ECS713P - Functional Programming

Group Course Work Report

Team Members
Sri Harsha Musunuru
Sai Krishan Rajesh Chittoor
Sai Siddharth Ponugoti
Mahidhar Krishna Marisetty

Group 20

December 17, 2021

Application

Our application consists of several Haskell files such as

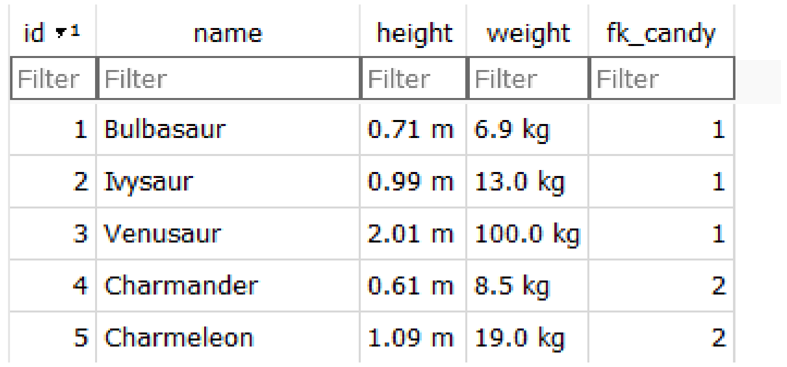
- Main.hs
- Fetch.hs
- Types.hs
- Database.hs
- Paarse.hs

The way our application runs is our **main.hs** provides a terminal to the user where the user can choose what to do with the application and the application.vairous hs files are been imported to main for this purpose,for example if a user decides to downlaod the data from the internet applciation will be usign **fetch.hs** to downlaod the data and then our **parse.hs** parses the JSON type data we have downloaded to the requierd type data which has been defiend in **type.hs** finally we use **databse.hs** to store the values in proper fields in the so called tables with respective column structure defined.The secondary fucntion of **database.hs** is to do the queries asked by the user in the main terminal if the user decides to do the query rather than download the data.

Database

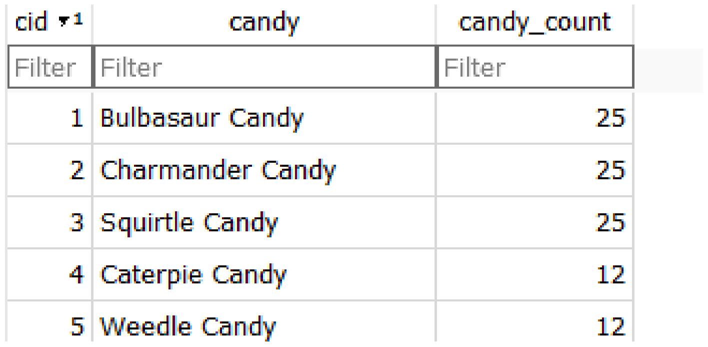
The data which we are using is a Pokedex data which contains information and details about every pokemon (https://raw.githubusercontent.com/Biuni/PokemonGO-Pokedex/master/pokedex.json/).This data is being fetched and stored via **fetch.hs** which gets parsed by **parse.hs** into datatypes defined in **types.hs** which gets stored in **database.hs**.In our case we have stored our data into two tables (PokemonDet & candies).PokemonDet contains details of pokemons and candies contains differen type of candies and count requierd for its type.

-



id	name	height	weight	fk_candy
Filter	Filter	Filter	Filter	Filter
1	Bulbasaur	0.71 m	6.9 kg	1
2	Ivysaur	0.99 m	13.0 kg	1
3	Venusaur	2.01 m	100.0 kg	1
4	Charmander	0.61 m	8.5 kg	2
5	Charmeleon	1.09 m	19.0 kg	2

PokemonDet



cid	candy	candy_count
Filter	Filter	Filter
1	Bulbasaur Candy	25
2	Charmander Candy	25
3	Squirtle Candy	25
4	Caterpie Candy	12
5	Weedle Candy	12

candies

As we can see above our two tables “candies” and “PokemonDet” are seen. The candyID (cid) is used as the foreign key in PokemonDet top establish a connection between the two tables.
The user can do the following operations from the main terminal:

(1) Download data
(2) All Pokemons requiring the same candy
(3) Pokemons with less candies than the entered number
(4) Pokemons with more candies than the entered number
(5) To get the BMIs of pokemons with same candy
(6) Quit

Now since we have our requires tables set up, we can proceed with making the queries.
As per the application requirements we have generated the following queries to achieve the operations.
“queryAllPokemons” fetches us the names of all the Pokémon that has that specific candy type which is given by the user
“queryLessThan” fetches us the names of Pokémon with less than given number candies which is entered by the user
“queryMoreThan” fetches us the names of Pokémon with more than given number candies which is entered by the user
“queryHi” and “queryWi” fetches the heights and weights of the Pokémon which has the candy type that is mentioned by the user.

Additional Feature

As we know the operations that this application does.
In order to achieve the 5th operation that is to calculate the BMIs of the Pokémon that are of same candy type.
to calculate the bmi (weight/height*height)
We expect the values of height and weight to be in Float format but from the dataset that we chose, those were in string format for which units were mentioned for the desired quantity. So after fetching the heights and weights from the dataset we had to find a way in Haskell to remove the units for each values convert it into Float.
Thus we calculate the BMIs of each output and display them in the terminal of the application.