Final Project

DS 730

# Overview

In the final project you will be working with all of the technologies you have learned in this course. Along with solving each of the following problems, you must provide a small text document explaining why you chose to solve that problem in the way you solved it. For instance, if you solved the first problem using Pig, you should explain why you chose Pig over Hive or Java. Many of these do not have an obvious decision and your choice is subjective. Because of this, you must justify your choice. The only problem that must be solved in a specific way is the last problem.

As you work through the project, keep these points in mind:

* If there is a tie for any of the questions (for example, Task 1, Problem 3 may have more than one most common wind direction), you should print out all of them.
* If you are using Hadoop, you must assume the data is stored in the HDFS folder of: **/home/hduser/final/** and your output will be written to a folder called **/home/hduser/final/output**.
* If you are using the local filesystem to read in the files, you should not change the name of the files nor should you store them in a separate folder. For example, if you are writing a Java program to solve one of the questions, the input will be in the same folder as the Java file.

# Project Tasks

## Task 1: Analyze Weather Information

For the first task, you will be reading in from a file called **Weather.csv**. This file is available in the online course. You are encouraged to look at the file to see what the data looks like.

Once you have a good grasp on what is contained in the file, you are to answer the following problems. Note that all problems are *not* necessarily best solved using only 1 tool. It’s possible that Problem 1 is best solved with Pig, whereas Problem 2 is best solved with Java. You do not need to justify every single choice you make. Justifying them as a group is sufficient. For example, you could say, “I chose to solve Problems 1, 3, 8, and 10 with Pig because…”

**Important:**

The only constraint is that *you must solve each question with one of the tools learned in this class*:

* Java threads
* Python MapReduce
* Hadoop MapReduce
* Pig
* Hive.

If you wish to solve them in some other fashion, you must ask me first.

Some of these problems may have a subjective meaning. If I ask which day was the hottest, that could be interpreted in different ways. Do I want to know which day had the highest temperature or do I want to know which day was the hottest with respect to the average? For example, assume one day was 94 the entire day and another day was in the 70’s for most of the day and then jumped to 95 before falling back to the 70’s. Which day would be considered “hotter?” I will try and explain exactly what I am looking for. If a problem feels ambiguous, ask me before attempting it.

You should be aware of bogus values in the input. For instance, some of the values are set to -9999 if no value was recorded. These values should not be included in your calculations. However, the rest of the row should not be discarded.

**Example:** On January 26th, 2008, the temperature was recorded as -9999 but there is a valid wind speed. The wind speed should *not* be discarded.

Make sure to look at the data before beginning so you can figure out what values are invalid. Ask me if you are not sure if a value is valid or not.

Part of submitted work

You are to solve each of these problems and provide the answers in a file called **output.txt**. For each problem, make sure that you create a separate script or program to run for each problem.

| **Problem #** | **Description** |
| --- | --- |
| **1** | Which is more common:   * days where the temperature was below -9.9 (i.e. -10 or lower)   or   * days where the temperature was above 94.9 (i.e. 95 or higher)? |
| **2** | What is the average high and low temperature for each month? For this problem, we are taking the widely accepted interpretation of “average high for each month” to mean this: for a particular month, take the high temperature for each day and average them. If a month has 4 days and the highs for each respective day were 50, 55, 56 and 53, the average high for the month would be 53.5. Similarly for average low. |
| **3** | What was the most common wind direction? |
| **4** | Which month has the most precipitation? |
| **5** | What 7-day period was the hottest? By hottest I mean, the *average* temperature from 12:00am on day 1 to 11:59pm on day 7. |
| **6** | Which hour is the coldest time of the day? For each day, determine the coldest time for that day. The coldest time for any given day is defined as the hour that has the coldest average. For example, a day may have had two readings during the 4am hour, one at 4:15am and one at 4:45am. The temperatures may have been 10.5 and 15.3. The average for 4am is 12.9. The 5am hour for that day may have had two readings at 5:14am and 5:35am and those readings were 11.3 and 11.5. The average for 5am is 11.4. 5am is thus considered colder. Once you have the coldest hour for each day, return the hour that has the most occurrences of the coldest average. |
| **7** | What day had the largest temperature difference and what was that difference? That difference need not be from 12am to 11:59pm. The difference may have occurred in the middle of the day (e.g. 3:45pm to 5:30pm). |
| **8** | What time period that is 24 hours or less saw the largest temperature difference? For example, the largest temperature difference may have been from 3pm on a Tuesday to 3pm on a Wednesday. The largest temperature difference could have been from 11:07am on a Tuesday to 4:03am on a Wednesday. Or the largest difference could have been from 3:06pm on a Wednesday to 7:56pm on that same Wednesday. |
| **9** | What hour of the day is the least windy? By least windy, I mean *the average wind speed for that hour is the smallest*. |
| **10** | What years had more snowfall (precipitation-wise) in March than February? |
| **11** | How many days started off nice and then ended poorly? For this question, I want to know, how many days were *“Clear” for at least 1 recording before noon* and then had *“Heavy Rain” for at least 1 recording after noon*. |
| **12** | How many perfect days were there? By perfect day, I mean a day where:   * the temperature was *between 60 and 79* the entire day   and   * the condition was *clear or partly cloudy* the entire day   and   * the dew point was *under 60* the entire day   and   * and the wind was *10mph or less* the entire day. |
| **13** | What was the lowest and highest barometric pressure recorded (the sea level column) and on what days did those occur? |
| **14** | How long was the longest cooldown? In other words, *the temperature kept going down or stayed exactly the same*. You should give your answer in *hours and minutes*. |
| **15** | What is the *least* common humidity that occurred at least once? |
| **16** | What percentage of the time are the winds calm? For example, let’s say there are 3 days total. Day 1 has 10 valid readings and 2 of them were calm. Day 2 has 15 valid readings and 3 of them were calm. Day 3 has 20 valid readings and 4 of them were calm. As a whole, 15.55% (7/45) of the time it was calm. I am looking for the percentage only over the entire dataset. |
| **17** | What was the longest stretch of subzero temperatures? You should report your answer in *hours and minutes*. |
| **18** | What is the most common **(wind direction, condition)** combination when the wind gust is *over 30mph*? |
| **19** | As a runner, I want to know when the best time to run is. For each month, provide the hour that is the best time to run. The best time to run will be defined as the time where *the temperature is as close to 50 as possible*. If:   * there is a tie, *a tiebreaker will be the least windy hour* on average. * If there is *still* a tie, *both* hours are reported. |
| **20** | How many perfect driving days were there? A perfect driving day is defined as a day where:   * the visibility was *at least 9 miles*   and   * the wind speed was *under 15mph* for the entire day. |

## 

## Task 2: Determine Postal Routes

Due to budget cutbacks, the postal services at UW-Oshkosh can only afford 1 mail deliverer. Even worse, that deliverer is a student who works part-time. Postal services wants to minimize the amount of time that student has to work in order to save money. Because of this, they are interested in the fastest way to visit all buildings and return back to the Campus Services Building. There are obvious routes that are terrible (e.g. going from one side of campus to the other and then back) but the optimal route is not obvious. Your goal is *to read in a file that gives the time in seconds to get from a building to every other building and determine the best possible route such that you start at* ***BuildingOne****, visit all other buildings and end at* ***BuildingOne***.

The input file you will read in is called **input2.txt** and will be formatted in the following manner:

**BuildingOne : t(BuildingOne) t(BuildingTwo) t(BuildingThree) t(BuildingFour) t(BuildingFive)**

**BuildingTwo : t(BuildingOne) t(BuildingTwo) t(BuildingThree) t(BuildingFour) t(BuildingFive)**

**BuildingThree : t(BuildingOne) t(BuildingTwo) t(BuildingThree) t(BuildingFour) t(BuildingFive)**

**BuildingFour : t(BuildingOne) t(BuildingTwo) t(BuildingThree) t(BuildingFour) t(BuildingFive)**

**BuildingFive : t(BuildingOne) t(BuildingTwo) t(BuildingThree) t(BuildingFour) t(BuildingFive)**

Take the first line for example. **t(BuildingTwo)** will be an integer value denoting the number of seconds it takes to get from **BuildingOne** to **BuildingTwo**. On the first line, **t(BuildingOne)** will be 0 for obvious reasons. The input will always be formatted in this manner.

Note:

* If another building is constructed, it will be added to the end and the file will be updated accordingly. For example, if BuildingSix were constructed, the time to BuildingSix will be added at the end of every list and BuildingSix will be added to the end of the file.
* The time from BuildingOne to BuildingThree may *not* be the same as the time from BuildingThree to BuildingOne. There may be one-way streets; it may be uphill, etc.

Your goal is this: print out the total time taken to start with **BuildingOne**, visit all buildings and then return to **BuildingOne**. You must also output the order in which you visited the buildings. You should save this output in a file called **output2.txt**.

Part of submitted work

## Task 3: Report Time Differences

The goal of your last problem is to create a rather large file and run a couple of queries on it on your local Hadoop setup and also run it in the cloud. You will report the differences in time taken to run it locally vs in the cloud. You will be generating 1 file that contains randomly generated information.

**Important:**

Do *not* make any changes to these files or your final answers will not match mine.

The **Person.java** file will generate the following csv file:

**studentId, firstName, lastName, age, addressNum, addressDir, addressStreet, streetType, city, state**

In order to generate this information, you need to run 1 command:

**java RunMe**

It took me about 7 minutes to run this on my machine. If you want, you can download these files to your local Ubuntu filesystem by using this command:

**wget http://www.uwosh.edu/faculty\_staff/krohne/Final.zip**

You can unzip the folder by using: **unzip Final.zip**

You should generate this file on your local Hadoop setup and solve the following problem. I am giving you part of the answer because *the goal of this problem is not to solve the problem. The goal is to measure the difference in time between your local machine and AWS*.

**Problem**: What was the most common **age/addressStreet** combination?

**Answer**: There is at least 1 combination that occurs **52** times.

You are encouraged to solve this problem locally before attempting it on AWS. You do not want to waste your credits only to find out your solution was incorrect. Once you have an answer that looks correct, then you should attempt it on AWS.

**Part 1: Create .csv File**

In order to get your data on S3 without having to upload it from your local machine, use the following directions:

1. Sign in to your AWS console by going to <http://aws.amazon.com>
2. Click the **EC2** icon.
3. Click the **Launch Instance** button.
4. Select the **Amazon Linux AMI…** option.
5. Make sure the **General Purpose t2.micro** option is selected. It should say “Free tier eligible”.
6. Click **Next: Configure Instance Details**.
7. All of the defaults are good here. Click **Next: Add Storage**.
8. Change the GB size from 8GB to 20GB.
9. Click **Next: Tag Instance**.
10. The next page can be kept at the defaults. Click **Next: Configure Security Group**.
11. Change the **Source** from Anywhere to **My IP**. This assumes you will be connecting to the instance from your local machine.

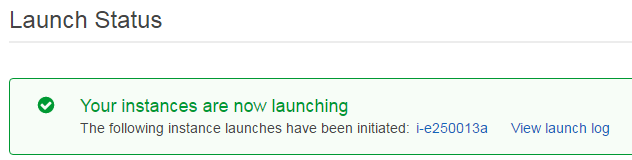
* You are welcome to leave it at Anywhere. It likely doesn’t matter.

1. Click **Review and Launch**.
2. Click the **Launch** button.
3. Click **Create a new key pair**.
4. Enter a Key pair name of whatever you want.
5. Once you enter in a name, click **Download Key Pair**.

* I called mine finalProject.

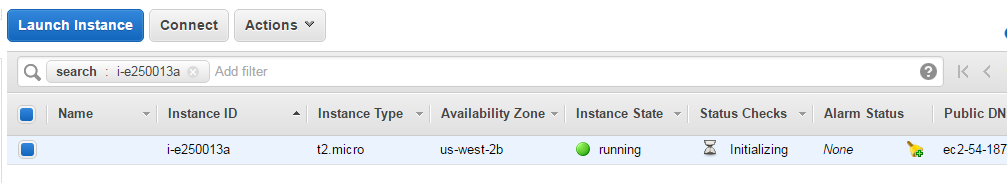
1. Save that file somewhere to your hard drive and then click **Launch Instances**.

* Once you click that button, you will see something like this:



1. Click the **i-e250013a** link. *Yours will almost certainly have a different name*.

* Once you click that link, you will see something similar to this:



1. You’ll notice it says **running**. If it is still initializing or waiting, wait for the **Instance State** to be in **running** mode before continuing.
2. Scroll to the right and find your Public DNS. Write this address down.

* Mine was called ec2-54-187-1-42.us-west-2.compute.amazonaws.com.

1. Once you are in running mode, open up your Ubuntu system.
2. The file I downloaded in step 16 was called **finalProject.pem**. Transfer this pem file to your Ubuntu filesystem.

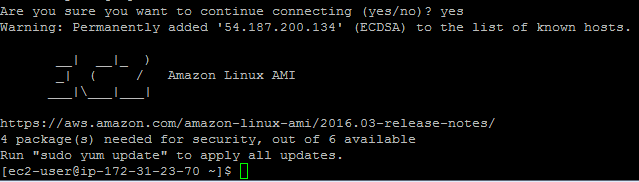
* I’ll assume this file is called **finalProject.pem**.

1. Go to your Ubuntu filesystem.
2. Go to the folder storing your **finalProject.pem** file and enter the following command:   
   **chmod 400 finalProject.pem**
3. Now we want to connect to that EC2 instance. Type the following command:

**ssh -i finalProject.pem ec2-user@ec2-54-187-1-42.us-west-2.compute.amazonaws.com**

The **finalProject.pem** is the file I just edited and the **ec2-54-187…** is the Public DNS from step 20.

1. If it asks about authenticity, you can type **yes**.
2. Once you are connected, you should see something like this:



1. Once you are connected, you should download the files you need to generate your data. You can download that by typing:

**wget http://www.uwosh.edu/faculty\_staff/krohne/Final.zip**

* It should download very quickly.

1. Once it is downloaded, type:   
   **unzip Final.zip**
2. We need to run a Java program but Java is not installed by default. To install it, type:

**sudo yum install java-devel**

1. Type **y** when it asks to install it.
2. Once Java is installed and the files are downloaded, you should compile your Java file by typing:

**javac \*.java**

1. Once your Java file is compiled, you can run it:

**java RunMe**

1. Your **people.csv** file should be **8748337113** bytes.

**Important:**

If your file is *not* exactly this much, it’s *unlikely it was created correctly*. Rerun the java program. If you continue to have problems generating it, contact me right away.

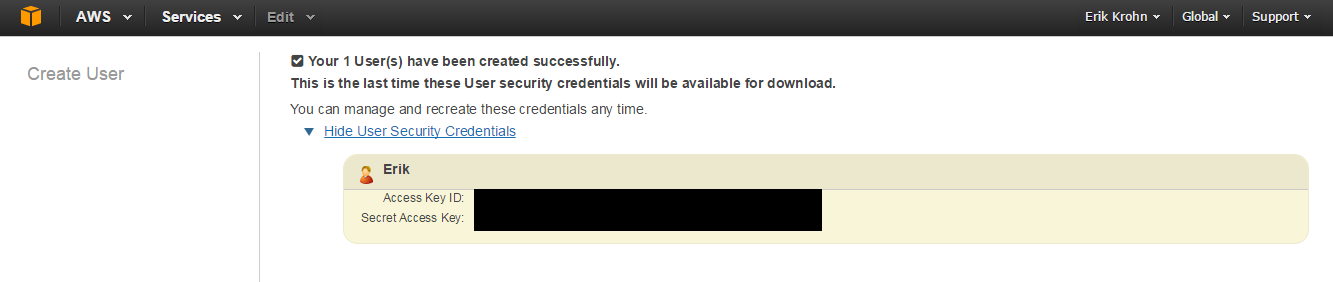
**Part 2: Create an IAM User**

This process is a repeat of steps in the *Hadoop and Cloud Computing* activity. *If you have already created an IAM user who can access the S3 buckets, you can skip to* ***Part 3: Upload File***.

1. We need to create a User who can access our S3 bucket. With AWS open, go up to **Services** and click **IAM**.
2. Click the **Users** option to the left.
3. Click the **Create New Users** button. You only need to create 1 user.
4. Pick a user name.
5. Leave the **Generate an access key for each user** checked.
6. Click the **Create** button.

* It should say something like “Your 1 User(s) have been created successfully.”

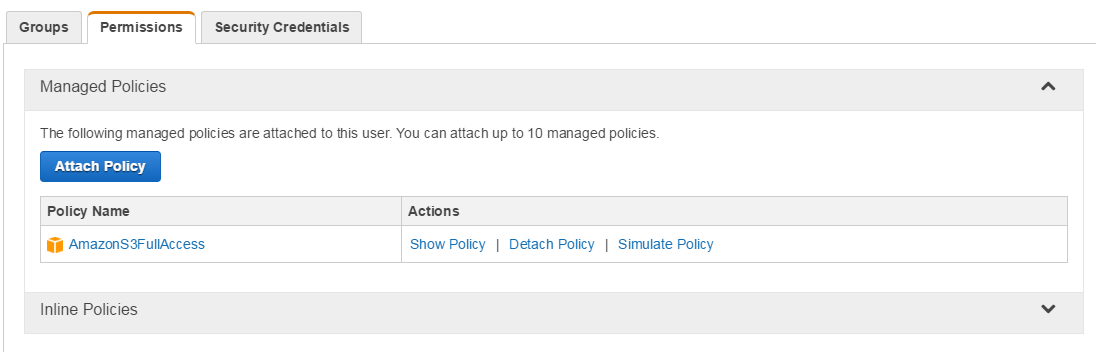
1. Click **Show User Security Credentials**.



1. Save your **Access Key ID** and **Secret Access Key**.

**Important:** *Do not leave this page without saving those values somewhere* as the Secret Access Key can *never be retrieved again*.

1. After you have saved the information, click **Close**.
2. Click the User Name you just created.
3. Underneath the **Permissions** tab, click **Attach Policy**.
4. Click **AmazonS3FullAccess** and then click **Attach Policy**. You should see something like this:



**Part 3: Upload File**

Now we want to add this user to our EC2 instance so we can upload our file easily.

1. Type **aws configure** and wait for the prompt.
2. For **Access Key ID**, type the access key ID you just saved and then press **ENTER**.
3. Do similarly for the **Secret Access Key**.
4. For the **Default region name**, type **us-east-1** assuming you chose the default region of US East (N. Virginia).

* If you chose something else, choose your region appropriately.

1. For the **Default output format**, just press **ENTER**.
2. Make a folder called Data:

**mkdir Data**

1. Move your people.csv file into that folder:

**mv people.csv Data**

1. Go into that folder:

**cd Data**

1. Once you are in the Data folder, you are ready to upload your S3 bucket. Make sure you have an S3 bucket already created and then type:  
   **aws s3 sync . s3://name-of-your-bucket-here/**
2. Your data is now in your S3 bucket and you are ready to run your MapReduce/Pig/Hive solution on it. When you are finished with your EC2 instance, go back to AWS and terminate the instance.

**Part 4: Record Times**

1. Now that your data is on AWS, run it and record the time it takes to solve.

**Important:** I created 17 m3.xlarge instances (1 master and 16 core nodes). With 17 instances, I ran a rather simple Pig script and it cost roughly $6. I ran a similar program in the past with 10 instances and noticed a considerable speedup and that running cost around $4. *Choose your instance size wisely. This is not something you want to run more than a couple of times. Be sure your solution is correct before running it on AWS*.

1. To get the total runtime, look at the bottom of the syslog log file. This will tell you how long the script ran.

Part of submitted work

1. When you are finished, store these in a file called **output3.txt**:

* your answer
* your script

1. Also, create a document explaining the difference in runtime you noticed from your local Hadoop setup and AWS.

# Submitting Your Work

Submit *a .zip file* containing these items to the **Final Project Dropbox**:

* **output.txt** (Task 1): the answers to all problems and a separate script or program to run for each problem.
* **output2.txt** (Task 2): the total time taken to start with **BuildingOne**, visit all buildings and then return to **BuildingOne**. You must also output the order in which you visited the buildings.
* **output3.txt** (Task 3): how long it took you to run your script locally vs through AWS. Also include your script.