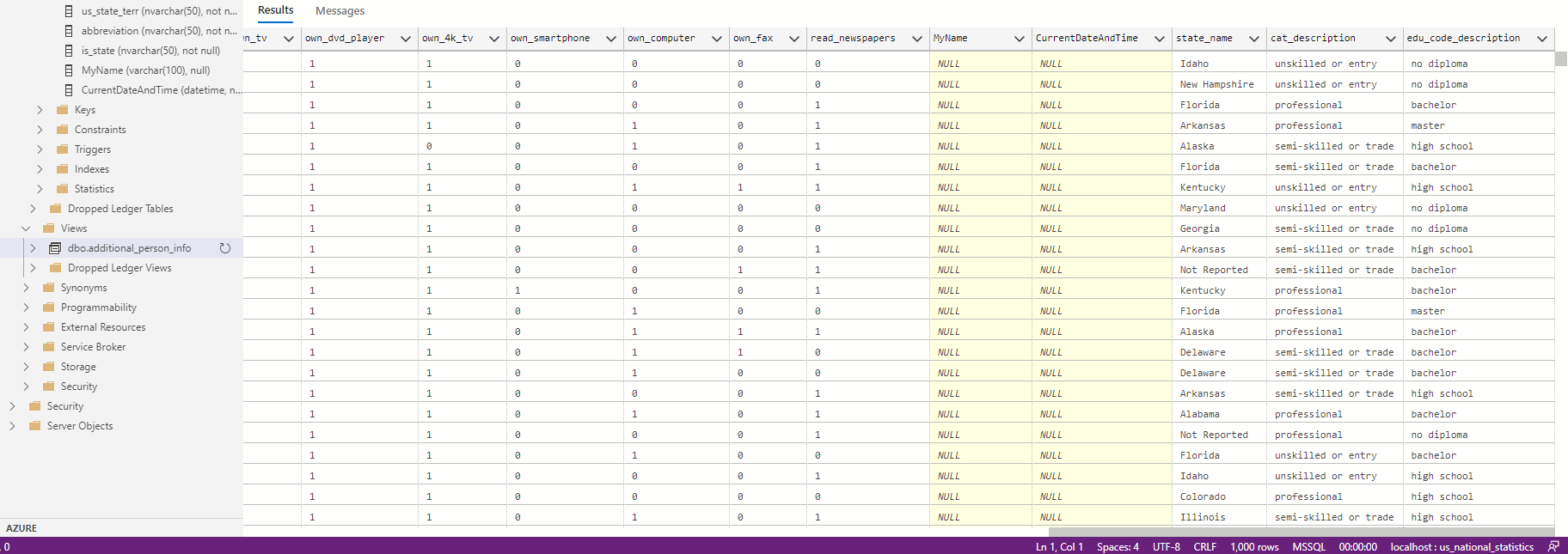
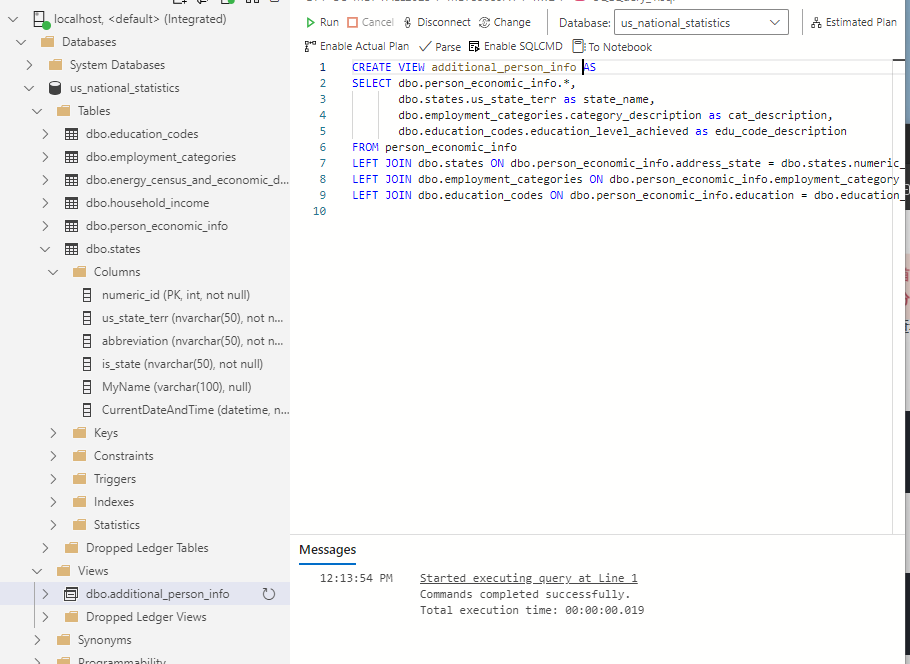
With each question, write ONE query that answers the question. Put the text of the query in your submission document, along with a screen shot of the results from running the query.

1. Create a view named additional\_person\_info
   * This view should give the *names* of states, employment categories, and education levels along with all the other fields in the person\_economic\_info table for use in later reports.
   * Hint: you don’t need to specify each field individually for the person\_economic\_info table, think about how to select all the columns from the table. Once you have your query constructed, turn it into a view.

**Text of query**:

|  |
| --- |
| CREATE VIEW additional\_person\_info AS  SELECT dbo.person\_economic\_info.\*,         dbo.states.us\_state\_terr as state\_name,         dbo.employment\_categories.category\_description as cat\_description,         dbo.education\_codes.education\_level\_achieved as edu\_code\_description  FROM person\_economic\_info  LEFT JOIN dbo.states ON dbo.person\_economic\_info.address\_state = dbo.states.numeric\_id  LEFT JOIN dbo.employment\_categories ON dbo.person\_economic\_info.employment\_category = dbo.employment\_categories.employment\_category  LEFT JOIN dbo.education\_codes ON dbo.person\_economic\_info.education = dbo.education\_codes.code; |

**Screenshot of result**:

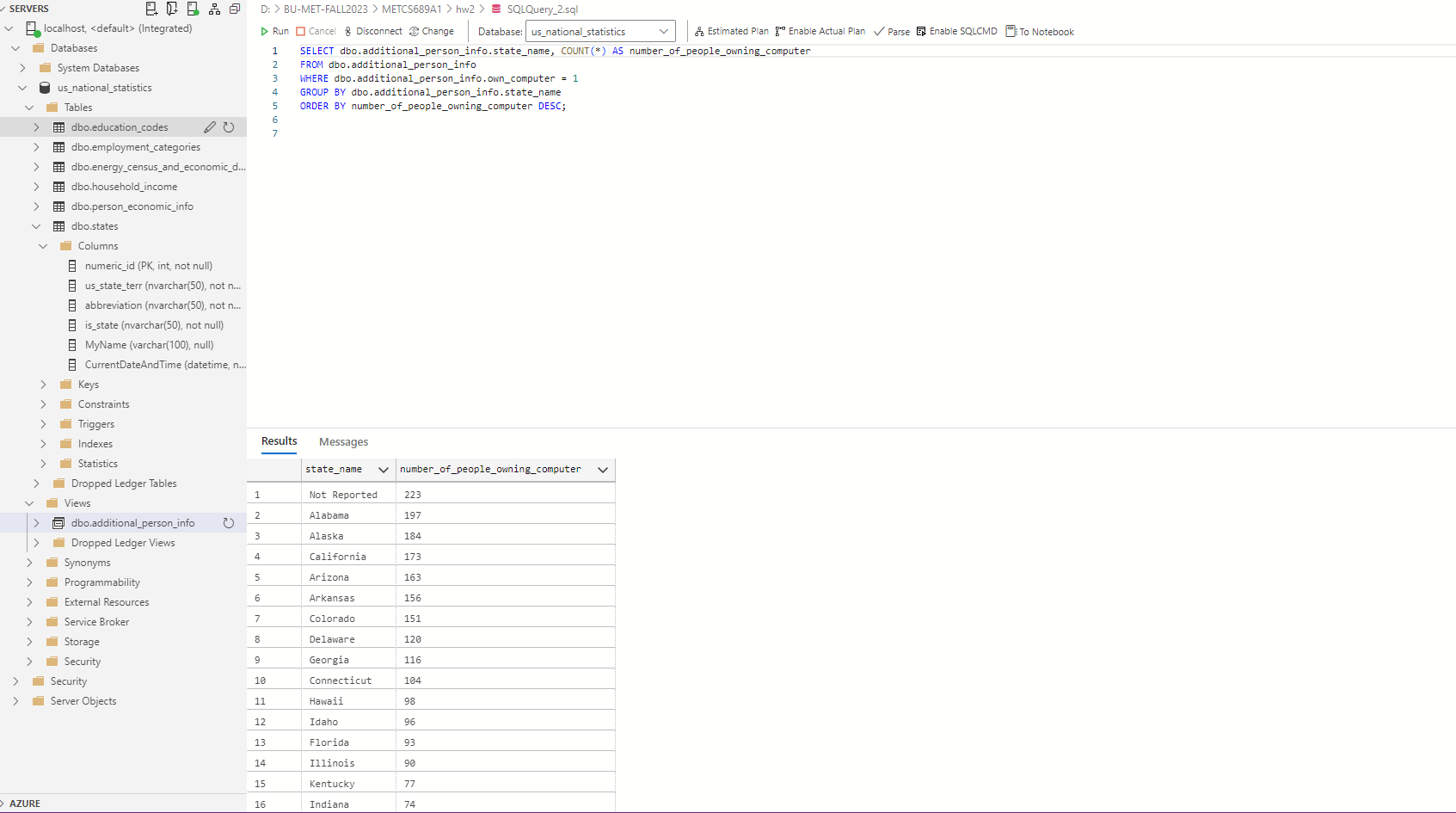


1. Using a single aggregate query to select from the additional\_person\_info view which shows states and how many people own a personal computer in each state.
   * Hints: think about aggregates such as count and sum, think about what to group by, consider the differences between where clause and having clause.

**Text of query**:

|  |
| --- |
| SELECT dbo.additional\_person\_info.state\_name, COUNT(\*) AS number\_of\_people\_owning\_computer  FROM dbo.additional\_person\_info  WHERE dbo.additional\_person\_info.own\_computer = 1  GROUP BY dbo.additional\_person\_info.state\_name  ORDER BY number\_of\_people\_owning\_computer DESC; |

**Screenshot of result:**

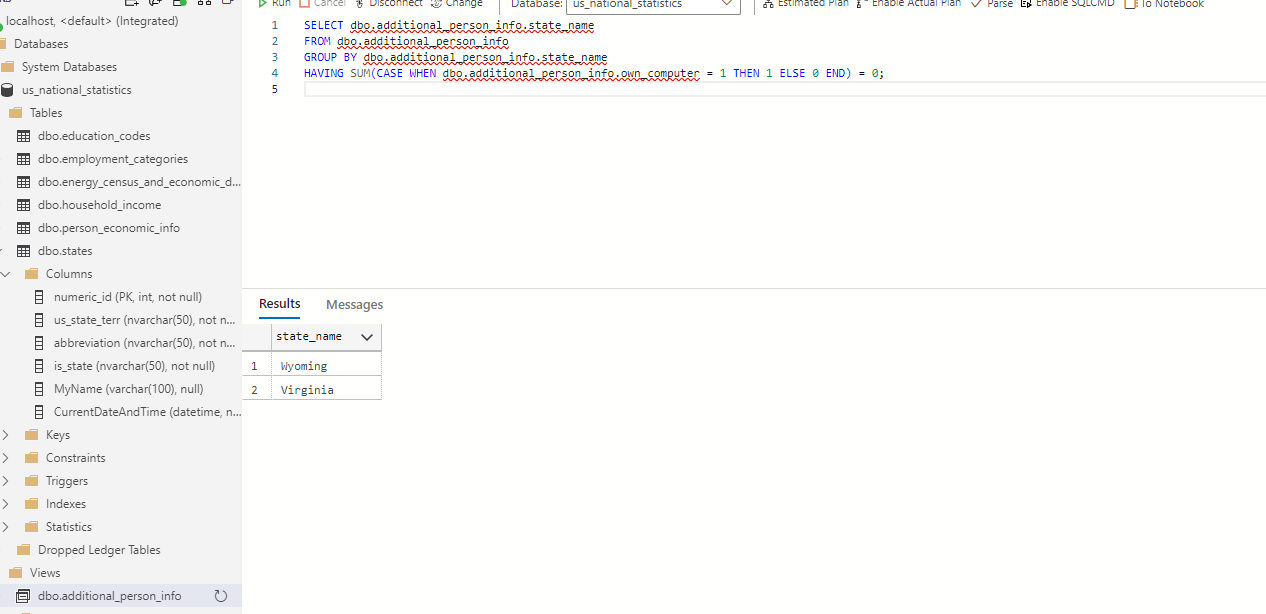
****

1. Modify the query above to show which states have respondents where no one owns a computer.
   * Hints: think about aggregates such as count and sum, think about what to group by, consider the differences between where clause and having clause.

**Text of query**:

|  |
| --- |
| SELECT dbo.additional\_person\_info.state\_name  FROM dbo.additional\_person\_info  GROUP BY dbo.additional\_person\_info.state\_name  HAVING SUM(CASE WHEN dbo.additional\_person\_info.own\_computer = 1 THEN 1 ELSE 0 END) = 0; |

**Screenshot of result:**



1. Use a single query to show each state’s aggregates by education level and all the attributes below. Use a CUBE to show subtotals by education level for each state, each state will be grouped together at the end.

The following attributes should be aggregated to show:

* + Number of people responding
  + Number of people who own computer
  + Average income

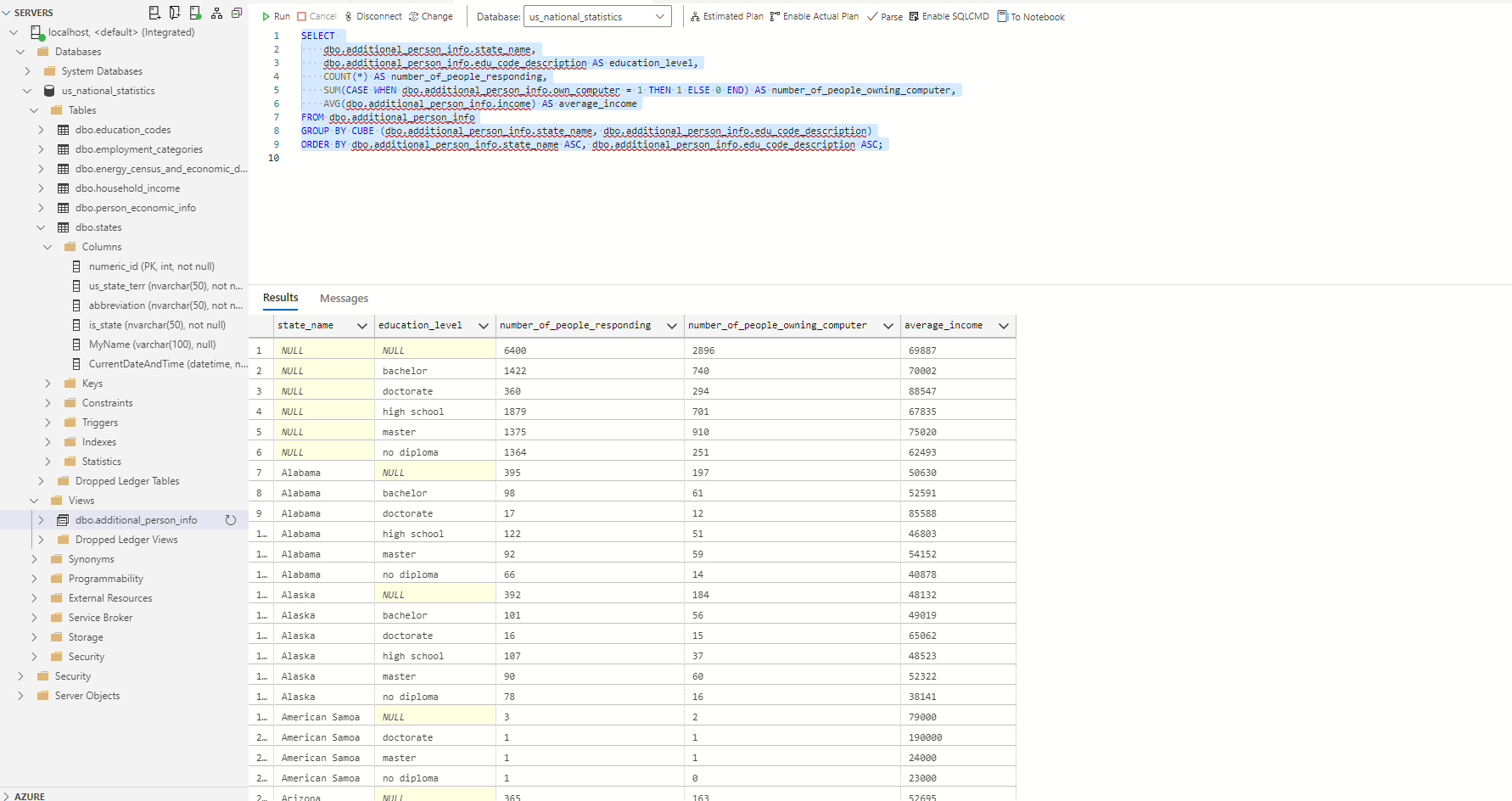
Hints:

* + Result should have five columns (state, education level, and the tree aggregates) and multiple rows for each state. For example, Alabama will have 6 rows, one for each education level and a subtotal for that state.
  + Write the aggregate first, then add the CUBE

**Text of query**:

|  |
| --- |
| SELECT      dbo.additional\_person\_info.state\_name,      dbo.additional\_person\_info.edu\_code\_description AS education\_level,      COUNT(\*) AS number\_of\_people\_responding,      SUM(CASE WHEN dbo.additional\_person\_info.own\_computer = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_owning\_computer,      AVG(dbo.additional\_person\_info.income) AS average\_income  FROM dbo.additional\_person\_info  GROUP BY CUBE (dbo.additional\_person\_info.state\_name, dbo.additional\_person\_info.edu\_code\_description)  ORDER BY dbo.additional\_person\_info.state\_name ASC, dbo.additional\_person\_info.edu\_code\_description ASC; |

**Screenshot of result**:

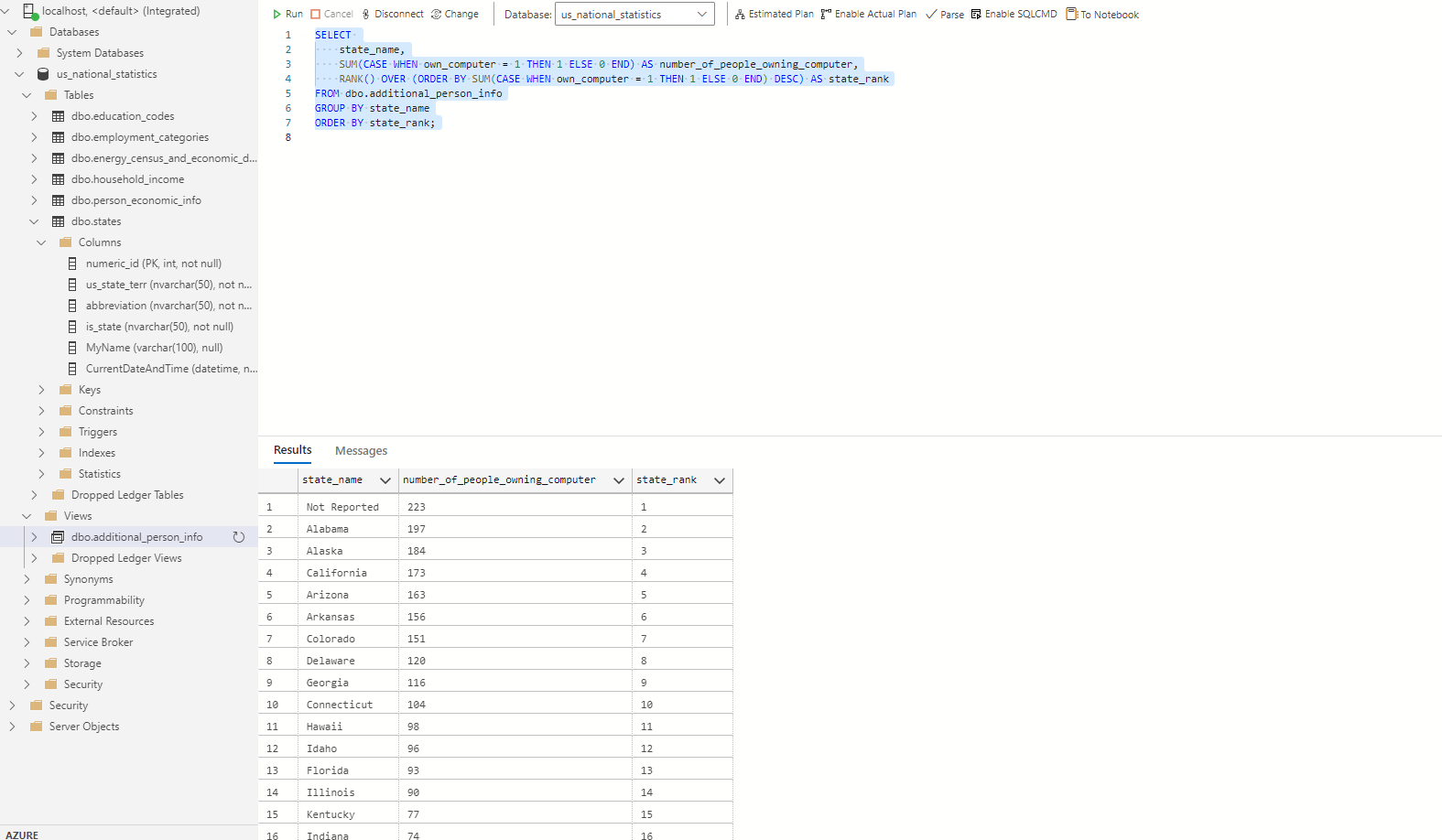


1. Modify the query above to show each state’s rank based on one of the aggregated columns below. Result will list each state once, the aggregated column, and its rank – 3 columns total. Order the result by the ranking of your choice. There is no need to CUBE the results.
   * People responding
   * Number of people who own computer
   * Average income

**Text of query**:

|  |
| --- |
| SELECT      state\_name,      SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_owning\_computer,      RANK() OVER (ORDER BY SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END) DESC) AS state\_rank  FROM dbo.additional\_person\_info  GROUP BY state\_name  ORDER BY state\_rank; |

**Screenshot of result**:

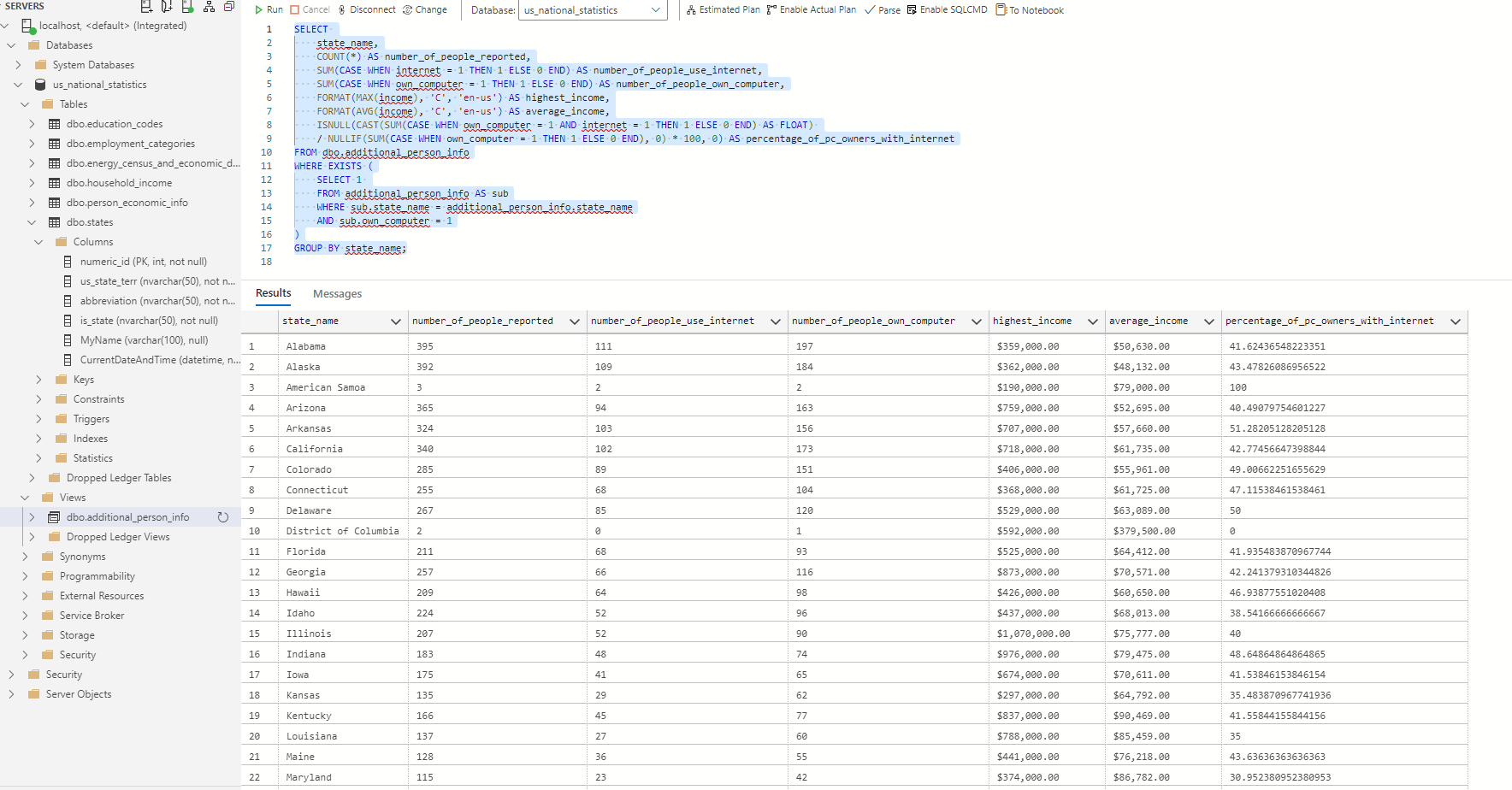


1. We want to look at statistics of states where there is at least one person using a computer. Using a single query select from the additional\_person\_info view: For each state, give the following summary information (result should have seven columns and one row for each state).
   1. State name
   2. Number of people reported
   3. Number of people who use the internet – Hint: review distinct values here, there will be a bit of transformation that you will need to do, look at case statements
   4. Number of people who own a personal computer
   5. Highest income (format as currency)
   6. Average income (format as currency)
   7. This is the challenging part: Of the people who own a personal computer, calculate the percentage of people who have a dedicated internet connection rather than dial-up (internet column).
      * Hint 1: you will need to cast your aggregates to evaluate this as decimals instead of integers.
      * Hint 2: look at ISNULL and NULLIF to solve the divide by zero issue
      * Hint 3: Consider solving this in steps, get columns a-f solved as an inline view, and then solve column g after.
      * Hint 4: If you are not able to figure out the divide by zero issue, you can filter out states where no one has a computer (the results of your last query can be an inline view or a CTE), this will be accepted as an alternate solution.
      * If you are totally stuck, run the query without g solved for partial credit.

**Text of query**:

|  |
| --- |
| SELECT      state\_name,      COUNT(\*) AS number\_of\_people\_reported,      SUM(CASE WHEN internet = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_use\_internet,      SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_own\_computer,      FORMAT(MAX(income), 'C', 'en-us') AS highest\_income,      FORMAT(AVG(income), 'C', 'en-us') AS average\_income,      ISNULL(CAST(SUM(CASE WHEN own\_computer = 1 AND internet = 1 THEN 1 ELSE 0 END) AS FLOAT)      / NULLIF(SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END), 0) \* 100, 0) AS percentage\_of\_pc\_owners\_with\_internet  FROM dbo.additional\_person\_info  WHERE EXISTS (      SELECT 1      FROM additional\_person\_info AS sub      WHERE sub.state\_name = additional\_person\_info.state\_name      AND sub.own\_computer = 1  )  GROUP BY state\_name; |

**Screenshot of result**:

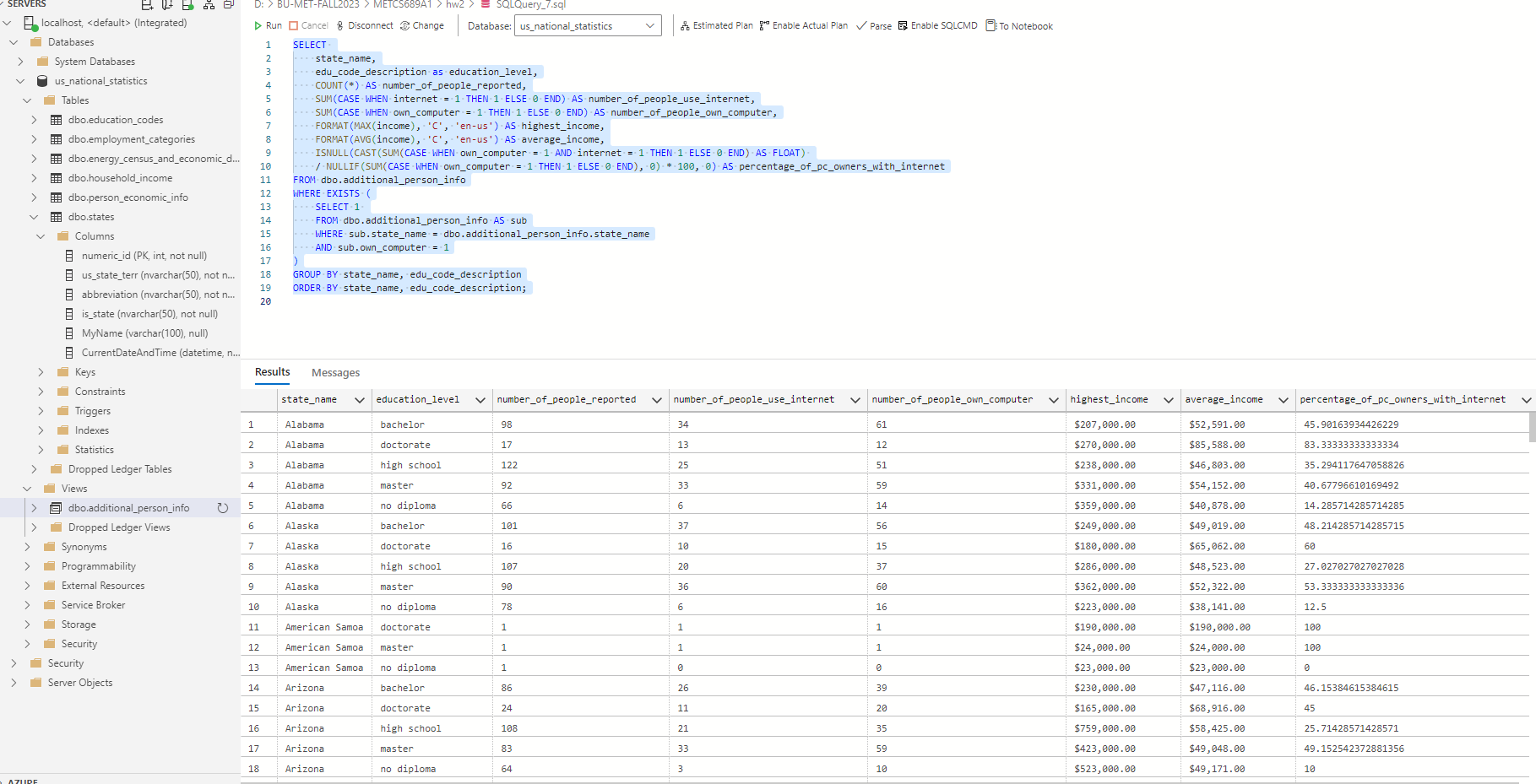


1. For each state AND education level, give the same information (result should have eight columns and multiple rows for each state). Hint: this should be a minor adjustment of the query above by adding a single attribute – education level.

**Text of query:**

|  |
| --- |
| SELECT      state\_name,      edu\_code\_description as education\_level,      COUNT(\*) AS number\_of\_people\_reported,      SUM(CASE WHEN internet = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_use\_internet,      SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END) AS number\_of\_people\_own\_computer,      FORMAT(MAX(income), 'C', 'en-us') AS highest\_income,      FORMAT(AVG(income), 'C', 'en-us') AS average\_income,      ISNULL(CAST(SUM(CASE WHEN own\_computer = 1 AND internet = 1 THEN 1 ELSE 0 END) AS FLOAT)      / NULLIF(SUM(CASE WHEN own\_computer = 1 THEN 1 ELSE 0 END), 0) \* 100, 0) AS percentage\_of\_pc\_owners\_with\_internet  FROM dbo.additional\_person\_info  WHERE EXISTS (      SELECT 1      FROM dbo.additional\_person\_info AS sub      WHERE sub.state\_name = dbo.additional\_person\_info.state\_name      AND sub.own\_computer = 1  )  GROUP BY state\_name, edu\_code\_description  ORDER BY state\_name, edu\_code\_description; |

**Screenshot of result:**

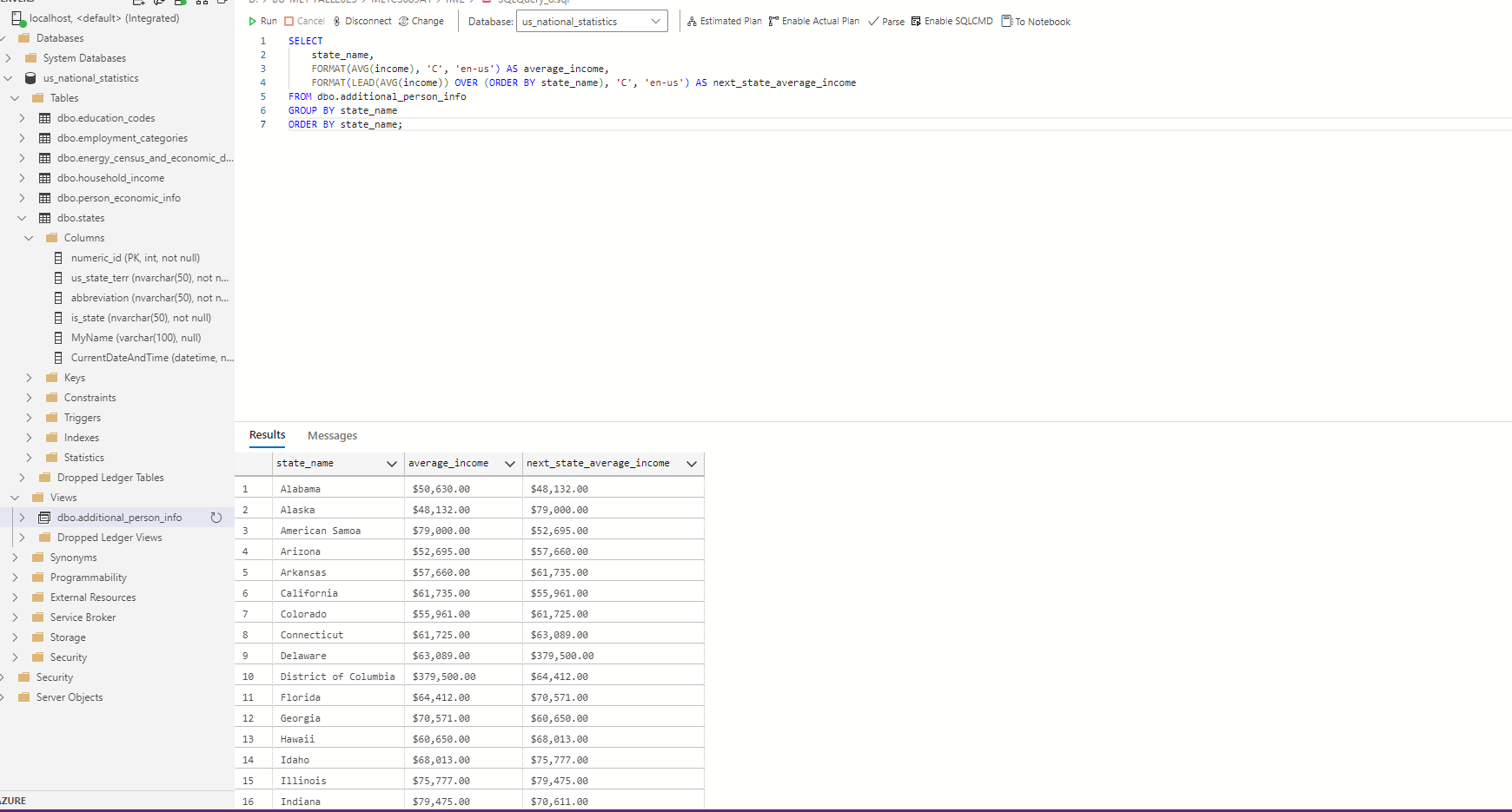
****

1. Extra credit (1 points): Implement a query using Lag/Lead or Pivot. In a single sentence explain what you are trying to accomplish.

**Text of query:**

|  |
| --- |
| SELECT      state\_name,      FORMAT(AVG(income), 'C', 'en-us') AS average\_income,      FORMAT(LEAD(AVG(income)) OVER (ORDER BY state\_name), 'C', 'en-us') AS next\_state\_average\_income  FROM dbo.additional\_person\_info  GROUP BY state\_name  ORDER BY state\_name; |

**Screenshot of result:**

****

With the provided SQL query using the LEAD() function, the goal is to compare the average income of each state with the average income of the subsequent state, when the states are listed in alphabetical order. This allows for a direct, row-by-row comparison between each state and its alphabetically following neighbor, enabling insights into income disparities or similarities between neighboring states in the list.

# Part 2 – Familiarization with Python

1. Make sure that all of the assignment .py, .csv and .ipynb files have been downloaded into the same folder.
2. From the start menu, open Anaconda Navigator (it may take a minute or two)
3. Launch Jupyter Notebook
4. Navigate to the folder containing the “Python Intro.ipynb” file
5. Run each of the first eight cells individually. In the submission file, summarize what these commands did.

**Your summary here – list each cell and what the command did:**

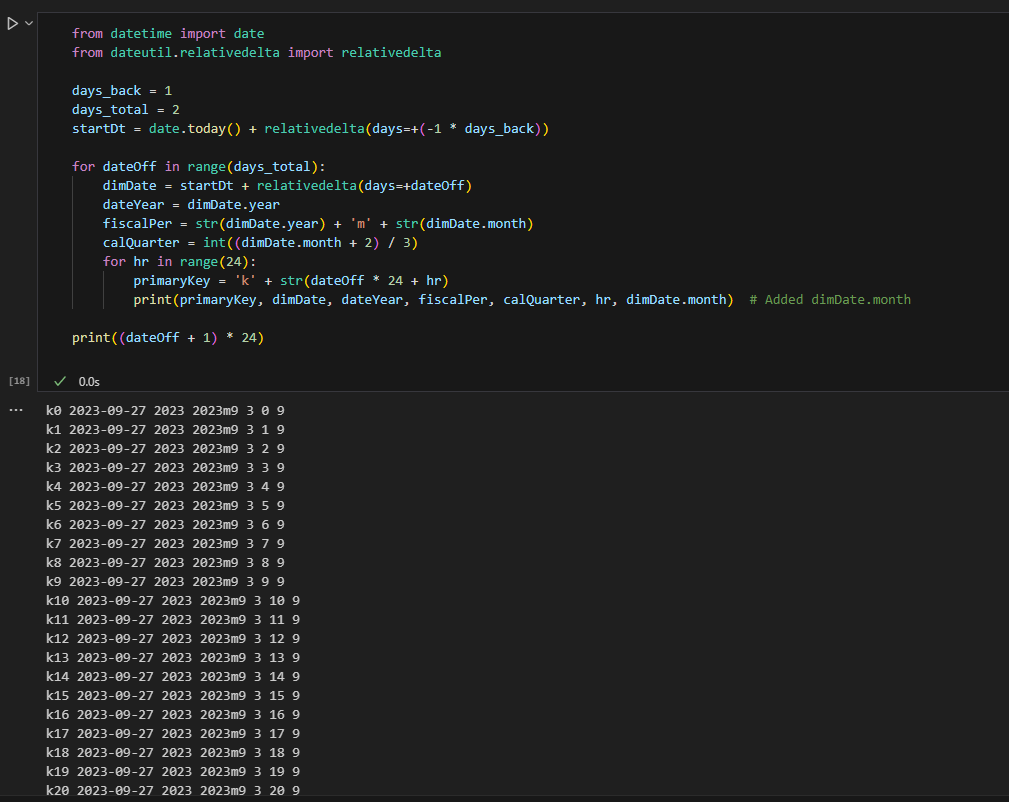
|  |
| --- |
| **Cell 1**  *import random*  This cell imports the random module, which contains functions for generating random numbers.  **Cell 2**  *random.randrange(50,500)*  This cell generates a random integer between 50 (inclusive) and 500 (exclusive).  **Cell 3**  *str(random.randrange(50,500))*  This cell generates another random integer between 50 and 500 and converts it to a string using the str() function.  **Cell 4**  *print ("Hello " + str(random.randrange(50,500)))*  This cell prints a string that starts with "Hello " followed by a random integer between 50 and 500, converted to a string.  **Cell 5**  *int((1 + 2) / 3)*  This cell performs the operation (1 + 2) / 3, which results in 1.0, and then converts it to an integer using the int() function, resulting in 1.  **Cell 6**  *int((2 + 2) / 3)*  This cell performs the operation (2 + 2) / 3, which results in approximately 1.3333, and then converts it to an integer using the int() function, resulting in 1.  **Cell 7**  *int((3 + 2) / 3)*  This cell performs the operation (3 + 2) / 3, which results in approximately 1.6667, and then converts it to an integer using the int() function, resulting in 1.  **Cell 8**  *for i in range(12):*  *print (str(i + 1) + " -> " + str(int((i + 3) / 3)) )*  This cell contains a for loop that iterates over the range of numbers from 0 to 11 (12 exclusive). For each iteration, it prints a string consisting of the current value of i + 1, followed by " -> ", and then the result of the operation int((i + 3) / 3). The operation adds 3 to the current value of i, divides the result by 3, and converts it to an integer.  In summary, these cells demonstrate the use of the random module to generate random numbers, basic arithmetic operations, string conversion, and printing, as well as iteration using a for loop. |

1. Run the ninth cell and add another column for the month. Summarize what these commands do. Paste a screen shot of the modified command and of the result.

**Your summary here:**

|  |
| --- |
| days\_back = 1 and days\_total = 2 set the number of days back from today and the total number of days to process, respectively.  startDt = date.today() + relativedelta(days=+(-1 \* days\_back)) calculates the start date by subtracting days\_back from today's date.  A for loop iterates over the range of days\_total, calculating dimDate, dateYear, fiscalPer, and calQuarter for each day in the range.  dimDate is the date being processed.  dateYear is the year of dimDate.  fiscalPer is a string representing the fiscal period, formatted as year + 'm' + month.  calQuarter is the calendar quarter of dimDate.  A nested for loop iterates over the 24 hours of each day, generating a primaryKey and printing out the values for each hour.  Finally, the code prints out the total number of rows generated, calculated as (dateOff + 1) \* 24. |

**Screenshot of result**:

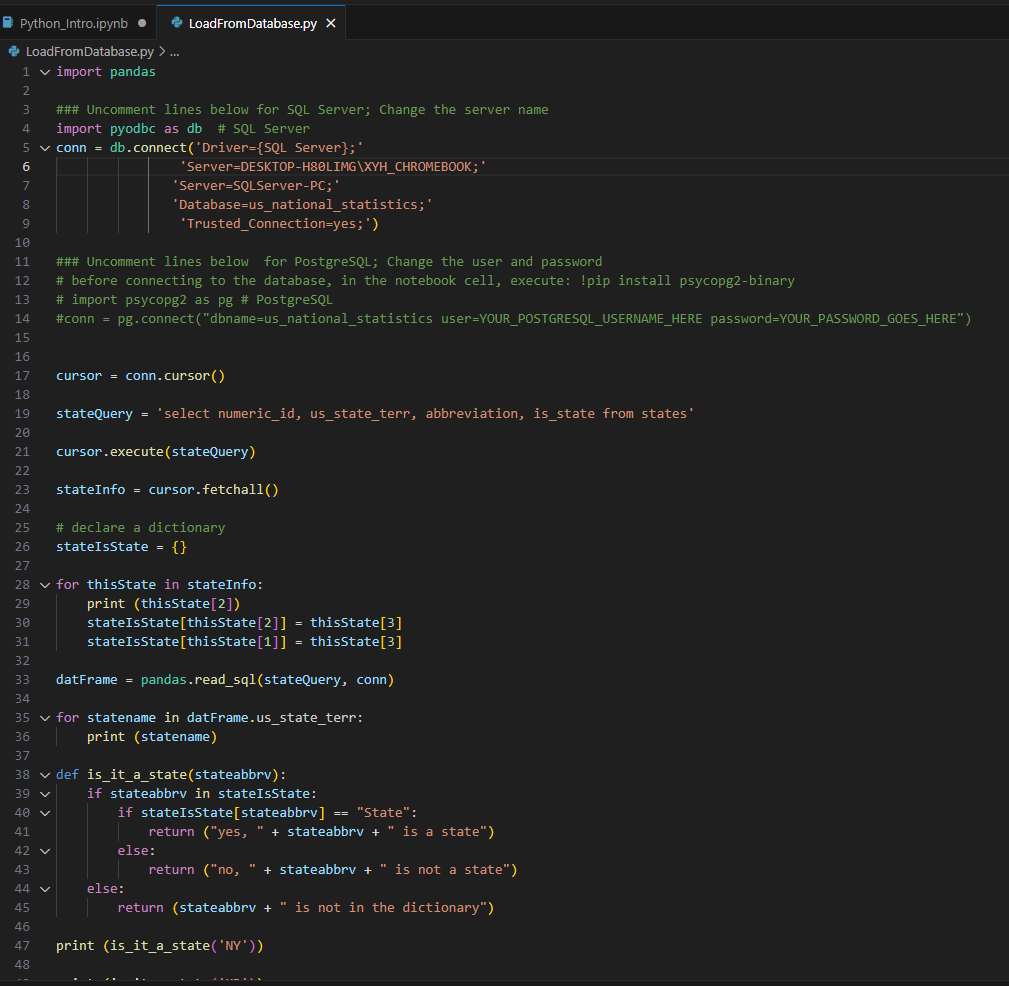


1. Run the next five cells, exploring the titanic.csv file. In the submission file, summarize what these commands did.

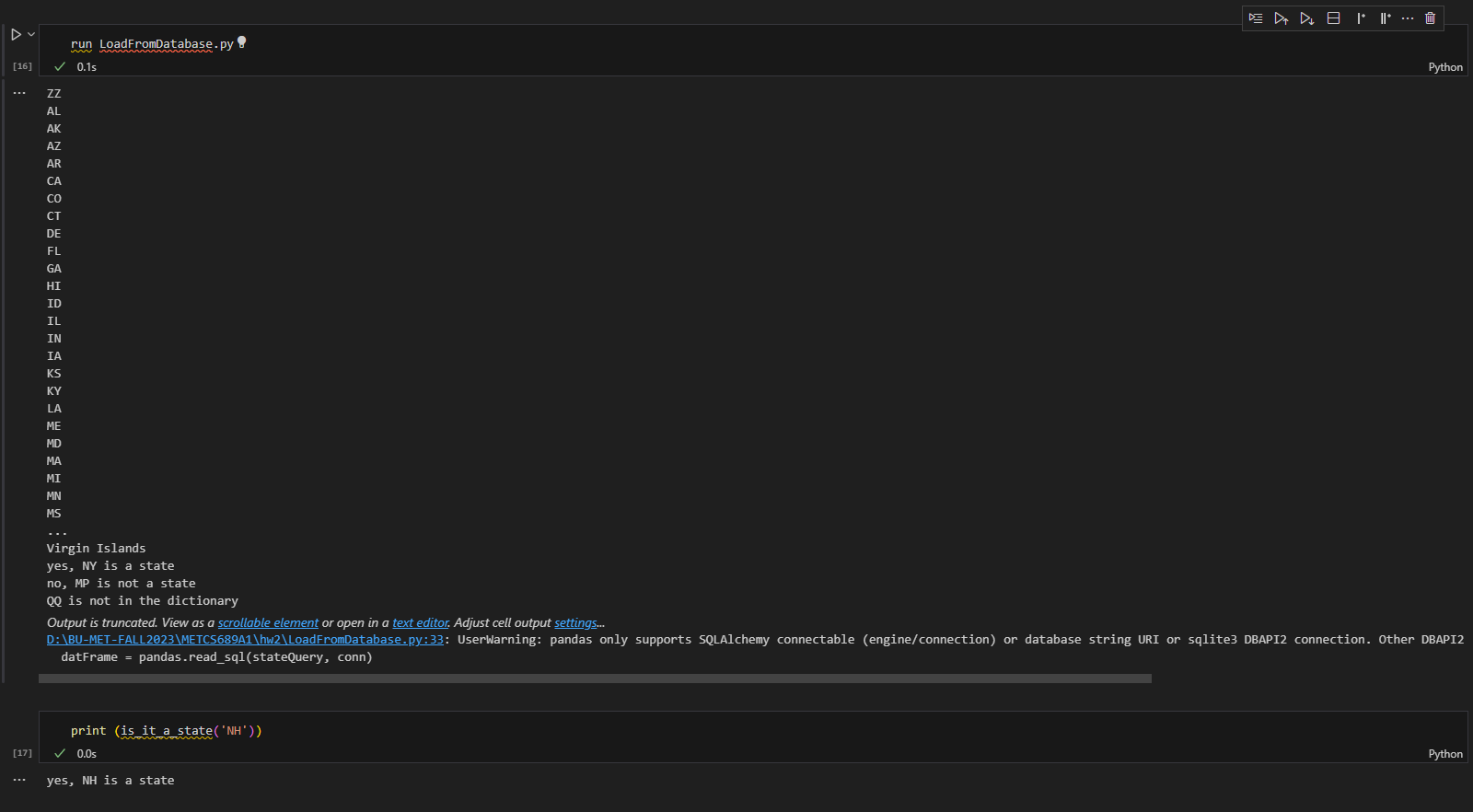
|  |
| --- |
| **1. run LoadTitanic.py**  This command runs a Python script named LoadTitanic.py. This script likely loads a dataset related to the Titanic, into a DataFrame named titanic\_ppl.  **2. list(titanic\_ppl.columns.values)**  This command lists the names of the columns in the titanic\_ppl DataFrame. The columns.values attribute retrieves the column names, and list() converts them to a list.  **3. titanic\_ppl.count()**  This command returns the count of non-NA/null values for each column in the titanic\_ppl DataFrame. It is useful for understanding how many valid (non-missing) data points exist in each column of the dataset.  **4. len(titanic\_ppl)**  This command returns the number of rows in the titanic\_ppl DataFrame. It gives the total count of records (or people) in the dataset.  **5. run dimPeople.py**  This command runs another Python script named dimPeople.py. Without more information, it's unclear what this script does, but it may perform some operations or transformations on a DataFrame, related to people on the Titanic. |

1. Extra credit (1 point): Using a notepad editor, edit the LoadFromDatabase.py file. Change the connection for your type of database server. Run the last two cells. In the submission file, summarize what these commands did. Note – we will explore how to connect to the database to both read and write to a database in a future assignment, so don’t worry if you can’t figure this out just yet.

**Screenshot of updated command**

****

**and running the last two cells**:



**Your summary here:**

|  |
| --- |
| **1. run LoadFromDatabase.py**  This command executes a Python script named LoadFromDatabase.py. The specific actions performed by this script are not clear from the command alone, but based on the name, it likely loads data from a database. It might be connecting to a database, retrieving data, and possibly storing it in a variable or a data structure, such as a DataFrame, for further use or analysis in the notebook.  **2. print(is\_it\_a\_state('NH'))**  This command calls a function is\_it\_a\_state with the argument 'NH' and prints the result. The function name suggests that it might be checking whether the provided argument corresponds to a U.S. state, possibly returning a Boolean value (True or False). In this case, 'NH' is the abbreviation for New Hampshire, a U.S. state, so if the function is correctly implemented, it would likely return True. |

Use the **Ask the Teaching Team Discussion Forum** if you have any questions regarding the how to approach this assignment.

Save your assignment as ***lastnameFirstname\_assign1\_B.docx*** and submit it in the *Assignments* section of the course.

For help uploading files please refer to the *Technical Support* page in the syllabus.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criterion** | **A** | **B** | **C** | **D** | **F** | **Letter Grade** |
| **Correctness and Completeness of Results (70%)** | All steps' results are entirely complete and correct | About ¾ of the steps' results are correct and complete | About half of the steps' results are correct and complete | About ¼ of the steps' results are correct and complete | Virtually none of the step's results are correct and complete |  |
| **Constitution of SQL/Python and Explanations (30%)** | Excellent use and integration of appropriate SQL/Python constructs and supporting explanations | Good use and integration of appropriate SQL/Python constructs and supporting explanations | Mediocre use and integration of appropriate SQL/Python constructs and supporting explanations | Substandard use and integration of appropriate SQL/Python constructs and supporting explanations | Virtually all SQL/Python constructs and supporting explanations are unsuitable or improperly integrated |  |
|  |  |  |  |  | Assignment Grade: |  |