PriceTracker

Live Demo :arrow_upper_right: Feature Video :tv:

Product price tracker, built as a framework, but implemented to track historical prices from LiquorandWineOutlets.com (http://LiquorandWineOutlets.com) . Frontend is a PHP based web-app, and data is stored in a MySQL database. Legacy data was scraped from PDF's using python, and current pricing data scraped from the web using PHP.

Frontend

PHP based frontend currently hosted HERE (http://chrispuglia.com/pt)

To Deploy

On a LAMP server running PHP 5.5+ run

```
git clone https://github.com/ailgup/PriceTracker.git
cd frontend

# create dbCreds.php which contains database credentials

echo "<?php
\$servername = \"localhost\";
\$host = \"localhost\";
\$username = \"username\";
\$password = \"password\";
\$dbname = \"dbname\";
\$db_name = \"dbname\";
\$tbl_prefix = \"\";
\$tbl_members = $tbl_prefix.\"members\";
\$tbl_attempts = $tbl_prefix.\"loginAttempts\";" >> dbCreds.php
```

Now navigating to `frontend/grid.php` will display the frontpage, if it is desired to have the `frontend` directory automatically point to this page then simply run the following from the `frontend` directory

```
echo  "<?php header( 'Location: grid.php' ) ;?> " >> index.php
```

Pages

Grid

	Main page of the site, features items visually with images, allows for search, and ordering based on many criteria. Is not exhaustive in filtering as the table does a much better job in this area. Provides basic info about the product and offers an option to mouse over to preview the graph. Clicking on an item will take you to it's page. Features pagination with a default of 60 products per page.

Table

	Data-centric hub of the site, does not feature all the graphical elements present in the grid, but makes up for it with very powerful sorting and searching tools, in addition to the display of a number of more advanced fields such as Price Per Liter, and Price Per ABV*L. Allowing the user to sort by these gives a much more powerful experience than the grid.

Deals

	Since prices are tracked historically the natural question arises, what are the best deals this month. To handle this the deals page uses an algorithm to rank products given how much the price has fallen from its maximum value while also seeing how much it has changed in the past month. Items at the top of the list will have often fallen 30+% in the past month and will often be at all-time lows, a great time to buy. You can also go to the other end of the spectrum and see the 'worst' deals, items that would not be recommended to purchase this month as the price is at a high point. The layout of the page is very similar to the Grid page.

Product

	This page gives all the details about a given product in a succinct manner. On the right side of the page the items details are listed along with an image of the product. If the user is logged in the quantity of that product available at the given location is also listed. Important data like the price change in the last month, and the price relative to the average are also listed for easy comparison. On the left side of the page the price history graph takes up the majority of the page. With related products listed below for possible consideration.

Account

	This page allows users to view their "tracked" items, see their availability at their selected store, as well as modify their selected store, or tracked products. This is very beneficial as it serves as a single place for a user to view the products that that they are interested in without the clutter of uninteresting products.

Admin Account

	If the user is logged in as an admin they will see the admin button on their user page. Clicking this button leads to the admin account page where more details concerning the backend of the site, especially items and prices that do not fit nicely into the database. The admin can add Items that have appeared in the most recent scrape but have not yet been added to the Items table. This is done by performing a live PHP-based scrape of the merchants site.

Grading Criteria

1. Create
    - New prices are mined using the `mine.py` program and then are inserted into the DB using `csvToSQL.py`
    - Adding to the Tracking table is done by a user clicking the "Track" button and the functionality found in `track.php`
    - New Items are mined using the `mine.py` program and then are inserted into the DB using `csvToSQL.py`
    - New Items can also be mined through PHP through the use of the functionality of `getItem.php` which is linked to in the admin page
    - Users are added through the `createuser.php` functionality.
2. Delete
    - Generally it is not desirable to delete Prices since even if an Item is deprecated it may come back eg. seasonal products so Prices are not deleted
    - Items however can be deleted if they are found to be no longer sold by the retailer, this is done with the `dropProduct.php` file
    - If a user no longer desires to track an Item it is dropped from the Tracking table, this is done with the `track.php` functionality
3. Read
    - Reading is the major function the web frontend utilizes, from `grid.php` to `table.php` to `product.php`, and the list goes on. Every page on the website does some read from the database. Whether it is product into, price info, user info or

some join of these reads are all throughout the program

4. Update
    - The User 'Home store' is something that is updated using the `setStore.php` file, this allows users to change their home store to see product stock at different locations.
    - The `updateImages.php` file also implements an update by scanning the available images and updating the Items table with which Items have images available. This functionality is available from the admin page.
    - Given how the Items and Prices are inserted into the Database, directly from the merchant website and with a fair amount of error checking and data scrubbing it would not be desirable for a user to modify the Item or Price data. Therefore, there is no functionality built into the GUI for updating the Item or Price data
5. User Operations
    - The user has a large number of operations made available including
        - Searching the Items table
        - Ordering search results in `grid.php` by a number of analytical functions
        - Viewing legacy Price data via the JavaScript graphs
        - Viewing related products
        - "Smart Data" not provided by retailer including Unit Price, Unit Price per ABV, % Change from last month, % Difference from average
        - Ability to track products of interest to a user, and receive live product availability from the store they shop at
        - Viewing of the best deals, using an advanced algorithm which using knowledge of price history determines if the current price is a great deal or just par for the course.
6. Modularization (Procedures, functions, triggers)
    - Triggers are used extensively to handle the input of data into the database triggers used in the
        - `CurrentPrices` table calculate the price_per_liter, price_per_abv, the_min (the sale price if on sale, otherwise the list price), perc_diff (percent difference from last month), and avg_dif_perc (the percent difference from the average price)
        - `Prices` table to calculate the_min (the sale price if on sale, otherwise the list price), which is crucial for graphing
        - `Items` table to ensure that if a new Item is created it's associated CurrentPrice is updated with the price_per_liter and price_per_abv
    - Functions are also used in many of these calculations to make it easier on the frontend for these repeated tasks such as
        - `actualVol`- given a Item id and which contains a string based volume eg. (750mL or 1.75L or 750mL 2 Pk) but it needs to be in Liters for the price_per_volume calculation, this function uses regex to extract the actual volume
        - `avgPrice`- given an Item id return the average price of the item over time

    - `getName`- given an Item id return the name of the item
    - `pricePerABV` - given an Item id return the price per liter of pure alcohol eg. ((current price)/(item volume))*(200/proof)
    - `pricePercDiff` - given an Item id return the percent difference from the previous price to the current price eg. (current-past)/current*100
    - `pricePerLiter` - given an Item id return the unit price in $/L of a given item
7. Error Handling
    - As this is a web app errors need to be considered, therefore all database connections contain catch's if the database is unavailable
    - Also on the forefront of concern was the possibility of a SQL injection attack, therefore all inputs were scrubbed before being entered into the queries. Particularly dangerous queries such as deletes and creates are only available behind the secure admin page, therefore restricting access to these sensitive commands.
    - The `products.php` page has error checking built in as does `graph.php` if the product id is not valid it will tell the user "Product not found"
8. Frontend Functionality
    - Web-based frontend fully deployed and online, (receiving actual user traffic from consumers!) Link :arrow_upper_right: (http://chrispuglia.com/pt)
9. Complex user operations
    - Login and user creation, while an open-source solution was deployed the ability to have users sign-up with email confirmation and login attempt limitations is an advanced user database feature
    - Users tracking of products, allowing users to have a home store and track products they are interested in is an advanced feature allowing for greater interaction between the database and the user.
    - The concept of having Prices without Items and Items that ave become deprecated is an advanced feature that was added to allow for representation of the complexity and the "real" nature of the domain. It was recognized that items will come in and out of stock and prices should still be trackable even if an Items is currently unavailable.
    - The Image scraping algorithm used in both python and through the admin panel is an advanced feature used to obtain efficiently and accurately product images.

Backend

To Deploy

On a MySQL server run `SQL\pricetracker.sql`

N.B. All Python Scripts run on Python 3.X Will require PyMySQL Library

To perform new Scrape

Run `Mine.py` as detailed below, can be run as a CRON job monthly to automate the process.

Once `Mine.py` has generated a CSV file `csvToSQL.py` can be run to upload the scrape to the MySQL database

Mine.py

Used to mine product data from the website, currently stored in .csv, ultimately will live in SQL

```
#Price Mine ~5min
python mine.py 1
```

will do **Price** mine and will save to a .csv

```
# Item Mine ~3hrs
python mine.py 2
```

will do **Item** mine and will save to a .csv (Takes much longer)

csvToSQL.py

Used to upload legacy .CSV files to the MySQL database upon completion of scrape

Note: IP address and database name will need to be modified

```
# upload test.csv to Prices table
python csvToSQL.py test.csv
```

ImageScrape.py

Finds the items which lack images and does a scrape of Google Images saving the first image found by searching Product_name+Volume. After running this program and uploading the images to GitHub the Update Images script should be run from the Admin Panel to update the SQL database with which products contain images and which do not.

```
# run google image scrape
python ImageScrape.py dir_to_save_to
```

After running this script is may be wise to manually go through and delete images which are not accurate.