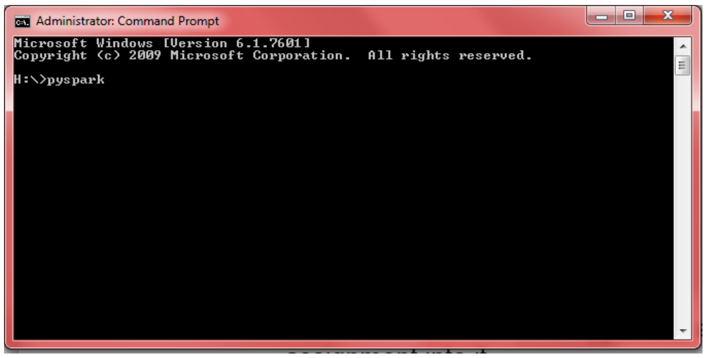
CSC 3355
Database Management
Assignment : Spark SQL

1. Create a folder named as SparkDB and download all files from canvas associated with this assignment into it.
2. Open a command prompt (search for cmd) and Type "pyspark"



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

H:\>pyspark
```

3. The command should open a jupyter notebook in your browser. Jupyter allows you to write python programs in a browser and let you visually see the results of your instructions immediately. The following is a good introduction to jupyter notebook in case you want to learn more.
   https://www.youtube.com/watch?v=HW29067qVWk

4. Right click to the file *employee.py* and click to "open with notepad". Do the same for *employee.json*.
5. In Jupyter, browse to the folder where you downloaded things from Canvas.
6. Open a python notebook in Jupyter (New → Python (default)) and copy the contents of employee.py in that. Use Cell → Run Cells to run this code and compare your output with the instructor's.

Creating DataFrames

The entry point into all functionality in Spark is the SparkSession class . With a SparkSession, applications can create DataFrames from an existing RDD, or from Spark data sources. Every Spark SQLprogram should start with the following

```
from pyspark.sql import SparkSession
```

As an example, the following creates a DataFrame based on the content of a JSON file:

```
df = spark.read.json("employee.json")
df.show()
+---+-------+------+
|age|   name|salary|
+---+-------+------+
| 29|Michael|  3000|
| 25|   Andy|  4500|
| 34| Justin|  3500|
| 36|  Berta|  4000|
| 29|  Nancy|  3700|
| 25| Monika|  3000|
+---+-------+------+
```

Operations on DataFrames

```
# Print the schema in a tree format

df.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
 |-- salary: long (nullable = true)
```

```
# Select only the "name" column

df.select("name").show()
```

```
+--------+
|    name|
+--------+
|Michael|
|    Andy|
|  Justin|
|   Berta|
|   Nancy|
|  Monika|
+--------+
```

```
# Select everybody, but double the salary

df.select(df['name'], df['salary'] * 2).show()
```

```
+--------+------------+
|    name|(salary * 2)|
+--------+------------+
|Michael|        6000|
|    Andy|        9000|
|  Justin|        7000|
|   Berta|        8000|
|   Nancy|        7400|
|  Monika|        6000|
+--------+------------+
```

In Python it's possible to access a DataFrame's columns either by attribute (df.age) or by indexing (df['age']). While the former is convenient for interactive data exploration, users are highly encouraged to use the latter form, which is future proof and won't break with column names that are also attributes on the DataFrame class.

```
# Select employee earning >= 4000

df.filter(df['salary'] >= 4000).show()
```

```
+---+-----+-------+
|age| name|salary|
+---+-----+-------+
| 25| Andy|   4500|
| 36|Berta|   4000|
```


```
# Count people by age

df.groupBy("age").count().show()
```

```
+---+-----+
|age|count|
+---+-----+
| 29|    2|
| 34|    1|
| 25|    2|
| 36|    1|
+---+-----+
```

For a complete list of the types of operations that can be performed on a DataFrame refer to the API Documentation.

Running SQL Queries Programmatically

The SQL function on a SparkSession enables applications to run SQL queries programmatically and returns the result as a DataFrame.

```
# Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("employee")
sqlDF = spark.sql("SELECT * FROM employee")
sqlDF.show()

+---+-------+------+
|age|   name|salary|
+---+-------+------+
| 29|Michael|  3000|
| 25|   Andy|  4500|
| 34| Justin|  3500|
| 36|  Berta|  4000|
| 29|  Nancy|  3700|
| 25| Monika|  3000|
+---+-------+------+
```

PART A: Your Turn: (5X8 = 40 points)

1. Following the above code, insert a cell in jupyter (in SQL) that selects only the “name” column from the employee table and shows the output.
2. Show all records with name and salary doubled
3. Show name and salary of the employees whose income is >= 4000
4. Show the employee whose age is greater than or equal to 30 and whose income is greater than or equal to 4000.
5. Show average salary of all employee grouped by age.

Working with real data (CSV file)

See the attached FB-small.csv file that contains historical prices of Facebook stock and was obtained from Yahoo Finance website (https://finance.yahoo.com/). Each record contains the stock values of a single date and includes the following attributes*: date, open price, high price, low price, close price, volume*, and *adjClose* (close price adjusted for dividends and splits). You may find it useful to add an additional attribute called *year,* extracted from *date,* in order to handle queries efficiently.

```
#To open this file
df=spark.read.csv('FB-small.csv', mode="DROPMALFORMED",inferSchema=True,
header = True)
df.show()

# Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("FBStock")

# Operations on the table
sqlDF = spark.sql("SELECT * FROM FBStock")
sqlDF.first()

sqlDF = spark.sql("SELECT Date, Open FROM FBStock WHERE (Open >= 175.00)")
sqlDF.show()
```

PART B: Your Turn: (60 points)

Download the FB-large.csv file. Investigate the contents of the file. Write a Spark SQL program that shows/answers the following queries.

1. Show the first 10 records in the table. [5 Points]
2. Show the number of records in the table. [5 Points]
3. Show all records that have gained value during daily transaction in the year 2018 (close >= open). [10 Points]
4. Which day did Facebook stock gain maximum value? [10 Points]
5. Show the first 10 highest stock values. [10 Points]
6. Show the average sale volume for last 5 years. [10 Points]
7. Add two more questions and write queries to answer them. [10 Points]

Extra Credit [20 points]

Download the OrderDB.txt file. You need to explore the ways to open a text file and create a DataFrame from it. Each line in this file contains {Order-ID, Customer_id, Order_date, total} where total is the amount of money spent be the customer on that order.Write a Spark SQL program that shows/answers the following queries.

- Find the total amount spent by each customer considering all the orders.
- Find out the date that has the most number of orders.
- Find out the average amount that customers spend for each day.

You will find following documentation handy to solve the above questions
https://spark.apache.org/docs/2.2.1/sql-programming-guide.html

Submission (Due on 4/26)

Upload your python files (the content of jupyter notebook) for each part separately (when everything is running good) in canvas. Include your and your partner's name in the submission.