# Charter – SQL Query Result to JSON Document Utility

## Paychex IT

* Paychex IT professionals develop software and services
* Project Sponsors
  + Development Liaison: Andrea Brown ([abrown@paychex.com](mailto:abrown@paychex.com))
  + Development Managers:
    - Jon Bigg ([jbigg@paychex.com](mailto:jbigg@paychex.com))
    - Julia Ortiz ([jaortiz@paychex.com](mailto:jaortiz@paychex.com))
  + Technical Resource:
    - Pawel Linek ([plinek@paychex.com](mailto:plinek@paychex.com))
    - David Johnston ([dtjohnston@paychex.com](mailto:dtjohnston@paychex.com))
* Manager of Project Sponsors:
  + Kamal Kothandaraman ([kkothandaraman@paychex.com](mailto:kkothandaraman@paychex.com))
  + Mike Di Spigna ([mdispigna@paychex.com](mailto:mdispigna@paychex.com))

## Project Outline

Create a utility to convert the results from a SQL query into a JSON document in a generic way. The utility will allow for the connection to a SQL database (such as Oracle) allowing the user to select the table, fields and related tables and fields and export the resulting data to a JSON document.

## Project Goals

* To provide the student team the ability to have a real-world experience developing a product to present to a world-class IT company
* To allow the students to gain development experience by working on and delivering a finished project that meets the project requirements
* To provide the students the experience of making a final product presentation

## Agile Development

* Team will deliver the project using the **Agile** development style. Many references can be found, but he following links have been provided for reference.
* The team should self-organize and work from their backlog either using the **Scrum** or **Kanban** framework.
* Quick Links:
  + Agile – <https://www.atlassian.com/agile>
    - Scrum – <https://www.atlassian.com/agile/scrum>
      * Scrum designates a Sprint cycle, usually two weeks, where the team plans on doing a set amount of work Using Scrum, each story gets assigned a number of points, based on the complexity of the effort, using the Fibonacci sequence (1,2,3,5,8,13)
      * Stories pointed over 5 points should be broken down into smaller stories as much as possible
    - Kanban – <https://www.atlassian.com/agile/kanban>
      * Kanban uses more of a prioritized "to do list", where the tasks are put in a To Do column
      * Work items are then pulled in based on a work in progress (WIP) limit, which limits the number of tasks being worked at a time
      * When a task is completed, the next one in the list can be pulled in
* A JIRA board has been created for the team to use to create and manage the stories and tasks workflow
  + <https://paychex-fayetteville.atlassian.net/jira/software/projects/FAYET/boards/1>
* Teams are usually organized as follows:
  + Team Triangle
    - Product Owner – will be someone from the Paychex development team
    - Scrum Master (like a project manager) – could be the professor or teaching assistant
    - Solution Lead (one of the developers) – should be one of the members of the team
  + Other developers and testers
* Ceremonies
  + Meetings in Agile are called ceremonies as described below:
    - Stand Up - This is a daily 15-minute meeting run by the Scrum Master
      * During the stand up, each person says "what I did yesterday, what I am doing today, and whether I have any blockers"
    - Backlog Grooming - This is a bi-weekly meeting to review the list of work and prioritize what should be worked on next
      * Normally, this is run by the Product Owner to prioritize the backlog, but for this project, it will be run by the team
      * Stories are organized in the backlog in priority order, by putting the next to work on at the top of the list
      * When using Scrum:
        + It is good to have 2 or 3 sprints planned in advance, in case stories in the sprint are finished early and work can be pulled in
        + The team can also determine their "velocity" based on how many points they complete during the sprint
        + That amount is then used as a guide to determine how many points can be put in future planned sprints
    - Sprint Planning (when using Scrum) - A bi-weekly meeting to organize the work that will be done in the next sprint
      * Usually held the day before the new two-week sprint will start
* The Scrum Master's job is to facilitate any needs the team may have, including eliminating any blockers.

## Team Expectations

* Team should be self-organizing
* Team should create a set of "norms", which are basically the rules everyone agrees to on how the team will run in order to hold the team accountable
* Team members should meet for daily stand ups (15 min) as described above
* It is suggested the team designate someone as a solution lead to oversee the solution and development
* To mimic the real world, the developers should review each other's code
* It can be beneficial to do pair programming
* Normally, the professor acts as the Scrum Master to make sure the work is being completed and facilitates any blockers the team may have
* Team will communicate using the collaboration tools described below and can contact the Development Liaison via email

## Collaboration Tools

* JIRA
  + The team will use JIRA to manage the development workflow and the following JIRA project has been created:
    - <https://paychex-fayetteville.atlassian.net/jira/software/projects/FAYET/boards/1>
    - It will just be up to the team to determine what works best for you
  + Tutorials of how to use JIRA for both Scrum and Kanban methods
    - Scrum - <https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software>
    - Kanban - <https://www.atlassian.com/agile/tutorials/how-to-do-kanban-with-jira-software>
* Communication
  + Team can choose to use Slack or any other communication tool to collaborate
  + A webex space has been created as well
    - webexteams://im?space=b6a63540-42af-11ee-ae99-f51995cea131
  + It should be noted that JIRA has a built-in sharing option that allows for communication about the stories or tasks
* Email
  + Team can communicate via email both within the team and to the project sponsors

## Product Requirements

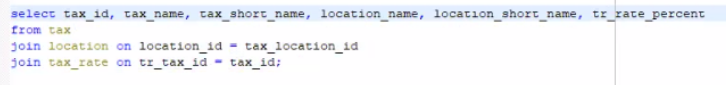
1. Utilize the sample database schema and tables in SQL server or any relational database
   1. Team is free to choose whatever relational database they desire (Oracle is preferred)
   2. A sample schema (tables and fields) will be provided along with a sample data set
   3. The team is free to expand on the sample schema and data provided
2. Create the database with the sample schema and data
   1. Additional data may be added to the sample data set - does not need to be factual, but simply accurate within the scope of the project itself
3. Create an application / user interface:
   1. To allow the user to select the tables, fields, and related tables
      1. Allow the user to determine the parent table
      2. Appropriately nest the selected related tables and fields within the JSON – see example below
   2. Also, to allow the user to optionally insert and execute a SQL query
   3. Display a data table(s) of results
      1. Display selected data
      2. Option needed to export the results to a JSON document
   4. Upload the resulting JSON document to GitHub
   5. User interface built (utilizing AngularJS Version 9 is preferred)
   6. Backend code utilizing Java 11
4. Solution should be agnostic for the relational database, table, and fields
   1. Avoid application specific functionality (for example not implementing functionality specific to SQL Server or Oracle)
   2. Adhere to [ANSI SQL Standards](https://www.itl.nist.gov/div897/ctg/dm/sql_info.html#:~:text=SQL%20is%20a%20popular%20relational,International%20Electrotechnical%20Commission%20(IEC).)
5. JSON document should be properly formatted, preserving types – see example below
   1. Translation may be needed for preserving types – see technical resource
6. Code should follow best practices and project should be properly documented
7. Team is free to go above and beyond and deliver any related items they think would be make the product more impressive

## Definition of Done

* Project will be considered complete when the items listed in the Product Requirements are available and work properly
* Team will deliver a final presentation of the product

## Example Output

Given this example query:



Which yields this output: 

We can expect something similar to the JSON representation below:

### JSON Example

### [

### {

### "tax\_id":2000003,

### "tax\_name":"California Income Tax",

### "tax\_short\_name":"CA IT",

### "location":{

### "location\_name":"California",

### "location\_short\_name":"CA"

### },

### "tr\_rate\_percent":"2.2"

### },

### {

### "tax\_id":2000004,

### "tax\_name":"California Unemployment Insurance",

### "tax\_short\_name":"CA UI",

### "location":{

### "location\_name":"California",

### "location\_short\_name":"CA"

### },

### "tr\_rate\_percent":"3.4"

### },

### {

### "tax\_id":2000005,

### "tax\_name":"New York Income Tax",

### "tax\_short\_name":"NY IT",

### "location":{

### "location\_name":"New York",

### "location\_short\_name":"NY"

### },

### "tr\_rate\_percent":"6.25"

### },

### {

### "tax\_id":2000006,

### "tax\_name":"New York Unemployment Insurance",

### "tax\_short\_name":"NY UI",

### "location":{

### "location\_name":"New York",

### "location\_short\_name":"NY"

### },

### "tr\_rate\_percent":"4.025"

### },

### {

### "tax\_id":2000007,

### "tax\_name":"Federal Income Tax",

### "tax\_short\_name":"FED IT",

### "location":{

### "location\_name":"United States",

### "location\_short\_name":"US"

### },

### "tr\_rate\_percent":"22"

### },

### {

### "tax\_id":2000008,

### "tax\_name":"Federal Unemployment Insurance",

### "tax\_short\_name":"FED UI",

### "location":{

### "location\_name":"United States",

### "location\_short\_name":"US"

### },

### "tr\_rate\_percent":"0.6"

### }

### ]

### Sample Data Schema

