**Property & Casualty Insurance Data**

**Reporting Project**

For **BriteCore’s** Data Team

By **Van F Wilson,**

Data Engineer,

AWS certified

Full Stack Developer

1 (415) 682-3810

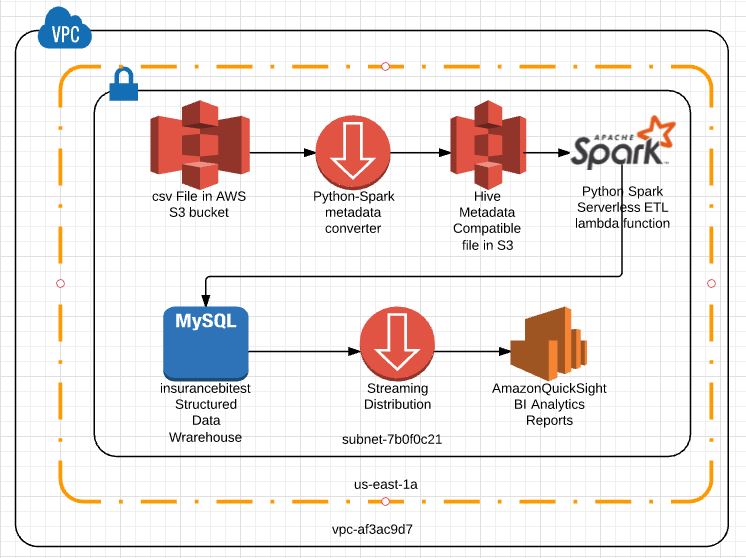
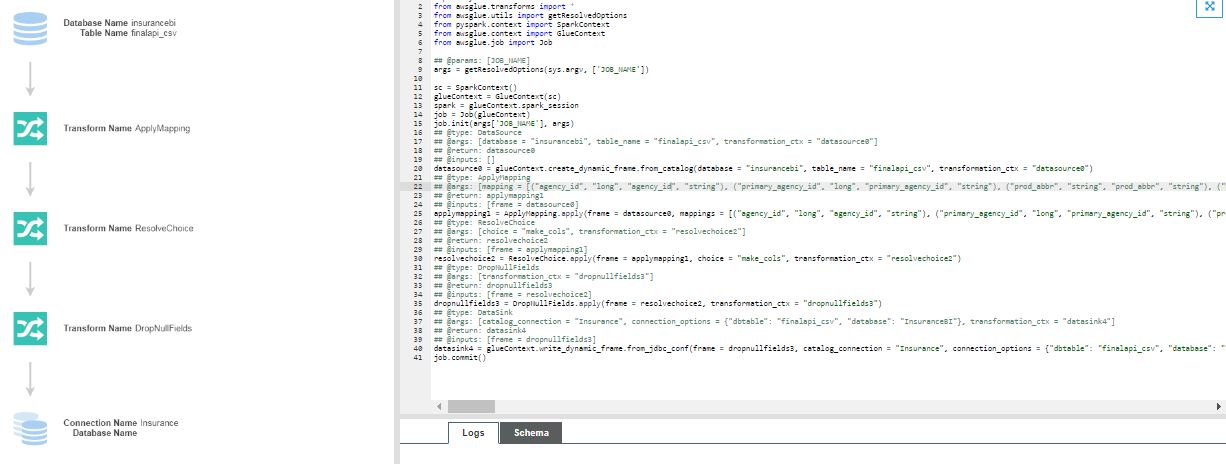
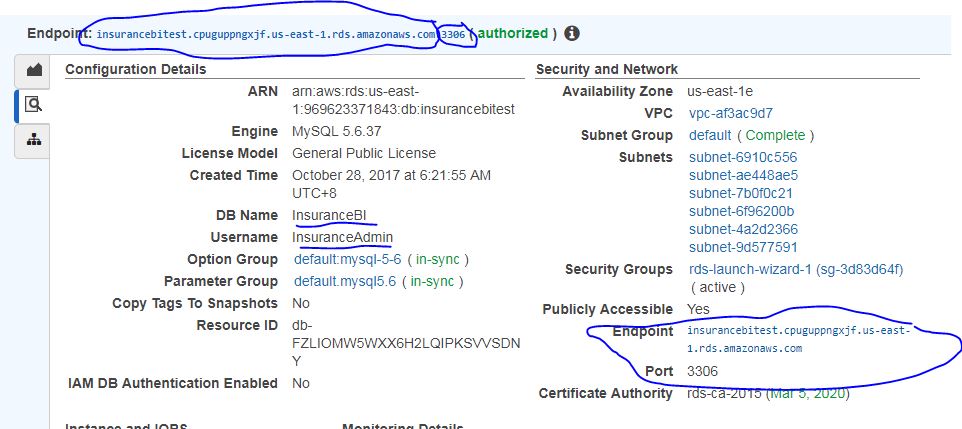
Skype: van-wilson of Hayward, CA

https://www.linkedin.com/in/v-francis-wilson-b66ab96b/

Contents:

* API with unittest and Documentation
* Finished Web App on AWS EC2 using python, pandas, flask, Mysql
* AWS Spark Streaming data to RDS then QuickSight

In agile fashion I formed a prototype set of final reports in Tableau. I used a limited set of dimensions and cubed measures for time sake and simplicity of use case (who would view this report) This tableau file: ***finalapi.tde*** included in the zip file can be pened with a tableau viewer at https://www.tableau.com/products/reader

* I completed 3 versions of this task:
  + a required API, using unittest.py for unit testing, and documentation
  + a finished website to display the graphed data on plotly on flask,
  + and AWS SPARK, Hive streaming infrastructure and displaying on AWS QuickSight. (Im AWS certified).
* **API**
* See ***API documentation.docx*** in the attached zip file.
* Data Pipeline/ETL process takes 1 CSV file stored in AWS S3 bucket as input and converts the data into separate tables as already calculated summarized views. The views were designed in a data warehouse schema optimized for the reports generated and ease of queries of the measures (KPIs) Written Premiums Amounts and New Business Amt against these various dimensions :
* The Data Pipeline/ETL can be manually triggered by running a Python 3.5 with Panda script on Ubuntu on a EC2 instance on AWS. instructions of how to do it are in the user guide.
* Build an API based on REST in the URL (instructions for use in the developers guide) :
  + We chose to use Written Premium amt as the key metric/ measure/ KPI. I did this because my wife is an insurance agent with licensees under her agency. I have built reporting systems for her and her team from corporate sales data like this. She frequently wants to see both commission schedules for salespeople as well as agency reports as part of executive management type reports. This data is incomplete however for each year. So a year over year comparison would be inaccurate. I did not use month because it’s not really a transactional date, it’s the number of months of data was collected. Thus to be accurate we would have to have equal number of months per year. Even then month data are not the same calendar months so it would not display peak seasons. I am nitpicking about accounting to demonstrate I am not only a proficient developer but understand accounting, business and insurance industry. I worked for a real estate property management firm building their online reporting. As such we had many property and casualty policies with details in the data structures I developed.
  + Summary and Detailed Reports of Written Premium amount using dimensions: agency, master agency, vendor, product, product line, year, state
  + Download an XLS, XLSX or CSV report with Premium info by Agency and Product Line using date range as parameters
  + The code was developed using unittest.py for unit testing logic as described in the user guide. This is so the front end developer can unit test the api against their code.
* **Finished Website deployed on AWS EC2 instance**
* See ***User Guide.docx*** in the attached zip file.
* We deployed a final solution displaying the reports on AWS EC2 using python 3.6, pandas, plotly, Flask. To see this solution launch a browser on
* <http://ec2-54-166-152-103.compute-1.amazonaws.com/plot/ProdLineYears>
* Or http://54.166.152.103/
* **AWS SPARK Hive Infrastructure**
  + **Final Quicksite charts**
  + https://us-east-1.quicksight.aws.amazon.com/sn/analyses/148ded80-1158-487e-9f5b-7b17e107a3c0
* We also deployed a AWS infrastructure to stream process the csv in big data fashion.
* 
  + The csv is stored in AWS S3
    - Pem security key for access is in insuranceec2ubuntupythonmysql.pem file
  + The csv is Loaded and parses the data and metadata using python spark code in glue shown below:
  + 
  + *Note: I finally shut down the glue protocol because this is a free project volunteering my time. Though I used as many free services as possible, I automated the infrastructure creation with billing tags I monitored. When I noticed my AWS bill started to rise rapidly I shut it down after I successfully transferred the data. Today I just show the QuickSight reports If you wish to access security, spark, glue, or other services I can walk you through.*
  + transformed into hive compatible metadata tabular files
  + streamed using SPARK on python and sent to MySQL in AWS RDS
  + 
  + it is then streamed using Athena
  + Amazon quicksight displays dashboard like reports as seen at