Vincent Nguyen

nvui@bu.edu

MET CS 689 Designing and Implementing a Data Warehouse

Assignment 1B

# 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 states.us\_state\_terr, employment\_categories.category\_description,

education\_codes.education\_level\_achieved, person\_economic\_info.\*

FROM person\_economic\_info

JOIN states ON states.numeric\_id = person\_economic\_info.address\_state

JOIN employment\_categories ON employment\_categories.employment\_category = person\_economic\_info.employment\_category

JOIN education\_codes ON education\_codes.code = person\_economic\_info.education;

**Screenshot of result**:

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

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 us\_state\_terr as state\_name, count (own\_computer) as Num\_own\_computers

FROM additional\_person\_info

WHERE own\_computer = 1

GROUP BY us\_state\_terr;

**Screenshot of result:**

**Table

Description automatically generated**

**Another alternative query for this question can be:**

**Text of query**:

SELECT us\_state\_terr as state\_name, sum(own\_computer) as Num\_own\_computers

FROM additional\_person\_info

GROUP BY us\_state\_terr

HAVING sum(own\_computer) > 0;Table

Description automatically generated

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**:

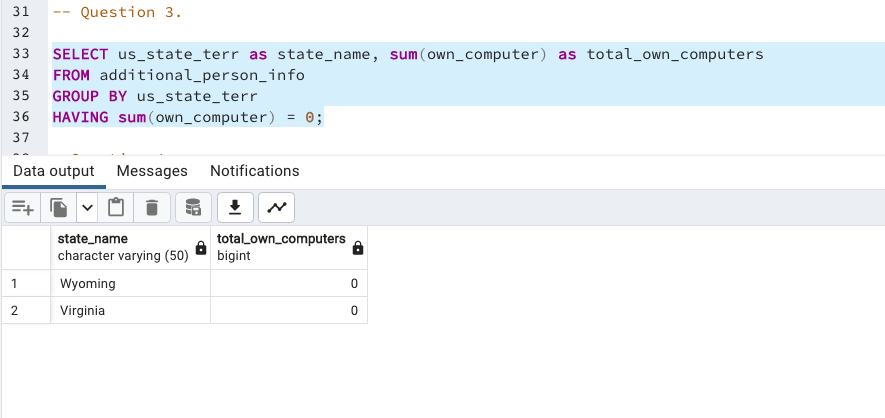
SELECT us\_state\_terr as state\_name, sum(own\_computer) as total\_own\_computers

FROM additional\_person\_info

GROUP BY us\_state\_terr

HAVING sum(own\_computer) = 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 us\_state\_terr as state\_name, education\_level\_achieved as Education\_Level,

count(\*) as Numbers\_respondents, sum(own\_computer) as Total\_own\_computers,

avg(income) as Avg\_income

FROM additional\_person\_info

GROUP BY CUBE(us\_state\_terr,education\_level\_achieved)

ORDER BY us\_state\_terr;

**Screenshot of result**:

Order By function has been added to the query to get better view.

Table

Description automatically generated

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**:

The query below shows each state’s rank based on Average income:

SELECT us\_state\_terr as state\_name, avg(income) as Avg\_income,

Dense\_rank() OVER (ORDER BY avg(income) DESC) as DenseRankState

FROM additional\_person\_info

GROUP BY us\_state\_terr

ORDER BY DenseRankState;

**Screenshot of result**:

Table

Description automatically generated

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 A.\*,

Round(((cast(A.internet\_user as decimal)/cast(nullif(A.num\_own\_computer,0) as decimal))\*100),2) as percent\_internet\_user

FROM

(SELECT us\_state\_terr as state\_name, count(\*) as num\_people\_reported,

count(case when internet <> 0 then 1 ELSE NULL END) as internet\_user,

count(case when own\_computer = 1 then 1 ELSE NULL END) as num\_own\_computer,

cast(max(income) as money) as highest\_income,

cast(avg(income) as money) as average\_income

from additional\_person\_info

group by us\_state\_terr

order by us\_state\_terr) A;

**Screenshot of result**:

Order By function has been added to the query to get better view.

Table

Description automatically generated

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 A.\*,

Round(((cast(A.internet\_user as decimal)/cast(nullif(A.num\_own\_computer,0) as decimal))\*100),2) as percent\_internet\_user

FROM

(SELECT us\_state\_terr as state\_name, education\_level\_achieved as education\_level,

count(\*) as num\_people\_reported,

count(case when internet <> 0 then 1 ELSE NULL END) as internet\_user,

count(case when own\_computer = 1 then 1 ELSE NULL END) as num\_own\_computer,

cast(max(income) as money) as highest\_income,

cast(avg(income) as money) as average\_income

from additional\_person\_info

group by (us\_state\_terr, education\_level\_achieved)

order by us\_state\_terr) A;

**Screenshot of result:**

**Table

Description automatically generated**

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 address\_state, us\_state\_terr as state\_name, cast(avg(income) as money) as Avg\_income,

cast(lag(avg(income)) over (order by address\_state) as money) as avg\_previous\_state\_income,

case

when avg(income) > lag(avg(income)) over(order by address\_state) then 'Higher than previous state'

when avg(income) < lag(avg(income)) over(order by address\_state) then 'Lower than previous state'

when avg(income) = lag(avg(income)) over(order by address\_state) then 'Same previous state'

end avg\_income\_range,

cast(lead(avg(income)) over (order by address\_state) as money) as avg\_next\_state\_income

FROM additional\_person\_info

GROUP BY us\_state\_terr, address\_state

**Screenshot of result:**

**Table

Description automatically generated**

* The query above is used to get the information of the average income in each state.
* LAG function is used to display the average income in the previous state and its format in monetary format (avg\_previoust\_state\_income). and this LAG function is also used to compare the average income in the current state is higher, lower or the same as the average income in the previous sate (avg\_income\_range column).
* LEAD function is used to display the average income in the next state (avg\_next\_state\_income) in monetary format.

# 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:**

Graphical user interface, text, application, email

Description automatically generated

* import random use to import random library
* random.randrange(50,500) use to generate a random integer number within the range of 50 and 500
* str(random.randrange(50,500)) generate a random integer number within the range of 50 and 500 and then convert that number to string.
* print ("Hello " + str(random.randrange(50,500))) use to print a message “Hello” and number in string type (a random integer number within the range of 50 and 500 ( this number is converted to string with str() function)).
* int((1 + 2) / 3), int((2 + 2) / 3) and int((3 + 2) / 3): convert floating point result after performing plus and division operation to an integer.
* for i in range(12):

print (str(i + 1) + " -> " + str(int((i + 3) / 3)) )

this is a loop which uses to print out message (print (str(i + 1) + " -> " + str(int((i + 3) / 3)) )) 12 times

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:**

These commands print out the series of primary key for each hour in a range of date (2 days) and the other information about each hour such as dimeDate (include year-month-date), dateYear (year), fiscalPer (its yearnmonth format, example 2023m3 which is year 2020 and m and march (3)), calQuater (Quarter which is 1), month and hour (has 24 hours).

from datetime import date

from dateutil.relativedelta import relativedelta

* These codes are using to import date and relativedelta modules.

days\_back = 1

days\_total = 2

startDt = date.today() + relativedelta(days=+(-1 \* days\_back))

* Set days\_back and days\_total number
* startDt is to calculate the start date, day back is 1 so the start date will be one day back

for dateOff in range(days\_total):

dimDate = startDt + relativedelta(days=+dateOff)

month = dimDate.month

dateYear = dimDate.year

fiscalPer = str (dimDate.year) + 'm' + str(dimDate.month)

calQuarter = int((dimDate.month + 2) / 3)

for hr in range(24):

primaryKey = 'k' + str(dateOff \* 24 + hr)

print (primaryKey, dimDate, dateYear, fiscalPer, calQuarter, month, hr)

* because days\_total = 2 so the outer loop will have 2 iterations
* each iteration of outer loop will calculate the dimdDate, month, dateYear, fiscalPer, calQuarter
* within each iteration of outer loop, the inner loop will have 24 iterations, it calculates primaryKey and then print out primaryKey, dataYear, fiscalPer, calQuarter, month, hr 24 times.

print ((dateOff + 1) \* 24) after completing the loop, this command will print out 48 (dateOff = 1 because loop start from index 0)

***Screenshot of result (not modified)****:*

Graphical user interface, text

Description automatically generated

***Screenshot of modified result:***

Text

Description automatically generated

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

**Your summary here:**

Graphical user interface, text, application, email

Description automatically generated

The command above print out the index and the column “name” in the titanic.csv file

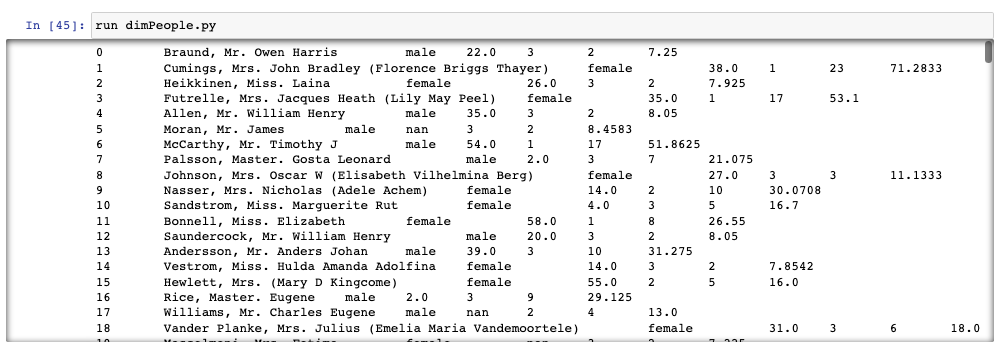
Graphical user interface, application

Description automatically generated

***list(titanic\_ppl.columns.values)*** returns the list which contains the column names.

***titanic\_ppl.count()*** use to count the number of non-null values in each column of the titanic\_ppl and the return the columns name and the number non-null values counted.

***len(titanic\_ppl)*** return number of rows in titanic\_ppl (891 rows)



***run dimPeople.py*** use to print out passenger’s information such as index, name, sex, age, passenger class, fare divided by 3 and fare as float.

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**:

Graphical user interface, text, application, Word

Description automatically generated

**Your summary here:**

* ***run LoadFromDatabase.py*** return the state name in abbreviation and full name of state
* ***print (is\_it\_sate(‘NH’))*** is to check if NH is a state and print out message (There is a function (is\_it\_a\_state(argument)) to check if NH is a state or not in LoadFromDatabase.py file.

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: |  |