Observation

As a part of this project, we are going to study and analyse the weather station data collected from 122 stations of southeast region (Brazil). The data set details are as below:

Name: Hourly Weather Surface - Brazil (Southeast region) [1].
URL: https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region

Dataset observation:

1. This data set was provided by National Meteorological Institute, Brazil (INMET).
2. It provides the data of hourly weather details from 122 weather stations of Brazil's southeast region.
3. The data set provides the total of 17 climate parameter values for each hour from 122 weather stations of southeast region (Brazil). The data was provided for below climate parameters for each station [1].
4. The information provided in the data set can be analysed to predict the temperature or predict the amount of rain for a specific region.

Column name	Column description
Wsid	Weather station id
Wsnm	Station name (usually city location or nickname)
Elvt	Elevation
Lat	Latitude
Lon	Longitude
Inme	Station number (INMET number) for the location
City	City
Prov	State (Province)
Mdct	Observation Datetime (complete date: date + time)
Prcp	Amount of precipitation in millimetres (last hour)
Stp	Air pressure for the hour in hPa to tenths (instant)
Smax	Maximum air pressure for the last hour in hPa to tenths
Smin	Minimum air pressure for the last hour in hPa to tenths
Gbrd	Solar radiation KJ/m2
Temp	Air temperature (instant) in Celsius degrees
Dewp	Dew point temperature (instant) in Celsius degrees
Tmax	Maximum temperature for the last hour in Celsius degrees
Dmax	Maximum dew point temperature for the last hour in Celsius degrees
Tmin	Minimum temperature for the last hour in Celsius degrees
Dmin	Minimum dew point temperature for the last hour in Celsius degrees
Hmdy	Relative humid in % (instant)
Hmax	Maximum relative humid temperature for the last hour in %
Hmin	Minimum relative humid temperature for the last hour in %
Wdsp	Wind speed in metres per second
Wdct	Wind direction in radius degrees (0-360)
Gust	Wind gust in metres per second

DATA CLEANING:

- The initial data provided in the given data set has been cleaned using the python script where the data was imported into a panda's data frame, cleaned and formatted the data then exported the resulted data frame into a new csv file.
- All the fields having empty values were replaced with value '0' as taking mean values depends on different scenarios like –
    - taking mean of all the records of that column or

- only the rows present above and below of that record or
- taking mean of all the records present in that week or month or year.

It depends on the business scenario to take the appropriate decision. For this case study, I have just replaced with '0' for all blank cells as per my convenience and to project it as minimum value for any record.

- All the records having value '0' for 17 climate parameters were deleted.
- All the records with special characters have been removed.
- All the columns which are not necessary like year, date, month, day and hour are removed as the data is already provided in a column named 'mdct' (Observation date and time).
- Below is the logic used in my python script.

```
df=pd.read_csv("/Users/akshayreddypoturi/Data_warehousing/case_study/sudeste.csv")
df ['dewp'].fillna(0,inplace=True)
df ['tmax'].fillna(0,inplace=True)
df ['dmax'].fillna(0,inplace=True)
df ['tmin'].fillna(0,inplace=True)
df ['dmin'].fillna(0,inplace=True)
df ['hmax'].fillna(0,inplace=True)
df ['hmin'].fillna(0,inplace=True)
df ['wdsp'].fillna(0,inplace=True)
df ['gust'].fillna(0,inplace=True)
df ['gbrd'].fillna(0,inplace=True)
df ['prcp'].fillna(0,inplace=True)
df ['temp'].fillna(0,inplace=True)
indexNames = df[ (df['prcp'] == 0) & (df['stp'] == 0) & (df['smax'] == 0) & (df['smin'] == 0) & (df['gbrd'] == 0) & (df
               & (df['dewp'] == 0) & (df['tmax'] == 0) & (df['dmax'] == 0) & (df['tmin'] == 0) &
               (df['dmin'] == 0) & (df['hmdy'] == 0) & (df['hmax'] == 0) & (df['hmin'] == 0) &
               (df['wdsp'] == 0) & (df['wdct'] == 0) & (df['gust'] == 0)].index

df.drop(indexNames , inplace=True)
del df['yr']
del df['mo']
del df['da']
del df['hr']
del df['date']
```

Data Model:

Storing the data:

1. The data was imported into My SQL database in the local machine using the csv file which we got after cleaning the dataset using the python script. Load data infile command is used to import the large csv file to My SQL database [2].
2. The same csv file has been used to import data into Mongo DB in local machine using the below command [3].

   mongoimport -d super -c hourlyweathersurface --type csv --file data.csv --headerline

   In the above command, "super" is the database name, "hourlyweathersurface" is the collection used to store the data in database. "data.csv" file has the cleaned data which needs to be imported to database.

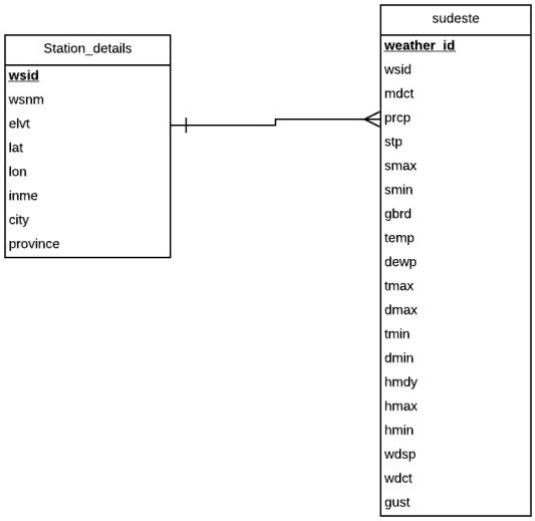
Data Normalization:

The database created using the initial design has only 1 entity with all the details of weather stations and their climate parameter values. The table has been populated with data using the cleaned csv file obtained from provided dataset. No partial dependencies are identified in design but there are few transitive dependencies which were identified, and a new database has been created with normalized data of 2 tables as shown below.

Tables separated in normalization:

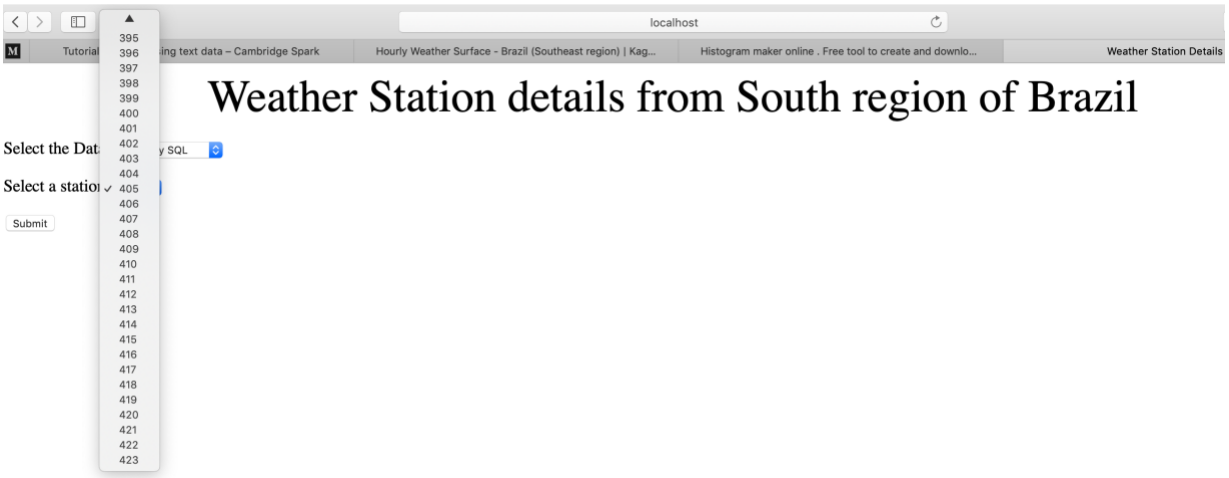
1. There is a transitive dependency in “sudeste” table where the weather station’s details like station name, latitude, longitude, station number for location, city, province depends on the weather station id. So, I have removed the transitive dependency and created a new table called Station_details.
2. “Wsid” is the primary key in “station_details” table and a new surrogate key (auto incremented values) “weather_id” has been created as a primary key in “sudeste” table. Both the tables are joined using the “wsid” field.

My SQL DB structure after converting to 3NF:



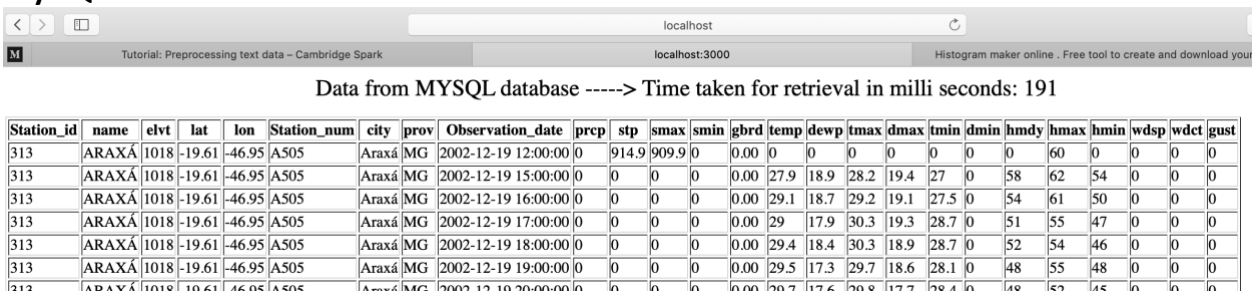
Response Time Measure:

- A simple webpage has been created to access the data from both Mongo DB and SQL database with user’s choice.
- The webpage has been created using the Node JS and Express framework technologies where the EJS is the view engine. The connection to the database’s (My SQL and Mongo DB) has been created from JavaScript code [4].
- When the webpage is loaded initially, it will be displayed with options to select the database and station id as below:



- When My SQL database and a station id is selected and submit button is clicked, the data from database is loaded along with the response time of the data as shown in below screenshot.

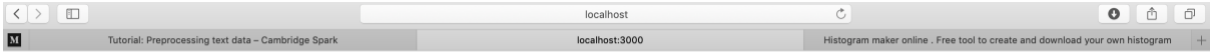
My SQL DB:



Station_id	name	elvt	lat	lon	Station_num	city	prov	Observation_date	prcp	stp	smax	smin	gbrd	temp	dewp	tmax	dmax	tmin	dmin	hmdy	hmax	hmin	wdsp	wdct	gust
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 12:00:00	0	914.9	909.9	0	0.00	0	0	0	0	0	0	0	60	0	0	0	0
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 15:00:00	0	0	0	0	0.00	27.9	18.9	28.2	19.4	27	0	58	62	54	0	0	0
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 16:00:00	0	0	0	0	0.00	29.1	18.7	29.2	19.1	27.5	0	54	61	50	0	0	0
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 17:00:00	0	0	0	0	0.00	29	17.9	30.3	19.3	28.7	0	51	55	47	0	0	0
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 18:00:00	0	0	0	0	0.00	29.4	18.4	30.3	18.9	28.7	0	52	54	46	0	0	0
313	ARAXÁ	1018	-19.61	-46.95	A505	Araxá	MG	2002-12-19 19:00:00	0	0	0	0	0.00	29.5	17.3	29.7	18.6	28.1	0	48	55	48	0	0	0

- When the Mongo DB database and a station id is selected and submit button is clicked, the data from database is loaded along with the response time of the data as shown in below screenshot.

Mongo DB:

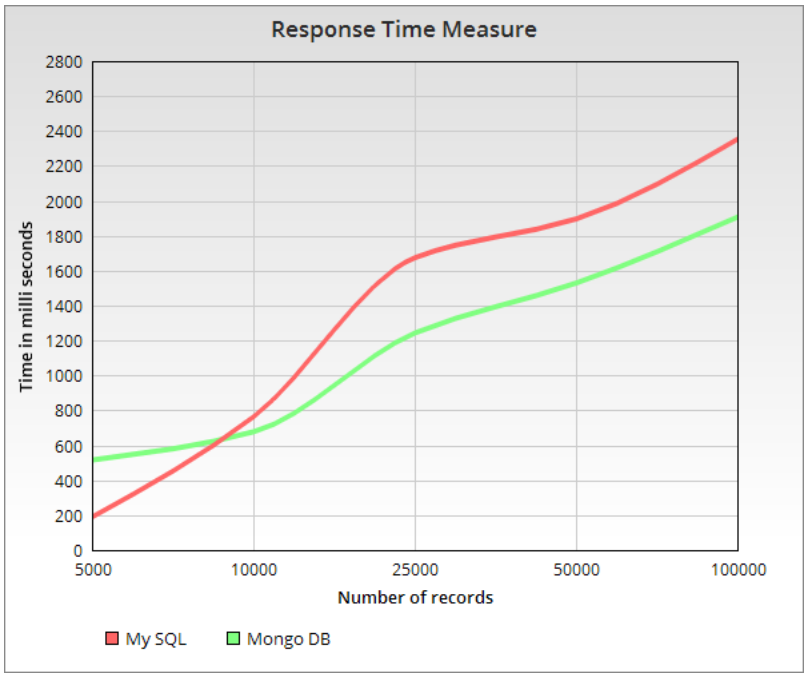


Station_id	name	elvt	lat	lon	Station_num	city	prov	Observation_date	prcp	stp	smax	smin	gbrd	temp	dewp	tmax	dmax	tmin	dmin	hmdy	hmax	hmin	wdsp	wdct	gust
313	ARAXÁ	1018	-19.605696	-46.949616999999996	A505	Araxá	MG	2002-12-19 12:00:00	0	914.9	909.9	0	0.00	0	0	0	0	0	0	0	60	0	0	0	0
313	ARAXÁ	1018	-19.605696	-46.949616999999996	A505	Araxá	MG	2002-12-19 15:00:00	0	0	0	0	0.00	27.9	18.9	28.2	19.4	27	0	58	62	54	0	0	0
313	ARAXÁ	1018	-19.605696	-46.949616999999996	A505	Araxá	MG	2002-12-19 16:00:00	0	0	0	0	0.00	29.1	18.7	29.2	19.1	27.5	0	54	61	50	0	0	0
313	ARAXÁ	1018	-19.605696	-46.949616999999996	A505	Araxá	MG	2002-12-19 17:00:00	0	0	0	0	0.00	29	17.9	30.3	19.3	28.7	0	51	55	47	0	0	0

Below table shows the response times to fetch various number of records from both the databases.

Number of Records	My SQL response (Milli sec)	Mongo DB response (Milli sec)
5000	191	516
10000	765	678
25000	1675	1244
50000	1897	1530
100000	2354	1908

- As shown in above screenshots it was clear that for retrieving large number of records from a database of almost 91,000,00 records, mongo DB took very less time compared to My SQL. From this case study, I observed that Mongo DB efficiently handles huge amounts of data compared to relational databases like My SQL. As the size of data increases, the response time for fetching the data is increasing in both the databases but Mongo DB is fetching faster than My SQL database.
- Below is the graph plotted for response times of both My SQL and Mongo DB database's for fetching different number of records from the data of 9.1 million records [5]. This graph shows how much time difference is there for both My SQL and Mongo DB databases for fetching different number of records for any selected station id.



References:

[1]"Hourly Weather Surface - Brazil (Southeast region)", *Kaggle.com*, 2019. [Online]. Available: https://www.kaggle.com/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region. [Accessed: 21- Jun- 2019].

[2]I. mySql, A. Ivasku, A. Ivasku and V. Kumar, "Import Big csv file into mySql", *Stack Overflow*, 2019. [Online]. Available: https://stackoverflow.com/questions/33944872/import-big-csv-file-into-mysql. [Accessed: 21- Jun- 2019].

[3]"mongoimport — MongoDB Manual", *Docs.mongodb.com*, 2019. [Online]. Available: https://docs.mongodb.com/manual/reference/program/mongoimport/. [Accessed: 21- Jun- 2019].

[4]"Node.js MySQL", W3schools.com, 2019. [Online]. Available: https://www.w3schools.com/nodejs/nodejs_mysql.asp. [Accessed: 21- Jun- 2019].

[5]"ChartGo - create line graph and line chart", Chartgo.com, 2019. [Online]. Available: https://www.chartgo.com/en/chartline.jsp. [Accessed: 25- Jun- 2019].