BLM5127- Big Data Analytics
Project Assignment

- **Your Name:** __Mehmet Onur ERBOY___
- **Instructions for project**
  - **Downloading data:**
    First download the folder in the zip file attached to this assignment: **timeusage.zip.**
    For this assignment, you also need to download the data (164 MB):
    http://alaska.epfl.ch/~dockermoocs/bigdata/atussum.csv
    and place it in the folder **src/main/resources/timeusage/** in your project directory.

  - **The problem**
    The dataset is provided by Kaggle and is documented here:
    https://www.kaggle.com/bls/american-time-use-survey

    The file uses the comma-separated values format: the first line is a header defining the field names of each column, and every following line contains an information record, which is itself made of several columns. It contains information about how people spend their time (e.g., sleeping, eating, working, etc.).

    Our goal is to identify three groups of activities:
    - primary needs (sleeping and eating),
    - work,
    - other (leisure).

    And then to observe how do people allocate their time between these three kinds of activities, and if we can see differences between men and women, employed and unemployed people, and young (less than 22 years old), active (between 22 and 55 years old) and elder people.

    To answer the questions below, first read the dataset with Spark, transform it into an intermediate dataset which will be easier to work with, and finally compute information that will answer the questions.

  - **Read-in Data:**
    The simplest way to create a DataFrame consists in reading a file and letting Spark-sql infer the underlying schema. However this approach does not work well with CSV files, because the inferred column types are always String.

    In our case, the first column contains a String value identifying the respondent but all the other columns contain numeric values. Since this schema will not be correctly inferred by Spark-sql, we will define it programmatically. However, the number of columns is huge. So, instead of manually enumerating all the columns we can rely on the fact that, in the CSV file, the first line contains the name of all the columns of the dataset.

Our first task consists in turning this first line into a Spark-sql StructType. This is the purpose of the dfSchema method. This method returns a StructType describing the schema of the CSV file, where the first column has type StringType and all the others have type DoubleType. None of these columns are nullable.

def dfSchema(columnNames: List[String]): StructType

The second step is to be able to effectively read the CSV file is to turn each line into a Spark-sql Row containing columns that match the schema returned by dfSchema. That's the job of the row method.

def row(line: List[String]): Row

- **Project info:**

  The initial dataset contains lots of information that are not needed to answer the questions, and even the columns that contain useful information are too detailed. For instance, we are not interested in the exact age of each respondent, but just whether she was "young", "active" or "elder".

  Also, the time spent on each activity is very detailed (there are more than 50 reported activities). Again, this level of detail is not needed; only interested in three activities: primary needs, work and other. With this initial dataset it would a bit hard to express the queries that would give the answers.

  The second part of this assignment consists in transforming the initial dataset into a format that will be easier to work with.

  A first step in this direction is to identify which columns are related to the same activity. Based on the description of the activity corresponding to each column (given in https://www.kaggle.com/bls/american-time-use-survey), we deduce the following rules:
  "primary needs" activities (sleeping, eating, etc.) are reported in columns starting with "t01", "t03", "t11", "t1801" and "t1803" ;
  working activities are reported in columns starting with "t05" and "t1805" ;
  other activities (leisure) are reported in columns starting with "t02", "t04", "t06", "t07", "t08", "t09", "t10", "t12", "t13", "t14", "t15", "t16" and "t18" (only those which are not part of the previous groups).

  Then your work consists in implementing the classifiedColumns method, which classifies the given list of column names into three Column groups (primary needs, work or other). This method should return a triplet containing the "primary needs" columns list, the "work" columns list and the "other" columns list.

  **def classifiedColumns(columnNames: List[String]):
  (List[Column],List[Column], List[Column])**

  The second step is to implement the **timeUsageSummary** method, which projects the detailed dataset into a summarized dataset. This summary will contain only 6 columns: the working status of the respondent, sex, age, the amount of daily hours spent on primary needs activities, the amount of daily hours spent on working and the amount of daily hours spent on other activities.

```
def timeUsageSummary(
        primaryNeedsColumns: List[Column],
        workColumns: List[Column],
        otherColumns: List[Column],
        df: DataFrame
       ): DataFrame
```

Each "activity column" will contain the sum of the columns related to the same activity of the initial dataset. Note that time amounts are given in minutes in the initial dataset, whereas in our resulting dataset we want them to be in hours.

The columns describing the work status, the sex and the age, will contain simplified information compared to the initial dataset.

Last, people that are not employable will be filtered out of the resulting dataset.
The comment on top of the **timeUsageSummary** method will give you more specific information about what is expected in each column.

- **Aggregate Results**
  Compare the average time spent on each activity, for all the combinations of working status, sex and age.
  Implement the timeUsageGrouped method which computes the average number of hours spent on each activity, grouped by working status (employed or unemployed), sex and age (young, active or elder), and also ordered by working status, sex and age. The values will be rounded to the nearest tenth.

  ```
  def timeUsageGrouped(summed: DataFrame): DataFrame
  ```

  Now you can run the project and see what the final DataFrame contains.

- **Cassandra**
  Finally, save the dataframe **timeUsageSummary** that you generated into Cassandra Database. Also, try getting the same aggregate results in the previous step now by using Cassandra.

- **Submission:** The source code of your algorithms, a simple description on how to run your programs; and a screen shot on running each algorithm.
  You may use this document as a template and add your report text on the next page. Your report should be submitted as PDF or Word document.

- **Deadline of the report:** **December 30, 2021**

Questions

1. Answer the following questions based on the dataset:
   a) How much time do we spend on primary needs compared to other activities?

   b) Do women and men spend the same amount of time in working?

   c) Does the time spent on primary needs change when people get older? In other words, how much time elder people allocate to leisure compared to active people?

   d) How much time do employed people spend on leisure compared to unemployed people?

The requirements of the execution of code :

- There is a conda environment YAML file to specify the necessary Python library to execute the code successfully. With these, command you have to create a conda environment for this project :
  conda env create -f environment.yml
- The input file is described as a CSV file named **atussum.csv** located on the same directory with the **main.py**. For successful code running, you have to arrange the code and input file with these format.
- The Cassandra has to be downloaded locally to execute data loading part. With these queries, you can create desired environment if you executed them on cql sh :

```
create keyspace bigdataproject with replication = {'class' : 'SimpleStrategy',
'replication_factor' : 1};

use bigdataproject;

CREATE TABLE summarized_time_data(md
    working_status varchar,
    age varchar,
    sex varchar,
    primary_hours float,
    working_hours float,
    other_hours float,
    primary key (working_status, age, sex)
);
```

- After these operations, you can activate conda environment with **conda activate bigDataAnalyticProject** and then you can execute the code with **python main.py** code.
- **Warning :** The PyCharm is used as development environment. Trying the code and execute it on PyCharm is suggested to compare the execution result and homework result.

PySpark python library is used scope of this homework because of the easy to code and high ability of Python coding of developer.

The PySpark has a CSV data load method. So, read data operation could be done in one-line code.

The pattern of columns related the activities is given **Project Info** part. With these information, these column values are summarized and their values are placed with **primary_hours, working_hours, others_hours** dataframe column.

The columns except activity columns are **sex, working_status** and **age**. These columns are obtain with these logics :

- **sex** : Obtained from **tesex** column. These column contains 1 which denotes male and 2 which denotes female. These values are kept as **'male'** and **'female'**.

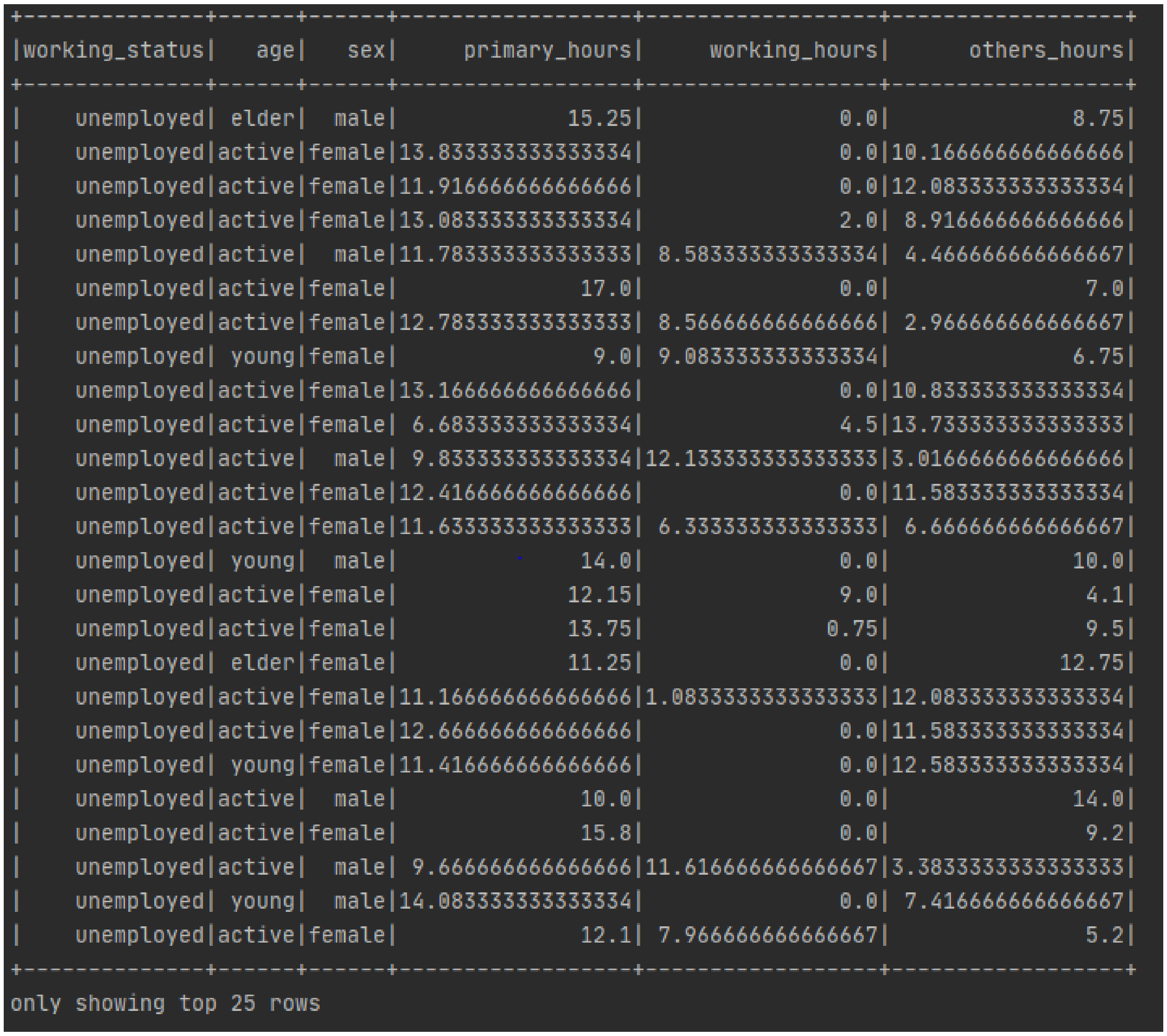
- age : Obtained from **teage** column. This column actually contains integer value as age. On the other hand, we have to follow this grouping rule : These values are grouped as **'young'** which denotes the people under 22, **'active'** which denotes the people between 22 and 55, **'elder'** which denotes people above 55.

- working_status : Obtained from **telfs.** This column is described as labor force information according to schema information of dataset. According to schema information, this field could get these values :

1 : Employed – at work
2 : Emloyed - absent
3 : Unemployed – on layoff
4 : Unemployed - looking
5 : Unemployed – not in a labor force

Scope of these values, I grouped the data as employed if its **telfs** value equals to **1 or 2**, otherwise I grouped the data as umemployed which they got **3 or 4 or 5** value as **telfs** value.

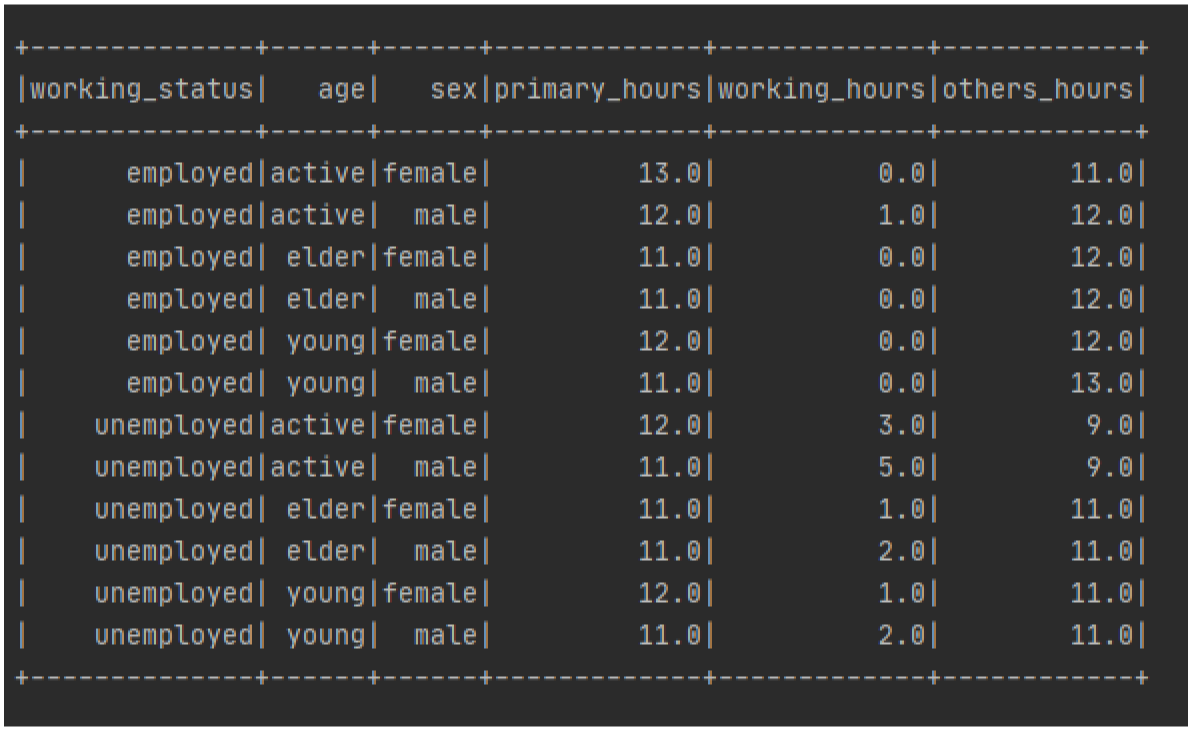
For this vision, the first 25 element of dataframe that obtained as a result of **timeUsageSummary()** function :



```
+--------------+------+------+-----------------+-----------------+-----------------+
|working_status|   age|   sex|    primary_hours|    working_hours|     others_hours|
+--------------+------+------+-----------------+-----------------+-----------------+
|    unemployed| elder|  male|            15.25|              0.0|             8.75|
|    unemployed|active|female|13.833333333333334|              0.0|10.166666666666666|
|    unemployed|active|female|11.916666666666666|              0.0|12.083333333333334|
|    unemployed|active|female|13.083333333333334|              2.0| 8.916666666666666|
|    unemployed|active|  male|11.783333333333333| 8.583333333333334| 4.466666666666667|
|    unemployed|active|female|             17.0|              0.0|              7.0|
|    unemployed|active|female|12.783333333333333| 8.566666666666666| 2.966666666666667|
|    unemployed| young|female|              9.0| 9.083333333333334|             6.75|
|    unemployed|active|female|13.166666666666666|              0.0|10.833333333333334|
|    unemployed|active|female| 6.683333333333334|              4.5|13.733333333333333|
|    unemployed|active|  male| 9.833333333333334|12.133333333333333|3.0166666666666666|
|    unemployed|active|female|12.416666666666666|              0.0|11.583333333333334|
|    unemployed|active|female|11.633333333333333| 6.333333333333333| 6.666666666666667|
|    unemployed| young|  male|             14.0|              0.0|             10.0|
|    unemployed|active|female|            12.15|              9.0|              4.1|
|    unemployed|active|female|            13.75|             0.75|              9.5|
|    unemployed| elder|female|            11.25|              0.0|            12.75|
|    unemployed|active|female|11.166666666666666|1.0833333333333333|12.083333333333334|
|    unemployed|active|female|12.666666666666666|              0.0|11.583333333333334|
|    unemployed| young|female|11.416666666666666|              0.0|12.583333333333334|
|    unemployed|active|  male|             10.0|              0.0|             14.0|
|    unemployed|active|female|             15.8|              0.0|              9.2|
|    unemployed|active|  male| 9.666666666666666|11.616666666666667|3.3833333333333333|
|    unemployed| young|  male|14.083333333333334|              0.0| 7.416666666666667|
|    unemployed|active|female|             12.1| 7.966666666666667|              5.2|
+--------------+------+------+-----------------+-----------------+-----------------+
only showing top 25 rows
```

Figure 1 : First 25 element of timeUsageSummary() function result dataframe

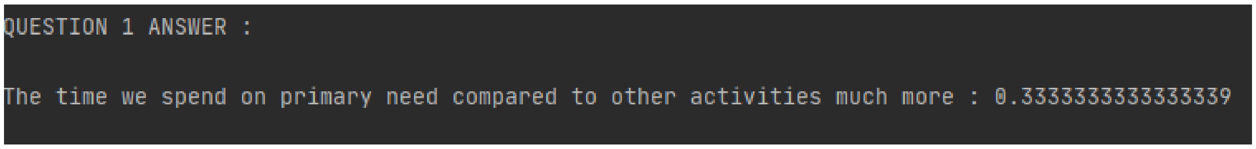
The grouped dataframe result after the calculation done scope of the **timeUsageGrouped()** function is represented with this screenshot :



```
+--------------+------+------+-------------+-------------+------------+
|working_status|   age|   sex|primary_hours|working_hours|others_hours|
+--------------+------+------+-------------+-------------+------------+
|      employed|active|female|         13.0|          0.0|        11.0|
|      employed|active|  male|         12.0|          1.0|        12.0|
|      employed| elder|female|         11.0|          0.0|        12.0|
|      employed| elder|  male|         11.0|          0.0|        12.0|
|      employed| young|female|         12.0|          0.0|        12.0|
|      employed| young|  male|         11.0|          0.0|        13.0|
|    unemployed|active|female|         12.0|          3.0|         9.0|
|    unemployed|active|  male|         11.0|          5.0|         9.0|
|    unemployed| elder|female|         11.0|          1.0|        11.0|
|    unemployed| elder|  male|         11.0|          2.0|        11.0|
|    unemployed| young|female|         12.0|          1.0|        11.0|
|    unemployed| young|  male|         11.0|          2.0|        11.0|
+--------------+------+------+-------------+-------------+------------+
```

Figure 2 : The dataframe that obtained after **timeUsageGrouped()** function call.

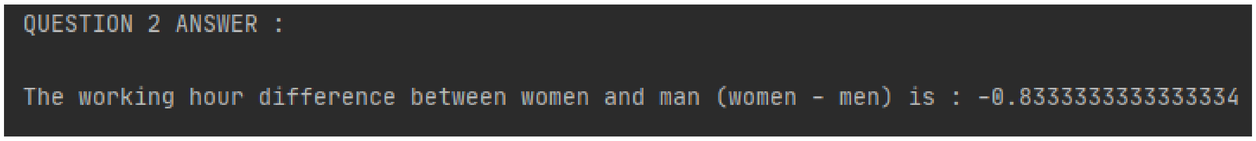
There are 4 questions are given scope of this homework to answer according to Figure 2 result. For this, 4 different function is coded to answer related the questions. Here, the answers are shown in the figures sequentially (The code of answers could be found on **main.py**):



```
QUESTION 1 ANSWER :

The time we spend on primary need compared to other activities much more : 0.3333333333333339
```

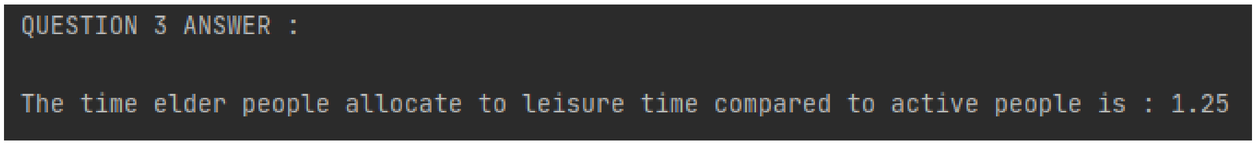
Figure 3 : Question 1 Answer



```
 QUESTION 2 ANSWER :

 The working hour difference between women and man (women - men) is : -0.8333333333333334
```

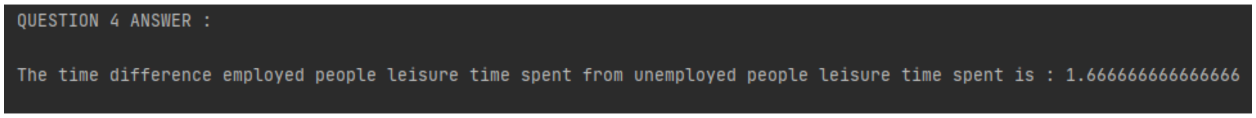
Figure 4 : Question 2 Answer



```
QUESTION 3 ANSWER :

The time elder people allocate to leisure time compared to active people is : 1.25
```

Figure 5 : Question 3 Answer



```
QUESTION 4 ANSWER :

The time difference employed people leisure time spent from unemployed people leisure time spent is : 1.6666666666666666
```

Figure 6 : Question 4 Answer

Cassandra side operation : The pyspark dataframe context could not loaded fully to Cassandra because of I could not found a batch way to load timeUsageSummary() result dataframe. I reached only sequential inserting one-by-one from pyspark dataframe to Cassandra. So, I only loaded some sample of the data. But I could not generate an aggregation method to obtain timeUsageGrouped() function result dataset.