Assignment one:

DATA CLEANING AND WRANGLING

TEAM 1

MANASI DALVI

VISHAL SATAM

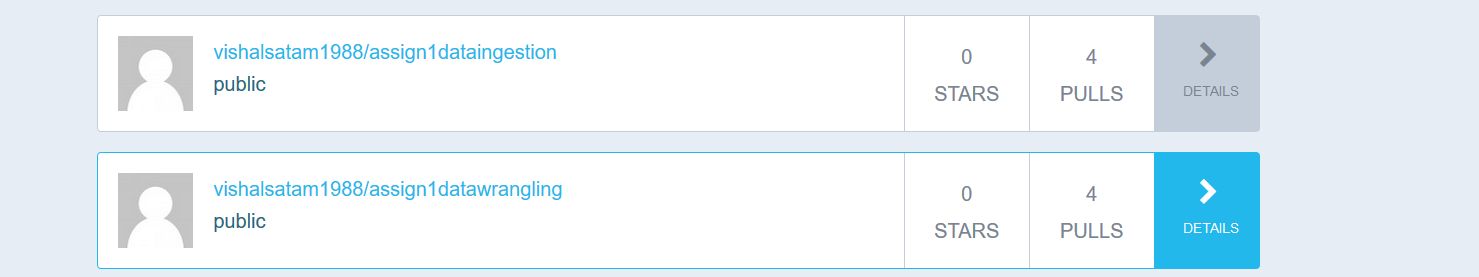
DOCKER & Execution Instructions

The required files have been packaged into 2 Docker images and are present on Docker Hub with the image names as vishalsatam1988/assign1dataingestion and vishalsatam1988/assign1datawrangling. The manual execution steps are given below.

**Note :** The docker container runs in the **UTC timezone**.

If you encounter any Memory Error issues. Please increase the RAM of your docker virtual machine to minimum 4 GB.

**Docker-hub image**

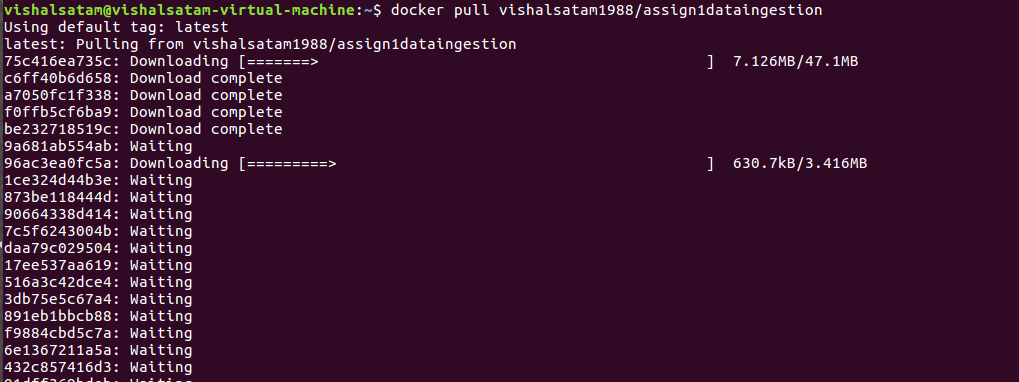


**Data Ingestion**

**Step 1** **: Pull the image**

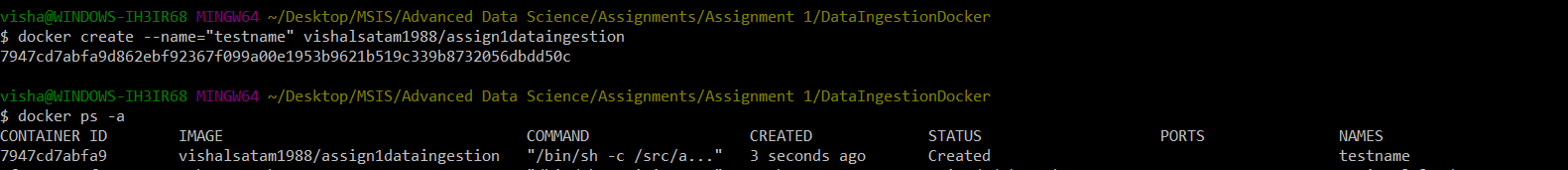
The docker image is present on the docker hub and is available to pull using the following command

docker pull vishalsatam1988/assign1dataingestion



**Step 2 : Create the container using**

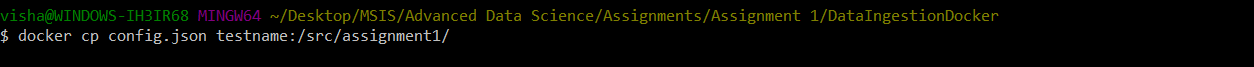
docker create --name="testname" vishalsatam1988/assign1dataingestion



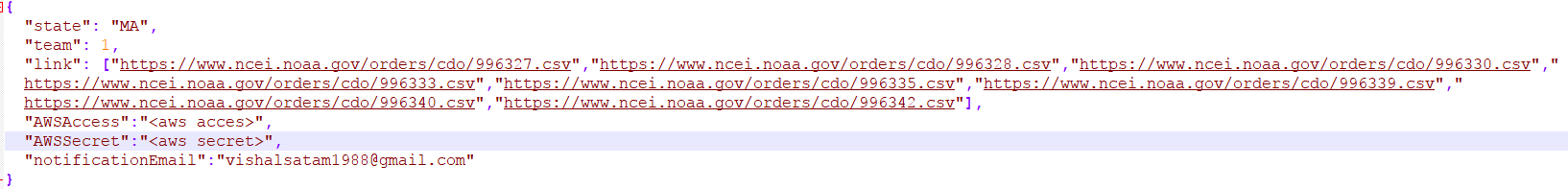
**Step 3 : copy your config.json and the configIntial.json file to update the links and your AWS credentials. Please do not change the name of these files.**

docker cp <local file path> <containername>:/src/assignment1/

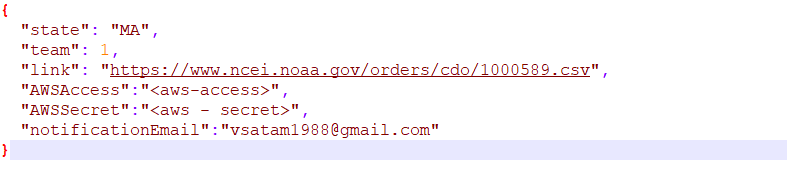
docker cp config.json testname:/src/assignment1/



Sample configInitial.json



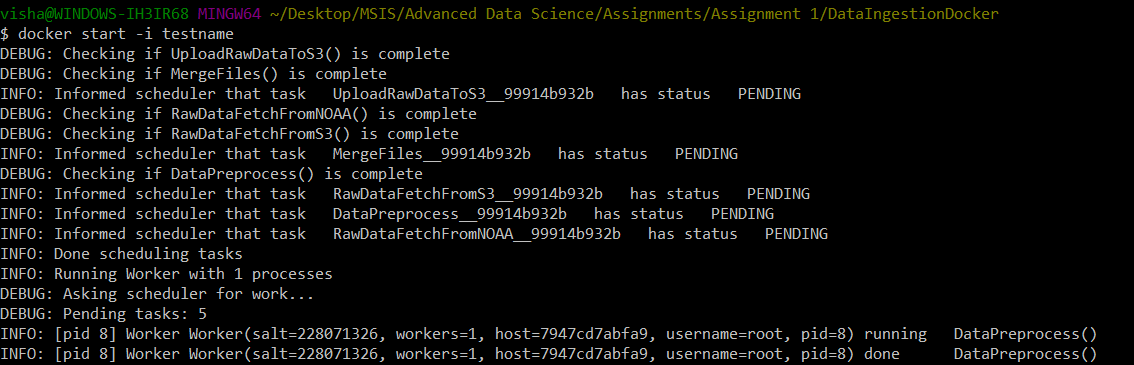
Sample config.json



**Step 4 : Start the container**

docker start <containername>

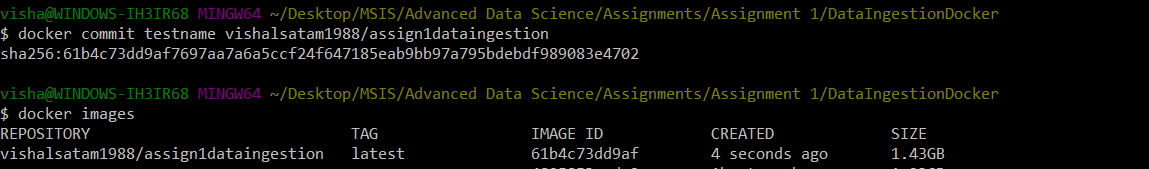
docker start -i testname



**Step 5 : Commit the container to persist the changes**

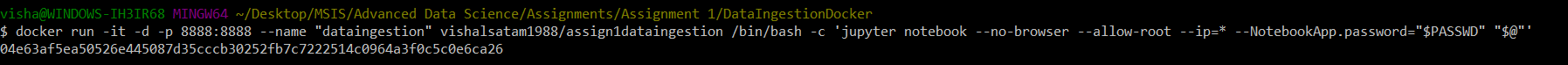
docker commit <containername> <new image name>

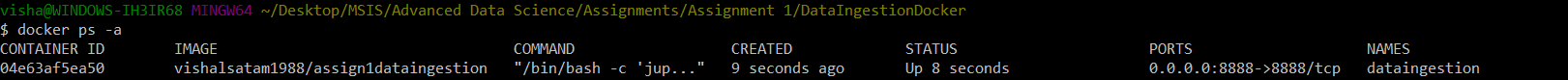
docker commit testname vishalsatam1988/assign1dataingestion



**Step 6 : Run the jupyter notebook on the committed image in detached mode**

docker run -it -d --name “dataingestion” -p 8888:8888 vishalsatam1988/assign1dataingestion /bin/bash -c 'jupyter notebook --no-browser --allow-root --ip=\* --NotebookApp.password="$PASSWD" "$@"'

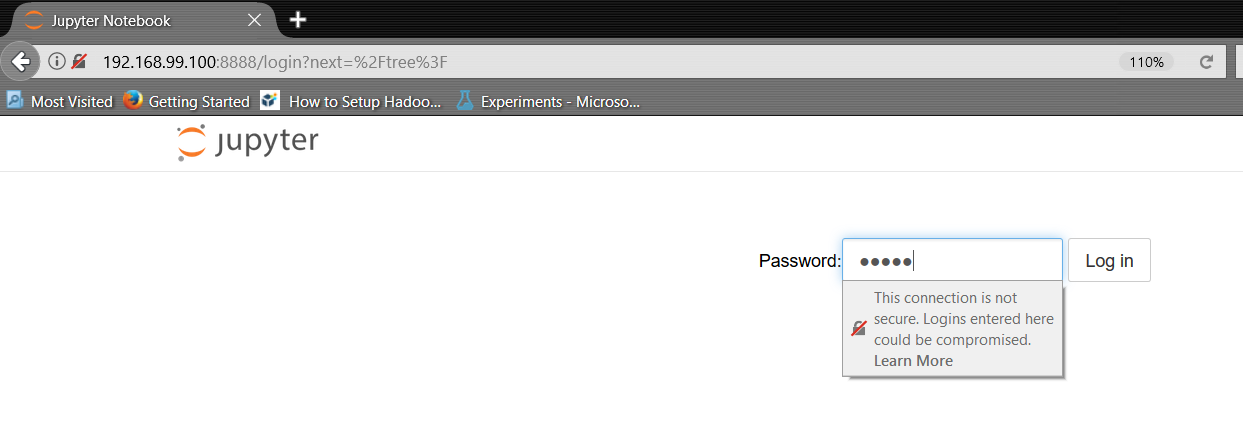




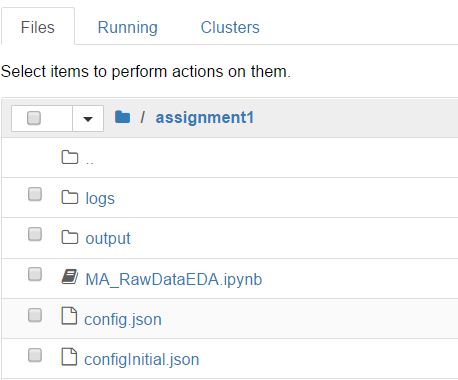
**Step 7 : Connect to this running container via browser by entering**

http://<docker machine ip address>:8888

**Password for the jupyter notebook is keras**

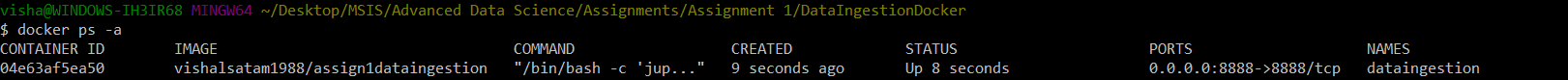


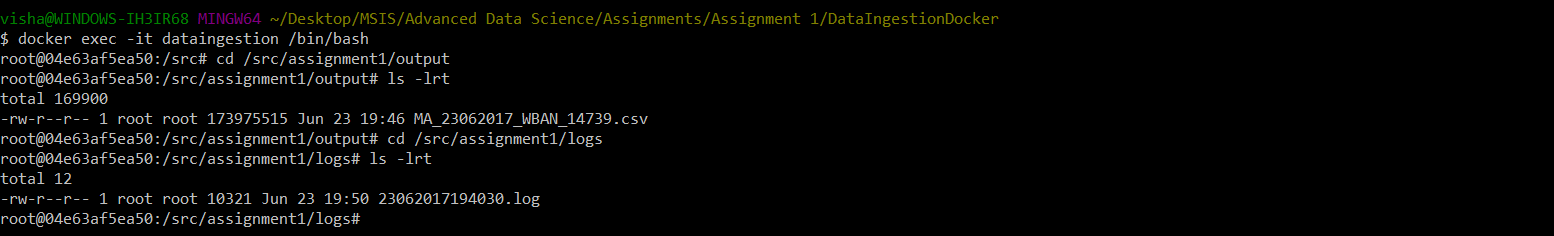
Navigate to the assignment1 folder and open the MA\_RawDataEDA ipython notebook to see the analysis



**Step 8 : Execute the bin/bash command to enter the running container to check output and logs**

docker exec -it dataingestion /bin/bash



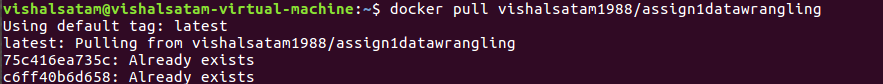


**Data Wrangling**

**Step 1 : Pull the image**

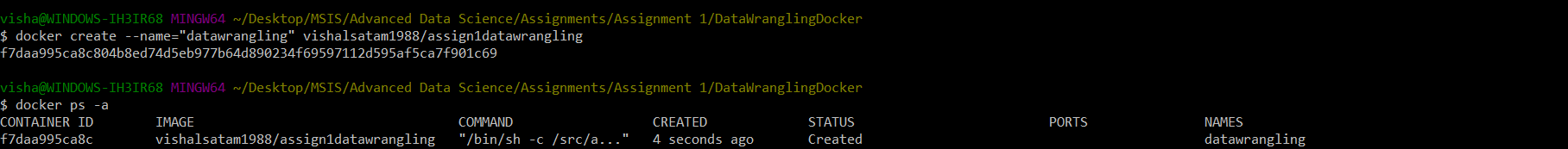
The docker image is present on the docker hub and is available to pull using the following command

docker pull vishalsatam1988/assign1datawrangling



**Step 2 : Create the container using the below command**

docker create --name="datawrangling" vishalsatam1988/assign1datawrangling



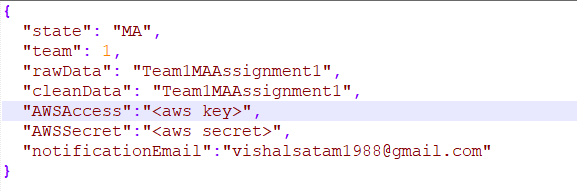
**Step 3 : copy your configWrangle.json file to update the link. Please do not change the name of the config file.**

docker cp <local file path> <containername>:/src/assignment1/

docker cp configWrangle.json datawrangling:/src/assignment1/



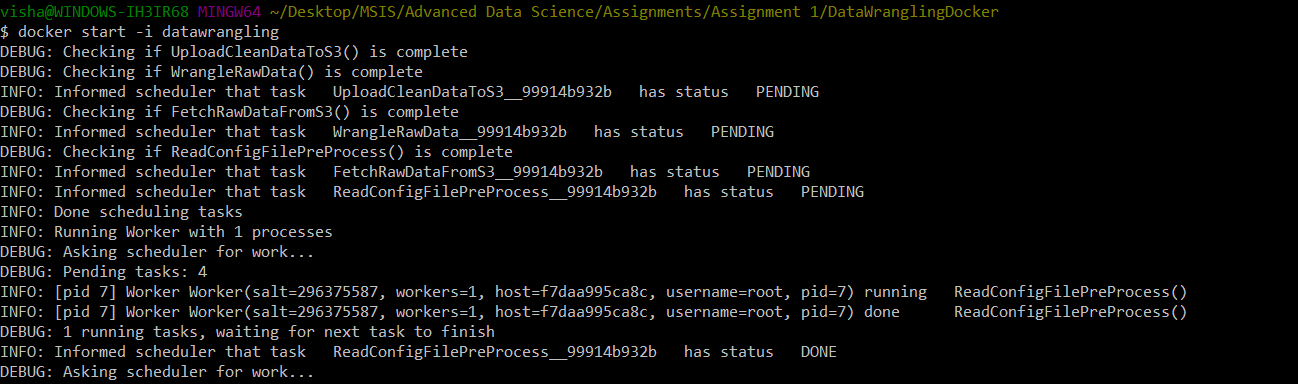
Sample configWrangle.json file



**Step 4 : Start the container**

docker start <containername>

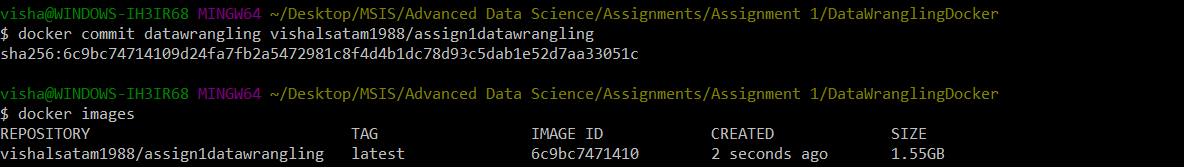
docker start -i datawrangling



**Step 5 : Commit the container to persist the changes**

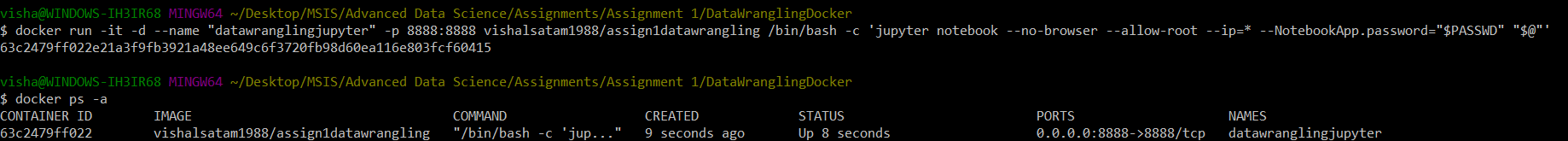
docker commit <containername> <new image name>

docker commit datawrangling vishalsatam1988/assign1datawrangling



Step 6 : Run the jupyter notebook on the committed image in detached mode

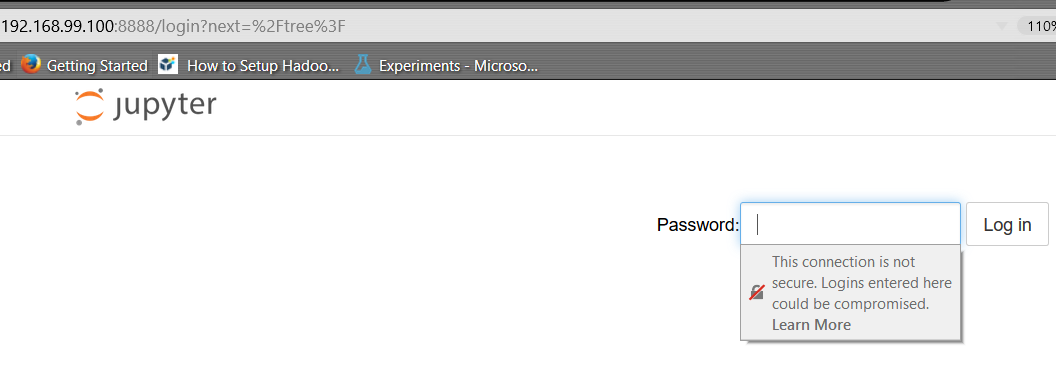
docker run -it -d --name “datawranglingjupyter” -p 8888:8888 vishalsatam1988/assign1datawrangling /bin/bash -c 'jupyter notebook --no-browser --allow-root --ip=\* --NotebookApp.password="$PASSWD" "$@"'



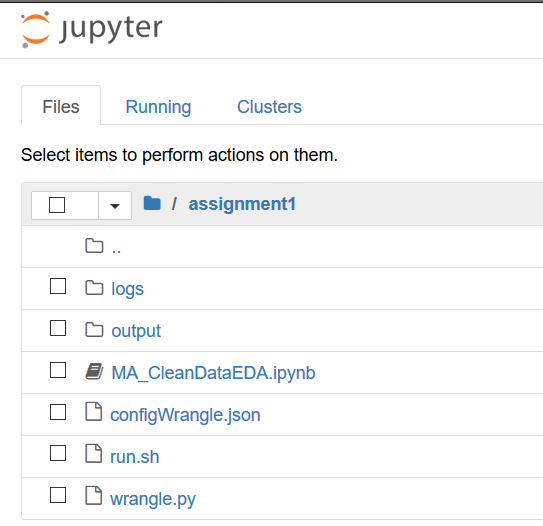
**Step 7 : Connect to this running container via browser by entering**

http://<docker machine ip address>:8888

**Password for the jupyter notebook is keras**

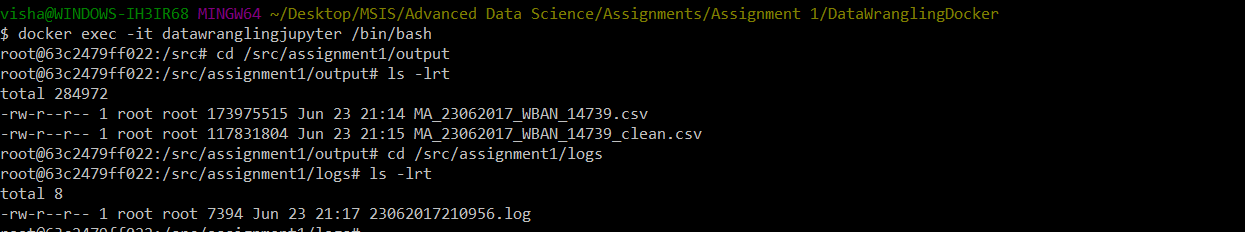


Navigate to the assignment1 folder and open the MA\_CleanDataEDA ipython notebook to see the analysis



**Step 8 : Execute the bin/bash command to enter the running container to check output and logs**

docker exec -it datawranglingjupyter /bin/bash

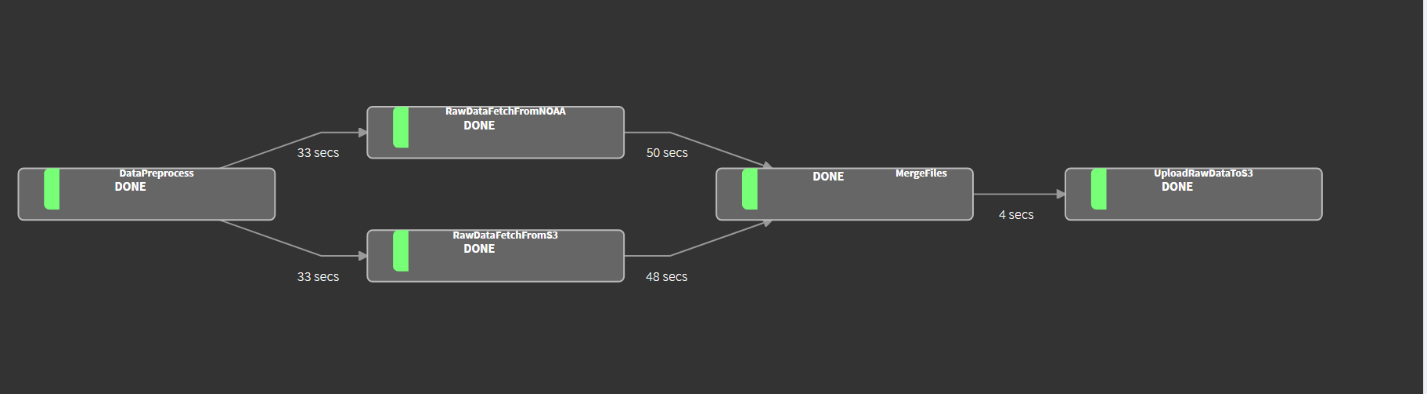


**Pipeline**

The steps for data ingestion and cleaning have been organized into a data processing pipeline. The pipelining framework that we have utilized for this step is Lugi. We have divided the steps into sub tasks to achieve the process of data ingestion and wrangling.

**Data Ingestion**

The following diagram shows the pipeline that we have built for data ingestion.



**Step 1 : DataPreprocess**

During this step, we check the config.json and configInitial.json file to ensure the validity of these files.

**Step2 : Fetching Data**

This step is divided into 2 tasks.

RawDataFetchFromNOAA – This task fetches the raw data from the NOAA website using the links defined in the config.json / configInitial.json file based on whether the S3 bucket is empty or not.

RawDataFetchFromS3 – This task connects to S3 and checks if there is any data present on the bucket. If there is no data on the bucket, then we don’t fetch anything. If there is data on the bucket, then the task downloads the latest file present according to the timestamp present in the file name.

The latest file is fetched just inn case we haven’t run the job on a previous date.

**Step3 : Merge Files**

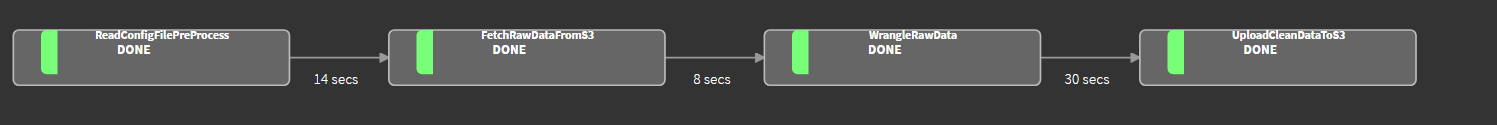
This task merges the data fetched from NOAA as well as the data fetched from S3 and outputs the final file on the local folder.

**Step4 : Upload Raw Data To S3**

This step collects the file returned from the merging step and uploads the data to S3

**Data Wrangling**

We have created a pipeline for this part of the data processing as illustrated in the below diagram



The pipeline has been divided into 4 steps.

**Step 1 : DataPreprocess**

This task checks the configWrangle.json file to ensure that it is valid and there are no errors in parsing this file.

**Step2 : Fetch Raw Data From S3**

During this task, we check for a couple of conditions.

1. If the bucket is empty, then the task logs an error and fails.
2. If the bucket already contains clean data which has todays date in the timestamp, then the task logs an error saying that todays file has already been processed. The task downloads todays clean file in case it is missing from the local directory.
3. If the bucket does not contain todays raw data then, the task fails by logging an error.
4. If the bucket contains todays raw data file which was put here by the data ingestion pipeline, then this task proceeds to download the data.

**Step 3 : Wrangling Raw Data**

This task performs the cleaning and wrangling on the raw data as explained in the Wrangling section of this document and outputs the final clean file to the local directory.

**Step4 : Upload Clean Data To S3**

This task collects the file from the earlier wrangling task and uploads the final Clean data file to the S3 bucket.

**Automation**

The docker containers have been automated to run using the run.sh scripts. These scripts internally invoke the luigi data processing pipelines. The main strategy for automation of these scripts is by setting up a cron job as described in the next section.

**Batch Job (Cron)**

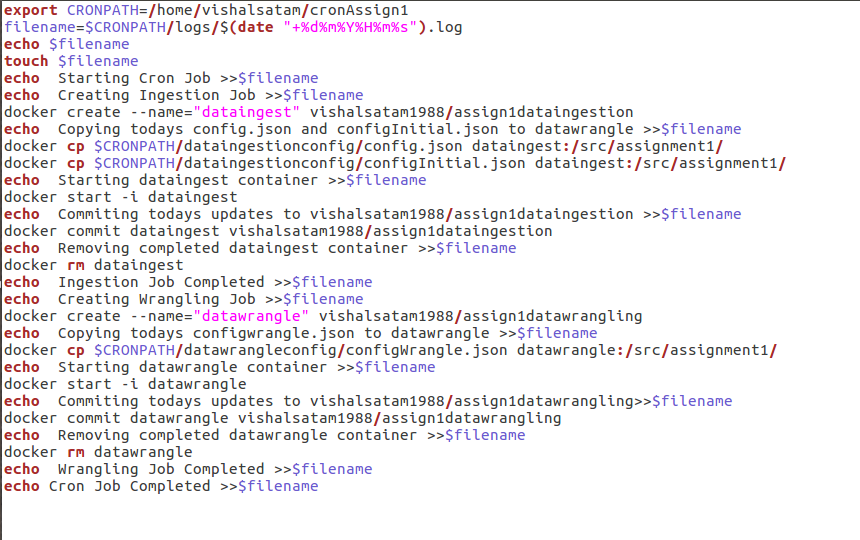
The Batch job has been designed in such a way that we will have to follow a protocol to put the config files into a folder on the local file system. The config files would mainly be updated for the daily link from the NOAA website before 8:00 PM. If one doesn’t follow this protocol, then we will end up running the job on the older links thereby wasting system resources.

The design for creating the batch job is such that we can update the config file with the new NOAA link every day in our local directory (~/dataingestionconfig and ~/datawrangleconfig) and the cron will read from this and execute the 2 docker images. Since, this is a daily batch job, we also stop and remove the running containers. This way, we preserve memory. The changes are incrementally commited to the docker images locally so that we can run our jupyter notebooks for analyzing the data later.

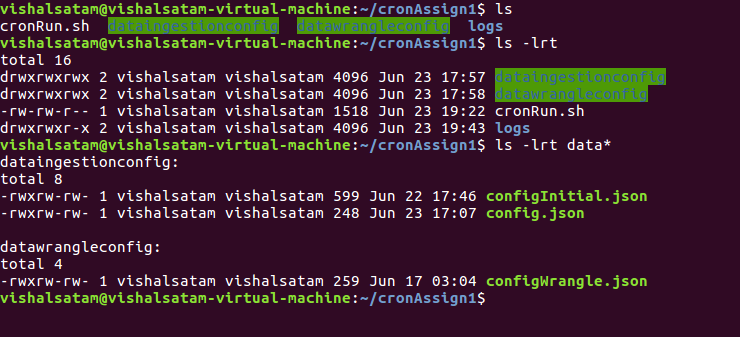
A cron job has been setup using the unix crontab which runs a shell script cronRun.sh to perform the following operations

1. Create a new container for dataingestion and datawrangling every day at 8:00 pm
2. copy the new config files containing the updated links
3. start the created containers
4. commit the changes
5. remove the running containers
6. The shell script also logs the output to a log file to indicate to the user about the execution summary of the cron job.

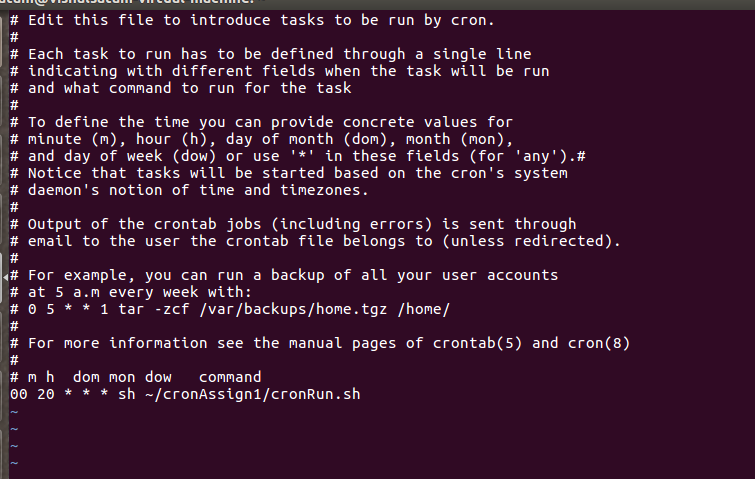
**Shell script that executes the 2 docker images which is scheduled to run at 8:00 PM everyday**



**Folder Structure**

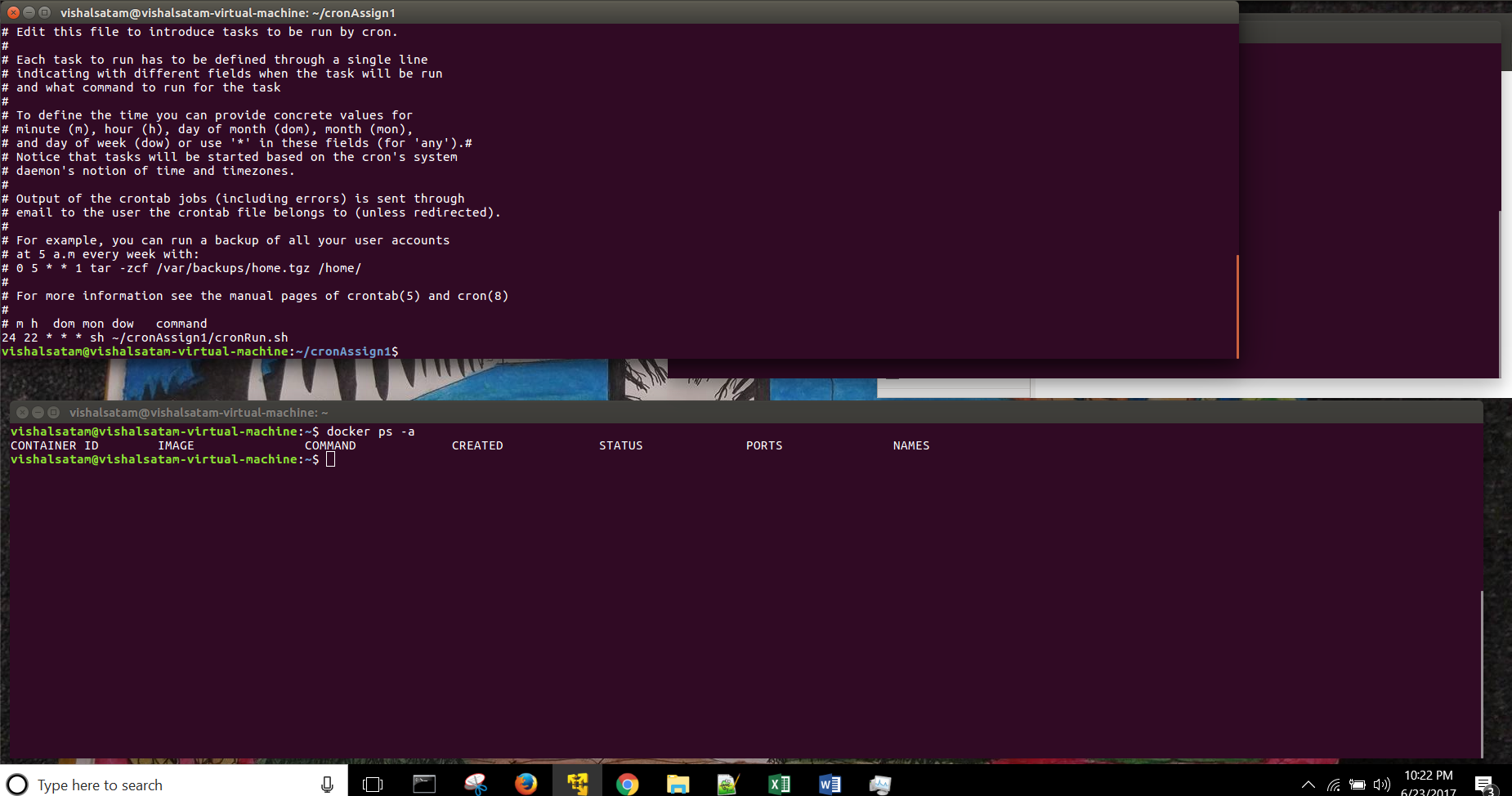


**Crontab is created as follows**

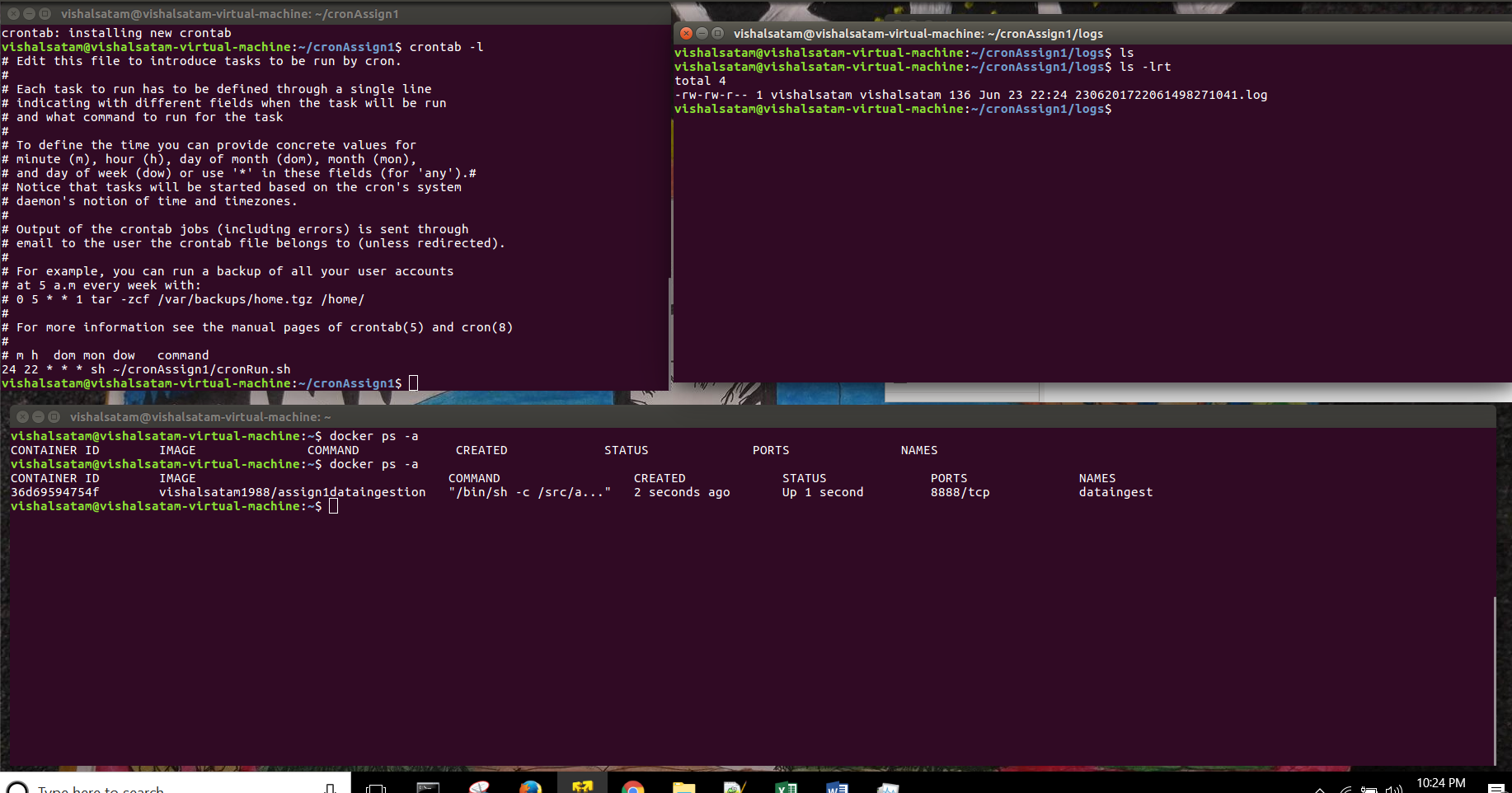


**CRON Demo**

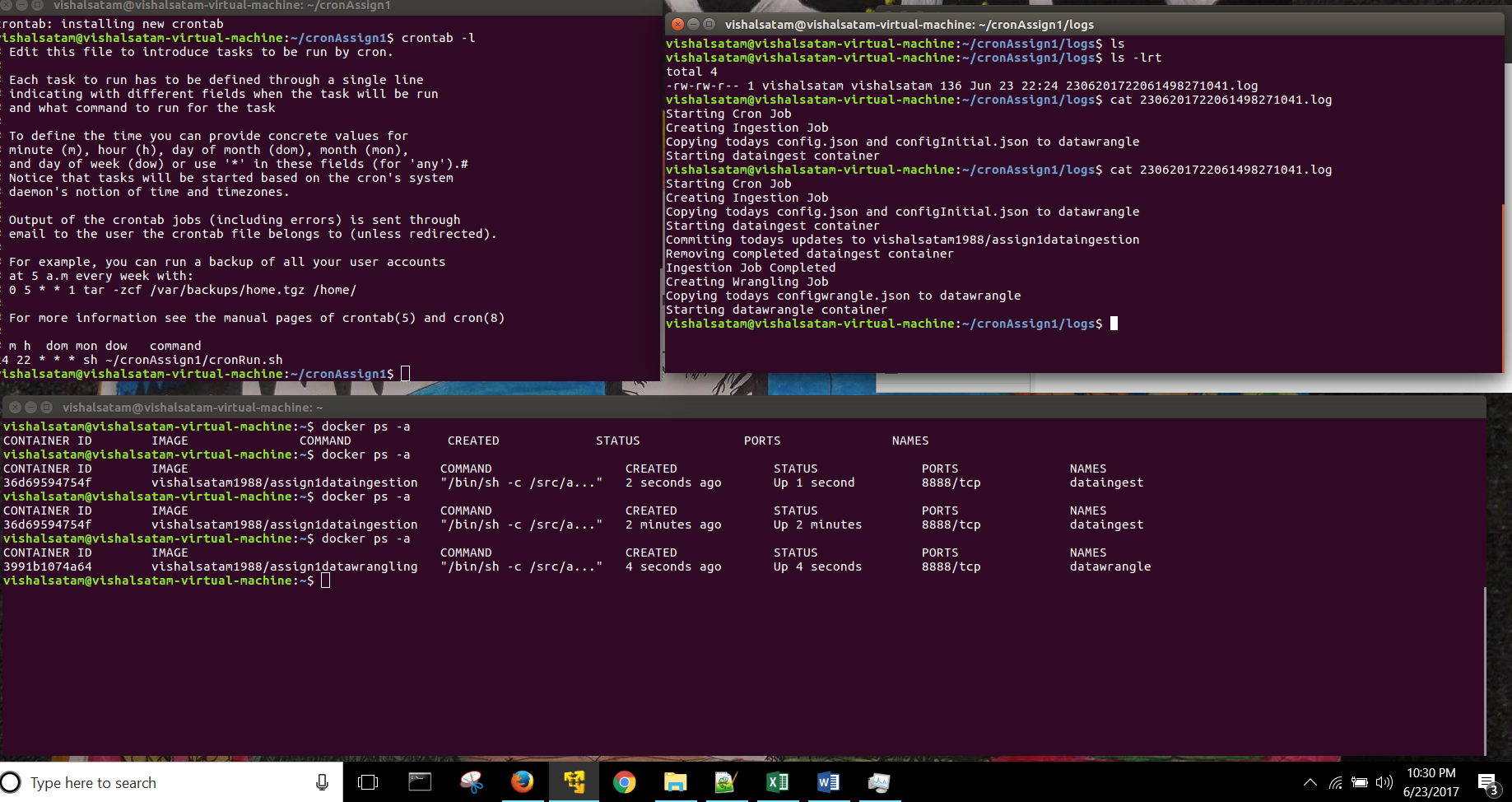
Scheduling the job at 22:24 daily for demo purposes. We should see a log file appear and the container execution starting



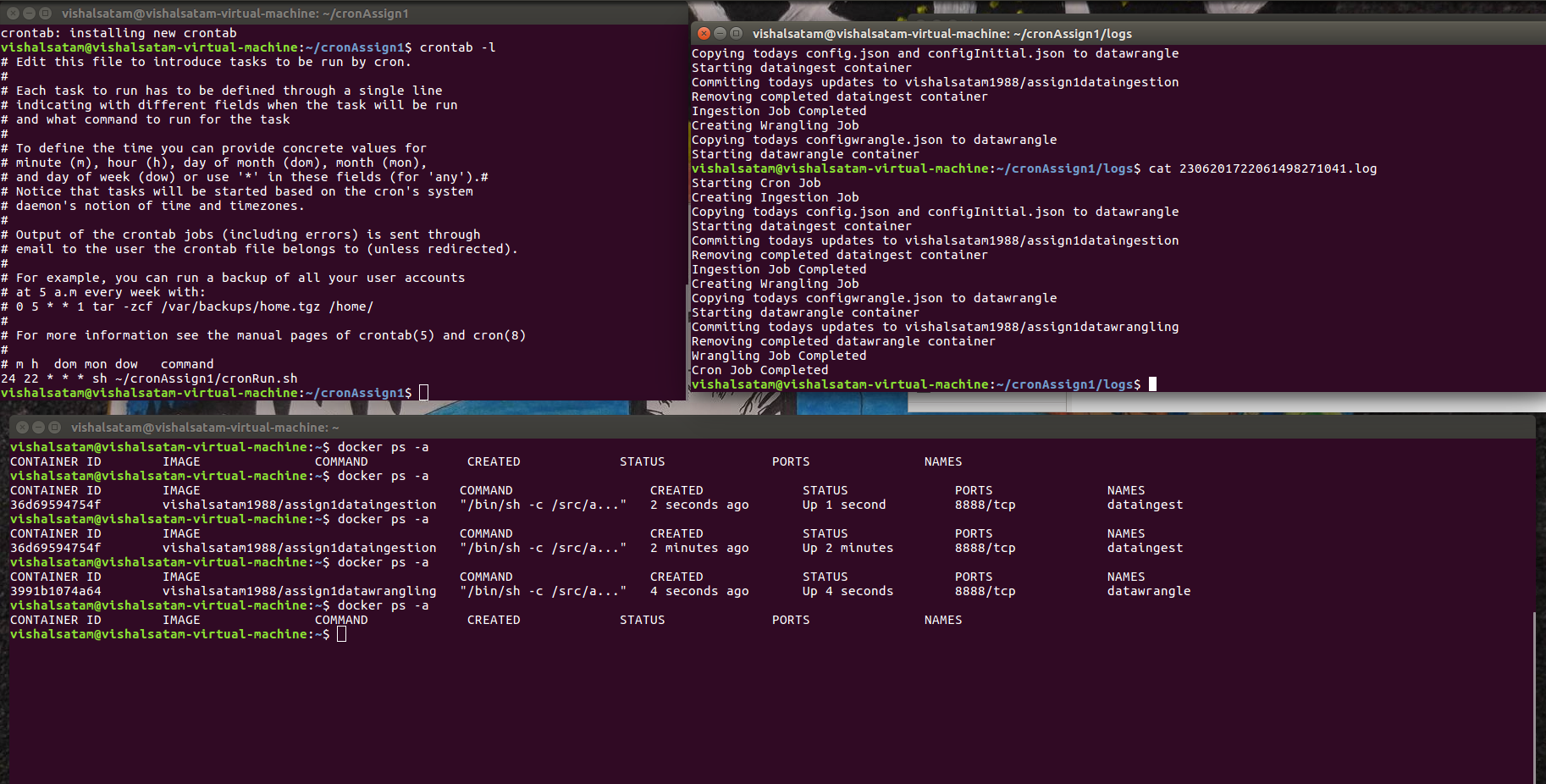
As we can see, the cron job has started execution.



Now we can see that the cron job has commited changes to the data ingestion container and has fired up the data wrangling docker container. Since, this is a daily batch job, we also stop and remove the running containers. This way, we preserve memory.



We can see from the logs that the cron job has completed execution. The 2 docker images were fired up sequentially and the data processing pipeline has been successfully executed.



**EDA on RAW DATA:**

**Dataset Description**: Local Climatological Data (LCD) summaries provide a synopsis of climatic values for a single weather station over a specific month. This dataset is for BOSTON MA station.

Total row count : 724623 Columns: 90

**STATION\_NAME**: BOSTON MA US

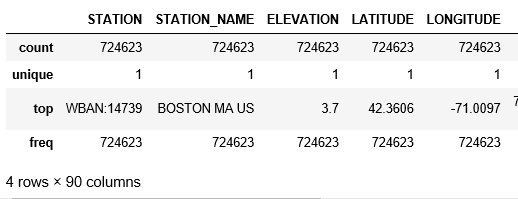
**STATION**: WBAN:14739

**ELEVATION**: 3.7

**LATITUDE**: 42.3606

**LONGITUDE:** -71.0097

**Cleaning strategy**: This data is constant and should not vary. For this dataset there are no missing values for the above columns. In case of missing data replace by median.



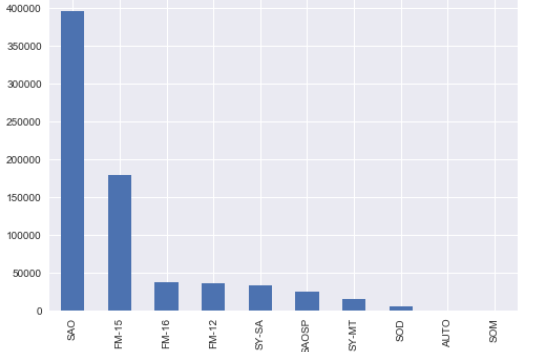
**DATE:**

This column consists of dates and timestamp starting from 11/21/1943 1:00:00 AM to 6/16/2017 11:59:00 PM., for readings of the weather report. It is the Index column and contains all unique values.

Characteristics: Date ranges are inconsistent due to the type of reporting used for the weather data. Data from 1943 to 1993 consists of 17 reports per day while the later data consists of 24 reports each for one hour.

Cleaning: check for nulls. If a value is null discard the row as it invalidates the data reported. For missing time series create day-wise bins for analysis.

**REPORTTPYE:**



SAO = Airways report FM-15 = METAR Aviation routine weather report

SAOSP = Airways special report FM-16 = SPECI Aviation selected special weather report

SY-MT = Synoptic and METAR merged report SY-SA = Synoptic and airways merged report

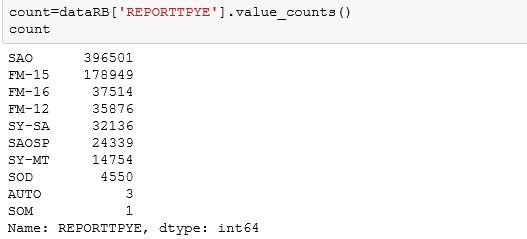
FM-12 = SYNOP Report of surface observation form a fixed land station

We have 10 types of report types as seen in in the above, with maximum types as SAO and FM-15..

SAO and SAOSP are taken till the year 1993 and later are replaced with FM-15 and FM-16. The report types SOD : Summary of Data and SOM : Summary of Month are introduced after 1993 with the FM-15 reports and are missing in the earlier data.

Cleaning:

NO null values found. FM-15 and SAO can be replaced in case of null values as they are taken on periodic basis in a span of 60 minutes. FM-12 and SY-MT can be replaced as they are also taken every three hours. FM-16 and SAOSP are special repots taken to indicate sudden weather changes. If the timestamp is not periodic it can be replaced by SAOSP for data prior to 1993 and FM-16 for later.

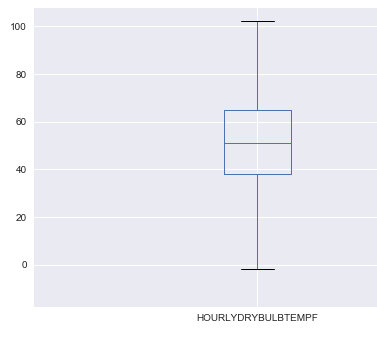
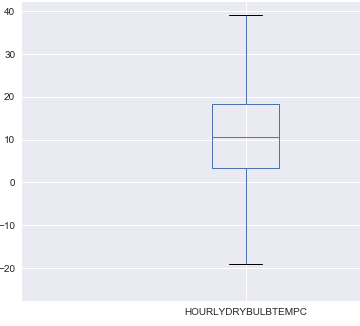


**HOURLYDRYBULBTEMPF & HOURLYDRYBULBTEMPC**

This column represents the standard air temperature and is given in whole Fahrenheit and tenths of degree Celsius.

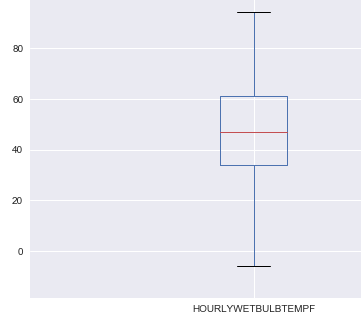
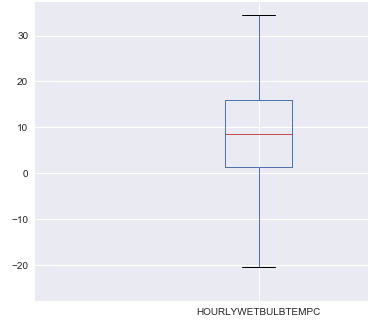
Cleaning: The data varies between -12 f to 101 F with mild outliers suggesting extreme weather. Missing values can be calculated by taking mean of values of the previous and next value. Can use interpolation.

Data missing for either columns can be filled with Fahrenheit to Celsius conversions and vice versa.

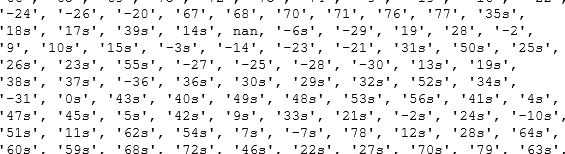
 

**HOURLYWETBULBTEMPF and HOURLYWETBULBTEMPC**

The wet-bulb temperature is the lowest temperature that can be reached under current ambient conditions by the evaporation of water only and is largely determined by both actual air temperature (dry-bulb temperature) and the amount of moisture in the air (humidity).

Cleaning: The data has many null values and has missing values for first two years. It also has character values as shown below. s = suspect value and v = variable value which can be replace by numbers.



For missing data, Interpolation or mean value substitution won’t work. Wetbulb temperature is highly correlated with DryBulb temperature and the Dewpoint temperature and an approximate value can be calculated for conditions where drybulb temperature is equal to or below 65 F. The formula is as follows:

X = DryBulbTemp – DewPointTemp

Y = X/3

Approximate WetBulbTemp = DryBulb Temp – Y

Once major chunks of missing data are filled, interpolation can be tried.

Data missing for either columns can be filled with Fahrenheit to Celsius conversions and vice versa.

[ For more info : [Wetbulbcalculation](http://www.theweatherprediction.com/habyhints/170/) ]

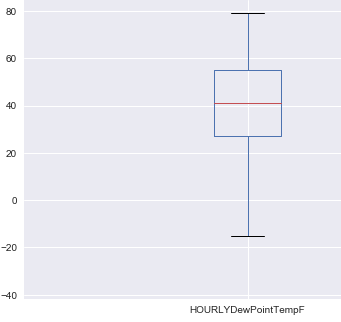
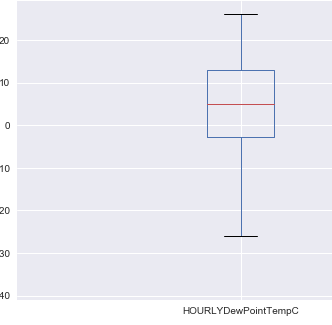
**HOURLYDewPointTempF & HOURLYDewPointTempC**

The temperature to which a given parcel of air must be cooled at constant pressure and water vapor content in order for saturation to occur. It is given in whole Fahrenheit and tenths of degree Celsius.

Cleaning:

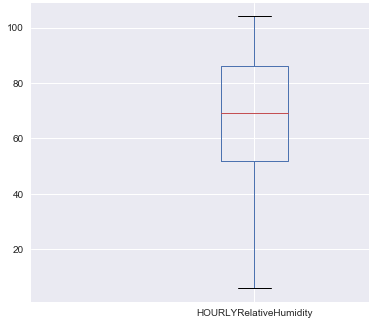
The data has many null values and has missing values for first two years. It also has character values as

s = suspect value and v = variable value which can be replace by numbers. Data missing for either columns can be filled with Fahrenheit to Celsius conversions and vice versa.

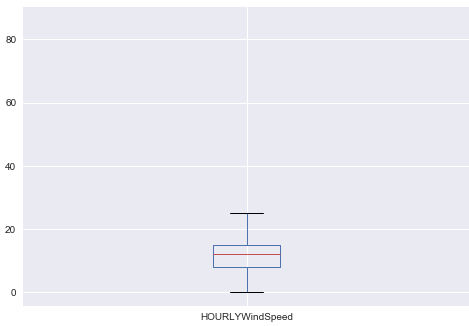
**HourlyRelativeHumidity:**

Relative humidity (RH) is the ratio of the partial pressure of water vapor to the equilibrium vapor pressure of water at a given temperature. Relative humidity depends on temperature and the pressure of the system of interest. The data is varying between 10 to 100 Hg, but with no outlier value, The extreme values can be considered as extreme temperature.



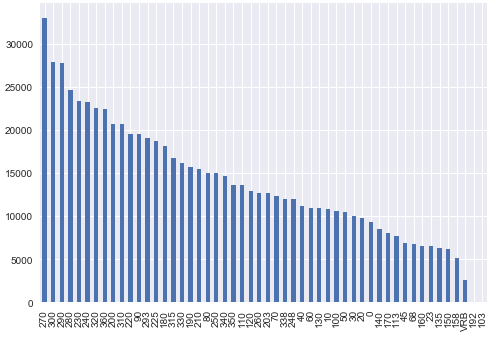
Cleaning: Data contains character values of suspect and variable type , nan values and adjusted values denoted by ‘\*’ can be interpolated and filled in case previous and next value is present.

**HOURLYWindSpeed**:

Speed of the wind at the time of observation given in miles per hour (mph), most common is 9mph. Data has character values, nan values. Values cannot be negative for speed, so a check for negative values. 

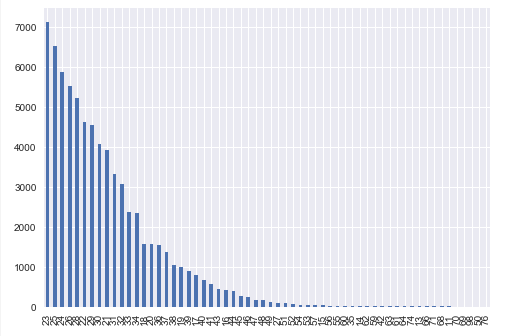
**HOURLYWindDirection:**

Wind direction from true north using compass directions ( 360 = true north, 180 = south, 270 = east) A direction of “000” is given for calm winds. Values cannot be negative for speed, so a check for negative values. Also 0 is not missing value but denotes ‘calm winds’.



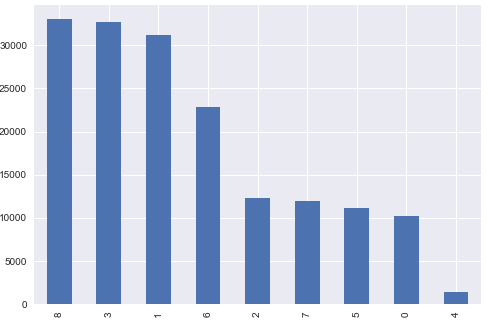
**HOURLYWindGustSpeed**:

Wind gusts occurring during time of observation. Given in miles per hour (mph).



**HOURLYPressureTendency:**

Pressure tendency (In general a 0 through 3 here indicates an increase in pressure over previous 3 hours and a 5 through 8 indicates a decrease over the previous 3 hours and 4 indicates no change during the previous 3 hours).



**HOURLYPrecip:**

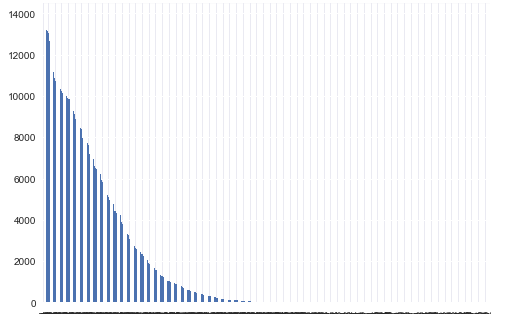
The liquid precipitation observation values given in inches to hundredths and are placed in the cell corresponding to the intersection of the day and hour. All quantities represent what amount of precipitation fell for the hour ending at the time indicated on the table. Trace amounts of precipitation are indicated with a “T.”

Blank/null = no precipitation was observed/reported for the hour ending at that date/time. M = missing

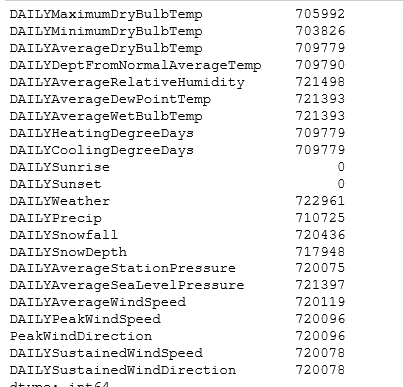
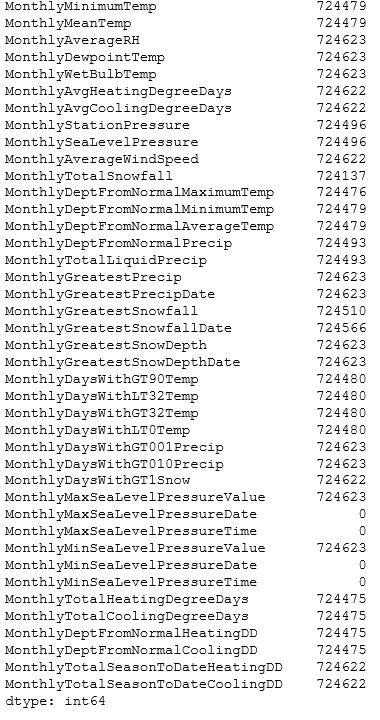
Cleaning: Clean character values, replace T or trace amounts by 0.005 to keep the column numeric.

**HOURLYSeaLevelPressure:**

Sea level pressure given in inches of Mercury (in Hg). Clean out the character values.



**For Daily and Monthly summary:**

**Maximum values are null except for the values for the day and respective months. Can be eliminated to maintain consistency of the Hourly data OR can be retained by eliminating the SOD and SOM rows. and shifting the summary values one row up,ie, on the last timestamp of the day or last day of the month.**

**DATA CLEANING SUMMARY:**

For preliminary cleaning, character values such as 0.9v, 9Vs, 89s are replaced by their numeric types as keeping v and s that is, variable and suspect values don’t add much value to the data. The VRB found in WindDirection column is replaced by -1. The Trace amounts or T found in HourlyPrecipitaion can be replaced by 0.005 as it is less than 0.005 [ mentioned in the document.]

For all temperature[HOURLYDRYBULBTEMPF, HOURLYWETBULBTEMPF,DEWPOINTTEMP] columns with Fahrenheit and Celsius values, we replaced the missing values based on their available counterpart. Calculated the Wetblbtemp missing values by the formula given above.

Removed the SOD and SOM columns and to make the Hourly data consistent for liner interpolation of values having previous and next value available.

**ISSUES FACED:**

Our initial data was from Bedford Station which had missing data for ten years, 1993 – 2003. Switched to BOSTON station.