ECS713 - FUNCTIONAL PROGRAMMING - GROUP PROJECT

Anket Dhoble, Jaikiran Sawant, Suruchi Rastogi

For our project to harvest the data we used a combination of three different .csv files, hosted on three different websites. From there we gathered our data to form two tables on which we will perform queries to show results.

From “http://bhaskarsaikia.in/home/changing\_variable.csv” we will get our .csv files from which we will download the data and store it on the database. The .csv file contains data on real estate transactions for Sacramento region reported over a five day period.It contains data on typeId, street, city, zip, state, beds, baths, sp\_\_ft, type, sale\_date, price, latitude , longitude.

The basic aim of the app is to show the total sales obtained in each category namely house-residential, house-condo, house-multifamily in the Sacramento real estate area over a period of 5 days.

Our first module DataDownload makes the connection to the above selected website and executes the getsalesdata function. Http request is made to the server and the .csv file is then downloaded as a response and returned by the first module.

Our second module DataParser performs convert and parse operations. In this module the data is read from the .csv file and then separated into lines. Here we made use of a different function to break down our .csv file. The .csv file was separated using ‘\r’ instead of the general convention of ‘\n’. To circumvent this problem we wrote a new function :

liner :: String -> [String]

liner "" = []

liner s = let (l, s') = break (== ‘\r’) s

in l : case s' of

[] -> []

(\_:s'') -> liner s’'

This separated the file as expected and but on the basis of ‘\r’ instead of ‘\n’. Then we convert the values into the defined data type.

After successfully parsing the values we reach the third module DataDb which deals with the database operations. It initialises the database as well as perform insert and select queries on the data. Firstly the database is initialised and the two tables are created types and salesdata. Type contains the number of the type of houses and Id and salesdata contains all the details of the property like typeId, street, city, zip, state, beds, baths, sq\_\_ft, type, sale\_date, price, latitude , longitude. Insert functions are successfully used to insert the data using appropriate Haskell datatypes. In this module the sales sum function is used to calculate the total sum of the sale of each type of house i.e residential, condo and multifamily. It takes input from the user and displays the total sum of that category.

Furthermore we have added a separate module DataTypes for our different datatypes.

Lastly we have the main module which is used to run the entire program. All the modules defined previously are imported here. It runs all the other modules and queries the database for the total sum depending on the property type.

To Compile and Run the application we use :

stack build

stack exec Housesaleanalysis-exe