PROG3070 – ASQL Group Project (W15)

Welcome to our group project! For this challenge, you are provided with a sample ‘legacy data file’ and tasked with creating a proof of concept, weather-reporting system as a web-app. Aspects of SQL performance tuning, ETL, Cloud hosting and data visualization will be required for to complete the project successfully. Your team will provide a results document and will demo your project to me during exam week during our normal class period.

Group Project and Functional Demo

This assignment is meant for group completion (3 people ideally, 4 max). One of the goals of the SET program is to provide you with collaborative, solution development experience.

**BE ADVISED** - This Group Project in this course has two major components, the Project itself and the Functional Demo, and although both are related, they each have their own marking scheme. Both of these components are documented here. There will also be a mandatory checkpoint required on the day of release/review in class.

## Historical Weather Reporting System (HWRS)

A sample data file (CSV) has been provided along with some supporting information. **Your team is tasked with designing and deploying an Azure hosted solution that will allow us to import/ETL historical weather records, and will provide users with an elegant way to generate reports, based on their selection of criteria**. Data charting libraries should be used to provide an appropriately formatted, graphical output from the user’s request. Please read on for more details.

## CHECKPOINT

On the day this project is formally presented in class, teams will be allowed to form, and information and some decisions and early information must be decided upon and submitted to the project dropbox. Information required will be:

Teams choose a ‘group’ in D2L for the project, and roster themselves. Each team submits one document/text file outlining:

1. Team member names
2. Web platform to be used in the project. I leave this to your team to decide, but I encourage you to use one that you are familiar with as time is perhaps too tight to learn a new framework.
3. Charting tools you are considering using. Ideal if this is decided now.
4. Assign roles/responsibilities for each team member, and document them for my reference

Note – a 10% penalty will apply if your team does not have this document submitted by the end of class. You may assign reasonable tasks to an absent team member and of course adjust as the project unfolds.

# Project Requirements (Note there are 3 categories for the Project)

PLEASE: Use a MS Word file to capture the specific requests/results as per below for the project, and submit it, along with any required output files (like .csv or .txt files) in a zip file for the D2L Drop Box by the deadline. Be sure that all team members’ names are on your title page.

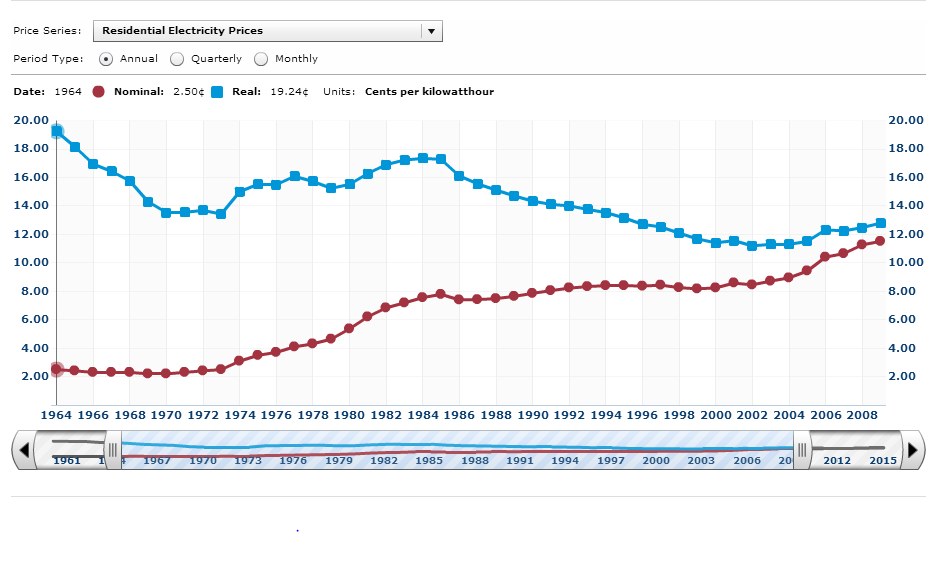
1. ETL On Demand – Your solution should have an ETL capability that will read CSV files formatted like the sample you’ve been provided. Extraction and loading is fairly straightforward and has been covered in other assignments, but we should discuss the expected transformations for the project.
   1. Extract requirement: A User wants to provide you with a user name and service code (e.g. pass code indicating they are approved to upload a CSV file to your service)
   2. Load requirement: The user’s data is loaded into their own schema or tables and kept separate from other users. The intent is that your system may support many researchers looking into historical weather data.
      1. Overwriting – In the case where a user has already loaded data into the Azure DB, you should inform them that a new load will cause all previous data to be overwritten. This should be logged (See below)
   3. Transform requirements: The following transformations are required…
      1. Convert Farenheit to Celsius temperature measures (TMIN, TMAX, TAVG)
      2. Convert inches of precipitation to mm of precipitation (PCP)
      3. YearMonth should not be carried in a single field in your DB (Looking for separate fields)
      4. CDD and HDD require no modifications and should be imported
      5. StateCode maps to a friendly/string name in the supporting docs, please make sure you have this association available in your DB as it will be needed for end-user operations and data labelling
      6. Division, PDSI, PHDI, ZNDX, PMDI and all SPxx fields may be dropped/not imported
   4. Logging requirements: Your solution should log ETL attempts and results. Include user name, time of ETL attempt, time of completion, Range of dates (From: Year/Mon To: Year/Mon), if the user had data before and if they agreed to overwrite (see above).

Evaluation:

* In your results document, describe your approach to Transform Requirements i – iii. **(15 marks)**
* For the demo, I will provide my own copy of historic weather data and will attempt to load the data using an existing user name (e.g. to test your overwrite/logging as well). **(25 marks)**

1. Live Graphing - After the data is loaded, we’d like to provide the users some ability to explore the data the data they’ve provided. The ideal design would function much like the ‘Real Prices Viewer’ located at: <http://www.eia.gov/forecasts/steo/realprices/> A screenshot and some further details are provided below.
   1. Series requirements (15) – You need to provide three series – Precipitation, Cooling Days/Heating Days (CDD/HDD plotted on the same series), Temperature (Min/Max/Avg on the same series).
   2. Region requirement (5)–The user may select one region at a time to explore. Provide an elegant way of letting them select the desired region.
   3. Summary Data requirement (10) – Users will want to select Yearly, Monthly or Quarterly plots. Your data is provided in monthly records, so if the user were to select Precipitation, Quarterly your solution must aggregate appropriately for the rendering of the series (4 data points per year to be displayed).
   4. X/Y Range requirement(s)(10) – Your solution must dynamically set the lower/upper bounds of the X (time) and Y (mm, C or days), leaving a bit of room at the top of the chart to enhance readability.
   5. Time Slider requirement (15)– As with our sample site, you should provide a ‘time slider’ that allows the user to dynamically change the time range of the data being viewed in the chart above. Ideally, the chart should not have to be refreshed once the range is changed. However, your implementation won’t be held to this (They seem to be cheating with Flash Player!)
   6. Printing(5) – We want the user to be able to print their graph with the series and ranges visible on the screen. Allow them to print to PDF using your choice of tool.

Sample of Real Price Viewer (Time Slider at bottom)



Evaluation:

* In your results document, describe the libraries and general approach to this section at a high level. (e.g. how did you decide to handle the aggregate calculations, setting dynamic ranges and slider… e.g. all 6 requirements). This should be worth about 1 – 1.5 pages at most. **(20 Marks)**
* For the demo, once my data is loaded, I want to use your data viewer to examine my data. I want to try all series outline above, and alter the time slider to get the chart I need for my meeting. I want to print this chart to PDF as well. **(60 marks, points as above)**

1. DB Design/Implementation – It is imperative that we make the best use of the DB backend for this project, and expose as little as possible of the DB schema or processing in the frontend. To that end, you are asked to implement stored procedures, functions, triggers, views and indexes as needed to keep the frontend as ‘lightweight’ as possible. For example, this means that you should pass parameters from the UI to a SProc in the DB that will then return the requested data for rendering, rather than pull all the raw data into your front end for crunching. There are at least 3 requirements that would need this approach, and easily some others.
   1. Schema Diagram – This is not a one-table solution, but it should not be endlessly normalized either. Provide a rationale for each table explaining why it meets 1-3NF. Diagram should also show any non-clustered indices
   2. Appendix of SQL – In an appendix to your results document, provide a copy of all stored SQL (SP, Functions, Triggers, Views, non-clustered index creation scripts)

Evaluation:

* The above should be added to your results document for 10 marks each **(20 Marks)**

Breaking up the Workload

At this time of the semester, most of us are looking towards the final few weeks with a bit of dread at the workload remaining. For a project of this scope to be manageable by your team you should get to work on it right away. The Checkpoint detailed above is intended to afford you the best possible start to this work. In the checkpoint document, you will be asked to assign tasks or areas of responsibility to the members of your team. Those areas are not limited to areas of the requirements above. You could assign documentation layout, testing, infrastructure tasks and so on. It is vital that every member of the team participate in the project, and thus during the demo, I will ask each team to complete a workload report – a short form that outlines the relative percentages of work undertaken in the project by each member. Showing me a good distribution of work was achieved will net up to 10 additional marks for the project.

Marks Overview

Results Document – 55 marks

Functional Demo – 85 marks

Workload Report – 10 marks

**Total = 150 marks**

*Note – 10% Penalty will apply if the checkpoint document is not completed and submitted in class.*