Northeastern University
College *of* Engineering `

Department of Mechanical and Industrial Engineering

IE 3425: Engineering Databases

Final Project Report

Submitted by
Ted Yee

Date Due: 11-15-23, 11:59 PM
Lab TAs: Katherine Polum, Johan Cho, Eva Justice
Course Instructor: Ibrahim Zeid

Editor's Note:

A live (static) version of the webpages is available here: https://tedyee114.github.io/milesplit/templates/index and all code is available on GitHub here: https://github.com/tedyee114/milesplit.git.

Due to Flask needing static resources in a direct subfolder called static and GitHub not using relative filepaths, there is no way to have both the web version and the local version correctly run. My solution to this was to add a check as to whether the site was local and then change the references, which works great locally. On the web version though, GitHub's static deployment means that the HTML does not evaluate the variable, and so everything prints twice, with the correct version at the bottom and the forms not being allowed. Please just scroll down to see what it is supposed to look like.

Also, this project is a copy of the real result database website called MileSplit, accessible here and through the webpage https://wa.milesplit.com/

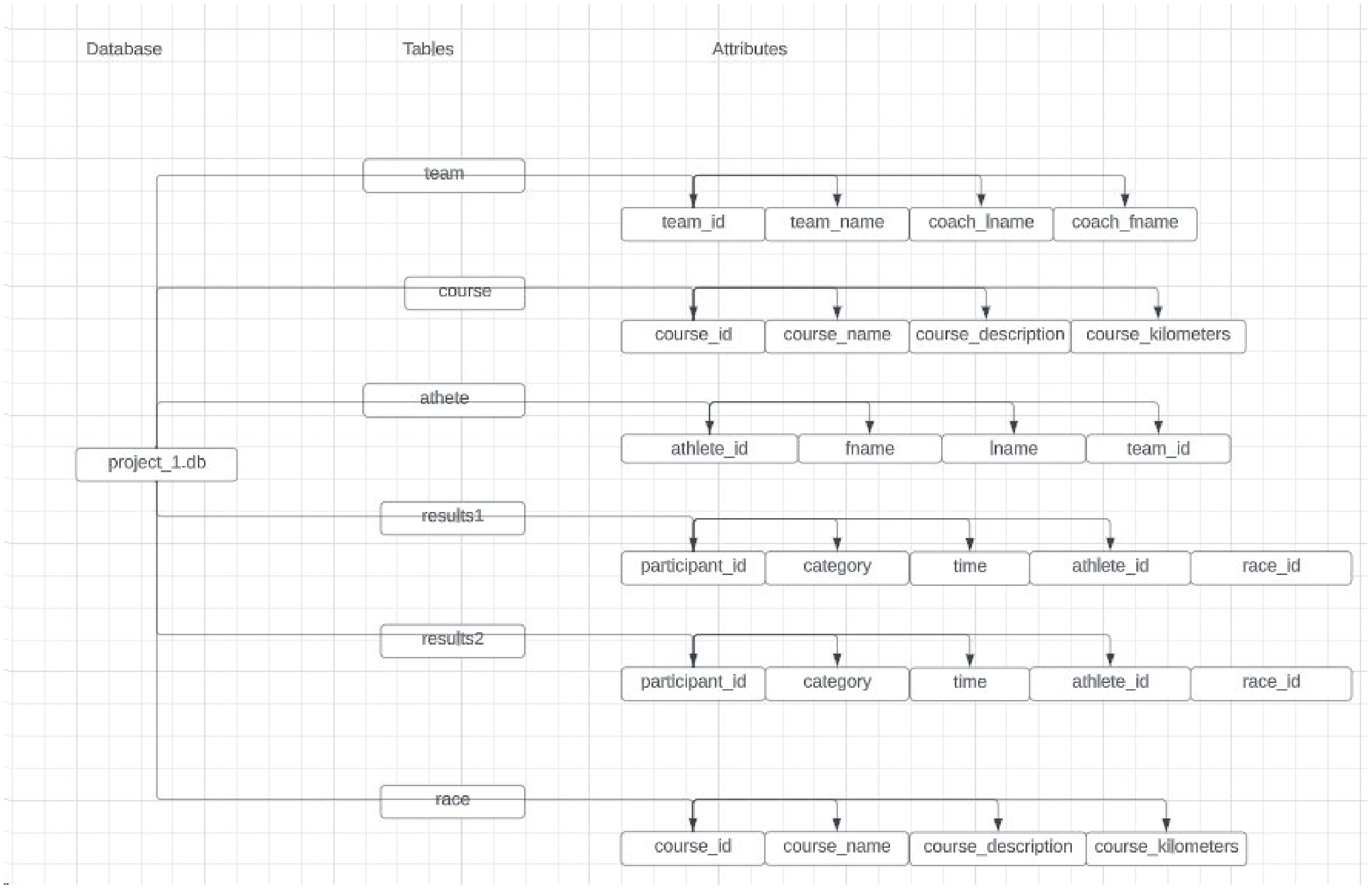
Abstract/Overview

This project is an HTML site which connects via python and SQLite3 to a database file to query and manipulate contained data. The site functions as a knock off the real site MileSplit©, keeping a list of athletes, race results, teams, race schedules, and courses, and makes the information readily available to those using the local version of the site. A web version of the site is available at the link above but is hosted on GitHub and is limited to being static, which prevents queries and evaluation of any sort of variables.

Contents

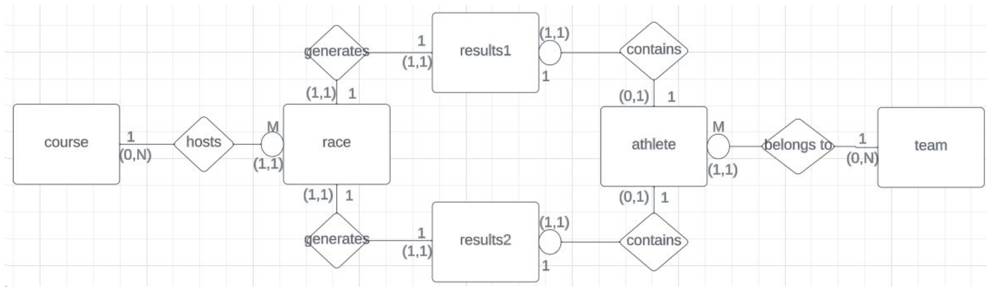
Backend Design

This section is about the design of the database itself. It is designed to be a management system for race results and a way to compile data about results, race schedules, courses, teams, and athletes together. Each of those has a table, as seen below, with an individual table for the results of each race. This way, the database is scalable as new results can be added.

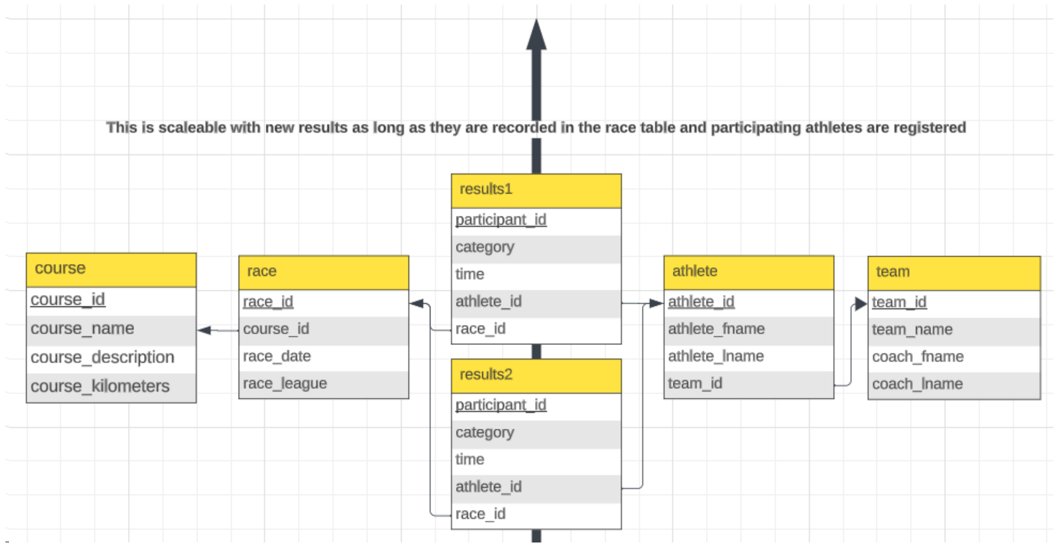


System Data (Meta) Model

The design of the database is fairly simple. The ‘race’ and ‘team tables are independent entities, which are each referenced by one other table, essentially creating two independent chains. These two chains are connected via foreign key references in every results table. In this example of the database, only 2 sets of results are shown (the apparent ring in the ERD), but as more races are added, there will be many results connecting the two pieces.



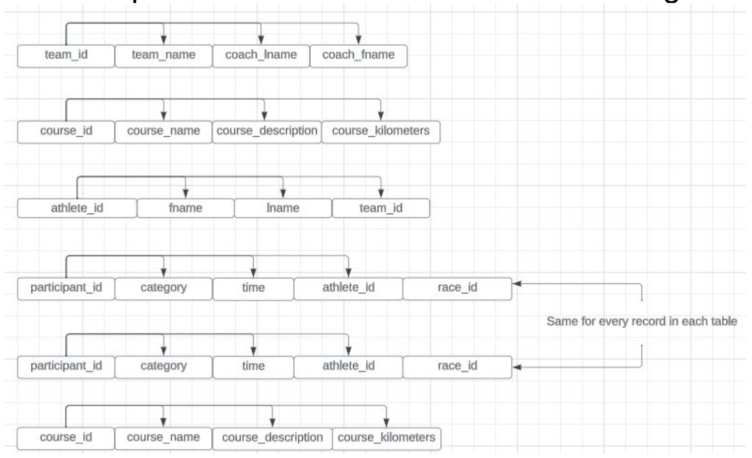
ERD Entity Relationship Diagram


This is scaleable with new results as long as they are recorded in the race table and participating athletes are registered

Relational Schema Diagram showing scalable applications

Note that more results can be easily added as long as the race_id and athlete_id are valid. These could be null, and would make the data a little less clean, but that's not preferrable. Instead, when new results are added that don't have valid FK's, another script could create new ones, or the database could allow null values.

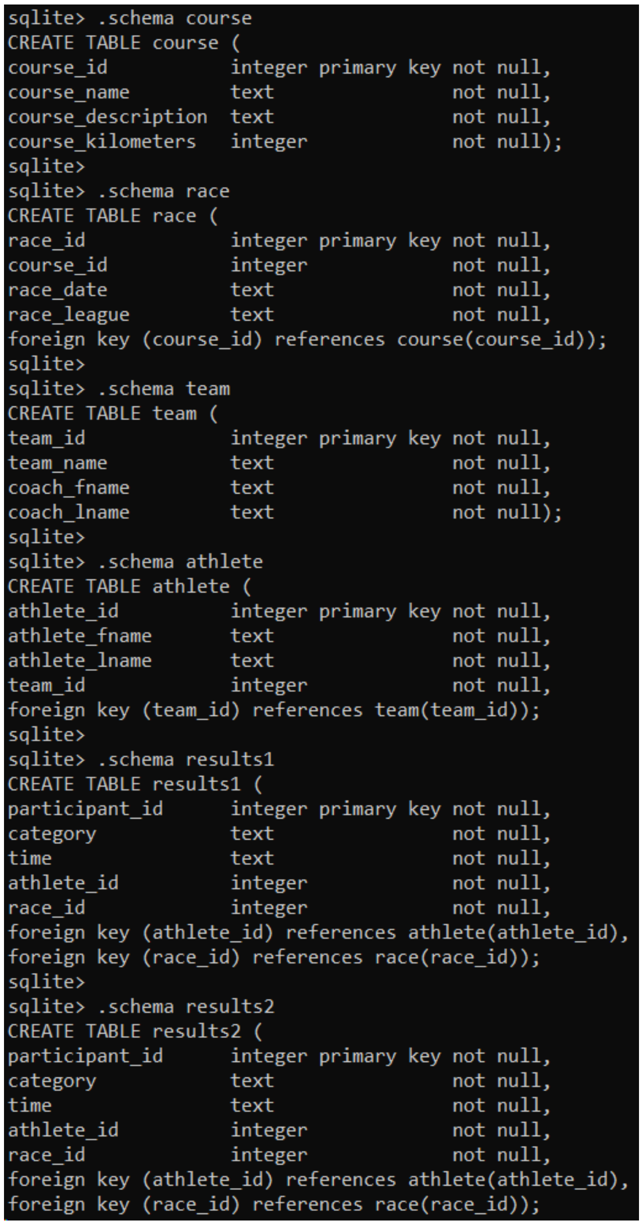
All tables are normalized because all attributes are dependent only on the foreign keys and there are no transitive or partial dependencies. The only oddity is the race_id attribute in each results table, which is essentially table metadata as every record in a given table is the same as the table's race_id. In this way, data can be queried as a variable rather than calling the table's name directly.



Normalization diagrams for all tables.

.

The data dictionary for each table and the .schema results showing how it was inputted into SQLite are shown below. This information covers the basics of what type of information every record can be.



```
sqlite> .schema course
CREATE TABLE course (
course_id           integer primary key not null,
course_name         text                not null,
course_description  text                not null,
course_kilometers   integer             not null);
sqlite>
sqlite> .schema race
CREATE TABLE race (
race_id             integer primary key not null,
course_id           integer             not null,
race_date           text                not null,
race_league         text                not null,
foreign key (course_id) references course(course_id));
sqlite>
sqlite> .schema team
CREATE TABLE team (
team_id             integer primary key not null,
team_name           text                not null,
coach_fname         text                not null,
coach_lname         text                not null);
sqlite>
sqlite> .schema athlete
CREATE TABLE athlete (
athlete_id          integer primary key not null,
athlete_fname       text                not null,
athlete_lname       text                not null,
team_id             integer             not null,
foreign key (team_id) references team(team_id));
sqlite>
sqlite> .schema results1
CREATE TABLE results1 (
participant_id      integer primary key not null,
category            text                not null,
time                text                not null,
athlete_id          integer             not null,
race_id             integer             not null,
foreign key (athlete_id) references athlete(athlete_id),
foreign key (race_id) references race(race_id));
sqlite>
sqlite> .schema results2
CREATE TABLE results2 (
participant_id      integer primary key not null,
category            text                not null,
time                text                not null,
athlete_id          integer             not null,
race_id             integer             not null,
foreign key (athlete_id) references athlete(athlete_id),
foreign key (race_id) references race(race_id));
```

Database Schema

‘course’ Table Data Dictionary

Attribute	Description	Data type	Data Format	Sample Value	PK/CPK	FK?	Derived attribute?
course_id	# assigned to identify course	integer	0000	1	PK	No	No
course_name	name of course/location city	short text	abc	Ruka	No	No	No
course_description	terrain/difficulty info	short text	abc	hilly	No	No	No
course_kilometers	course length in kilometers, human inferences sprint vs distance race type	integer	0000	10	No	No	No

‘race Table Data Dictionary

Attribute	Description	Data type	Data Format	Sample Value	PK/CPK	FK?	Derived attribute?
race_id	# assigned to identify team	integer	0000	1	PK	No	No
course_id	identifier of course	integer	0000	1	No	FK	No
race_date	when it happened	short text	yyyy-mm-dd	2023-11-24	No	No	No
race_league	race type/class	short text	abc	World Cup	No	No	No

‘team’ Table Data Dictionary

Attribute	Description	Data type	Data Format	Sample Value	PK/CPK	FK?	Derived attribute?
team_id	# assigned to identify team	integer	0000	1	PK	No	No
team_name	country of team	short text	abc	Germany	No	No	No
coach_fname	name	short text	abc	Bjorn	No	No	No
coach_lname	name	short text	abc	Daehlie	No	No	No

‘athlete' Table Data Dictionary

Attribute	Description	Data type	Data Format	Sample Value	PK/CPK?	FK?	Derived attribute?
athlete_id	# assigned to identify athlete	integer	0000	1	PK	No	No
athlete_fname	Name	short text	abc	John	No	No	No
athlete_lname	Name	short text	abc	Smith	No	No	No
team_id	team to which athlete belongs	integer	0000	1	No	Yes	No

‘results1’ Table Data Dictionary

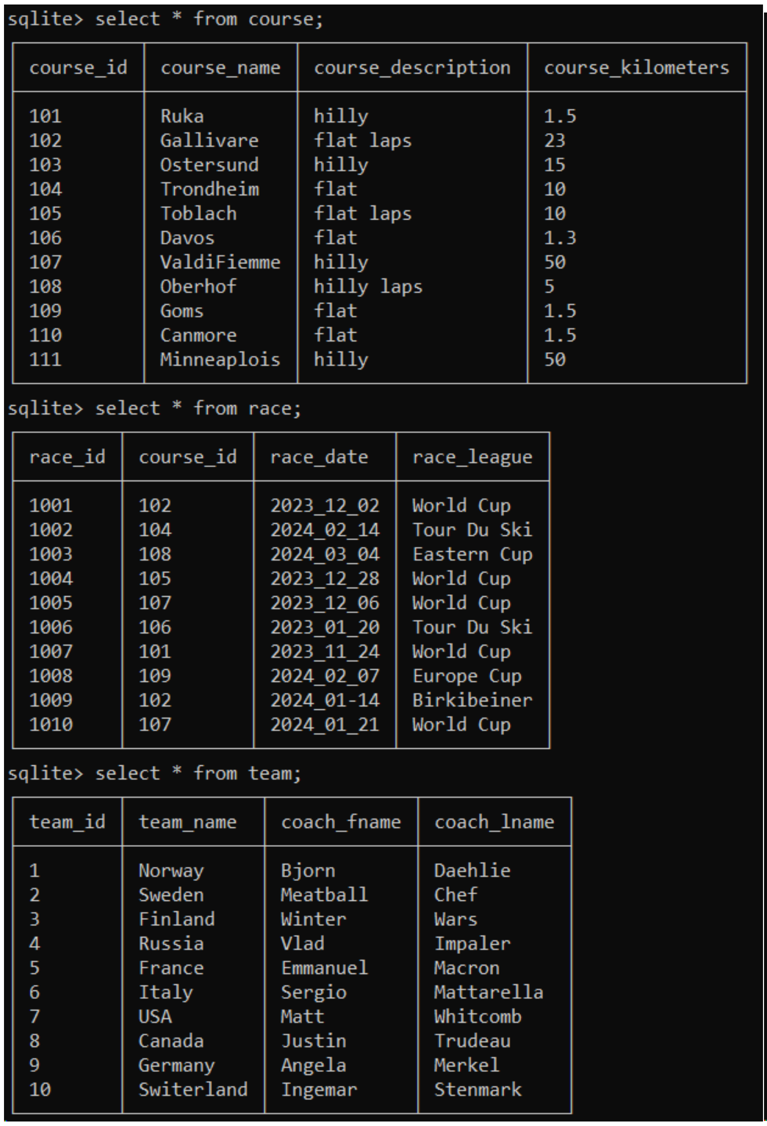
Attribute	Description	Data type	Data Format	Sample Value	PK/CPK?	FK?	Derived attribute?
participant_id	arbitrary	integer	0000	1	Yes	No	NO
category	men’s or women’s for scoring	text	abc	mens	No	No	No
time	decimal time taken to complete race or Disqualified, Did not finish, did not start	text	abc	DNF	No	No	No
athlete_id	# assigned to identify athlete	integer	0000	1	No	Yes	No
race_id	connects to race records table	integer	0000	1	No	Yes	No

‘results2’ Table Data Dictionary

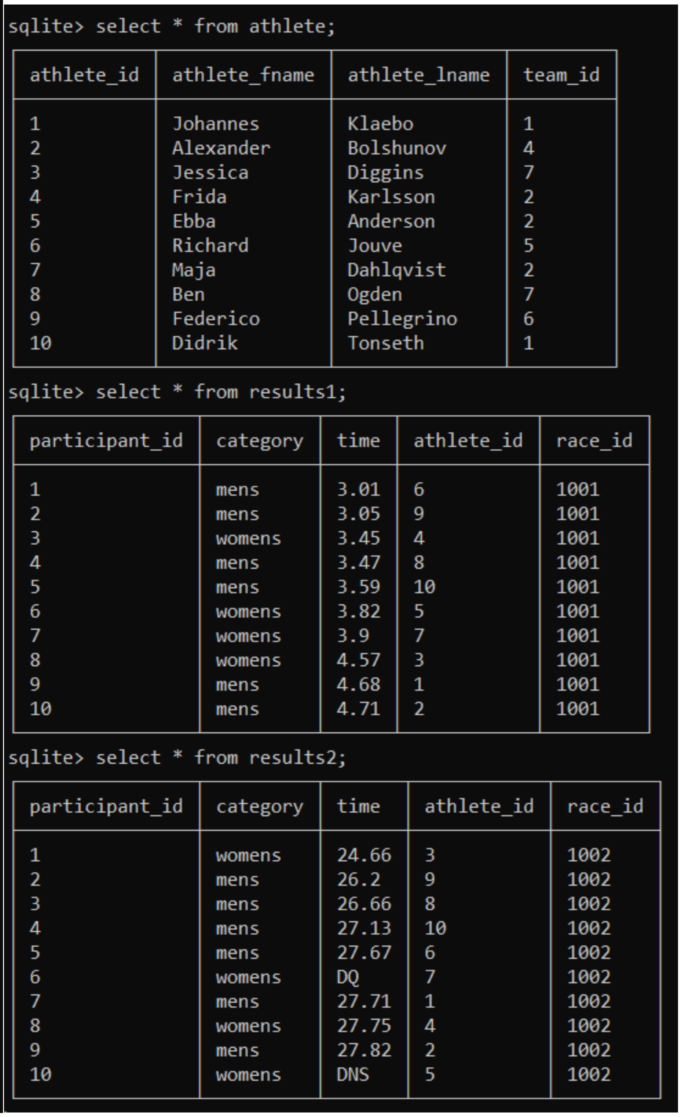
Attribute	Description	Data type	Data Format	Sample Value	PK/CPK?	FK?	Derived attribute?
participant_id	arbitrary	integer	0000	1	Yes	No	NO
category	men’s or women’s for scoring	text	abc	mens	No	No	No
time	decimal time taken to complete race or Disqualified, Did not finish, did not start	text	abc	DNF	No	No	No
athlete_id	# assigned to identify athlete	integer	0000	1	No	Yes	No
race_id	connects to race records table	integer	0000	1	No	Yes	No

Database Table Records

Here are simply the data points in each table. When this website is used, this information is entered either when users register their team, athletes, or course info, schedule a new race, or upload the results gathered at the timing of the race.



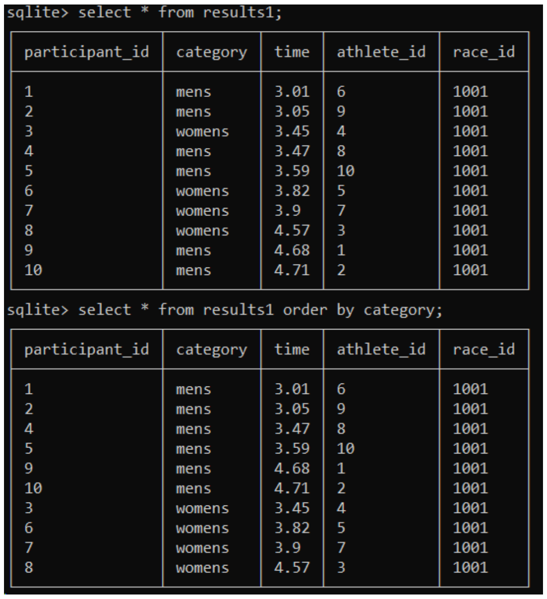
```
sqlite> select * from course;
course_id  course_name  course_description  course_kilometers

101        Ruka         hilly               1.5
102        Gallivare    flat laps           23
103        Ostersund    hilly               15
104        Trondheim    flat                10
105        Toblach      flat laps           10
106        Davos        flat                1.3
107        ValdiFiemme  hilly               50
108        Oberhof      hilly laps          5
109        Goms         flat                1.5
110        Canmore      flat                1.5
111        Minneaplois  hilly               50

sqlite> select * from race;
race_id  course_id  race_date   race_league

1001     102        2023_12_02  World Cup
1002     104        2024_02_14  Tour Du Ski
1003     108        2024_03_04  Eastern Cup
1004     105        2023_12_28  World Cup
1005     107        2023_12_06  World Cup
1006     106        2023_01_20  Tour Du Ski
1007     101        2023_11_24  World Cup
1008     109        2024_02_07  Europe Cup
1009     102        2024_01-14  Birkibeiner
1010     107        2024_01_21  World Cup

sqlite> select * from team;
team_id  team_name   coach_fname  coach_lname

1        Norway      Bjorn        Daehlie
2        Sweden      Meatball     Chef
3        Finland     Winter       Wars
4        Russia      Vlad         Impaler
5        France      Emmanuel     Macron
6        Italy       Sergio       Mattarella
7        USA         Matt         Whitcomb
8        Canada      Justin       Trudeau
9        Germany     Angela       Merkel
10       Switerland  Ingemar      Stenmark
```



```
sqlite> select * from athlete;
athlete_id  athlete_fname  athlete_lname  team_id

1           Johannes       Klaebo         1
2           Alexander      Bolshunov      4
3           Jessica        Diggins        7
4           Frida          Karlsson       2
5           Ebba           Anderson       2
6           Richard        Jouve          5
7           Maja           Dahlqvist      2
8           Ben            Ogden          7
9           Federico       Pellegrino     6
10          Didrik         Tonseth        1

sqlite> select * from results1;
participant_id  category  time  athlete_id  race_id

1               mens      3.01  6           1001
2               mens      3.05  9           1001
3               womens    3.45  4           1001
4               mens      3.47  8           1001
5               mens      3.59  10          1001
6               womens    3.82  5           1001
7               womens    3.9   7           1001
8               womens    4.57  3           1001
9               mens      4.68  1           1001
10              mens      4.71  2           1001

sqlite> select * from results2;
participant_id  category  time   athlete_id  race_id

1               womens    24.66  3           1002
2               mens      26.2   9           1002
3               mens      26.66  8           1002
4               mens      27.13  10          1002
5               mens      27.67  6           1002
6               womens    DQ     7           1002
7               mens      27.71  1           1002
8               womens    27.75  4           1002
9               mens      27.82  2           1002
10              womens    DNS    5           1002
```

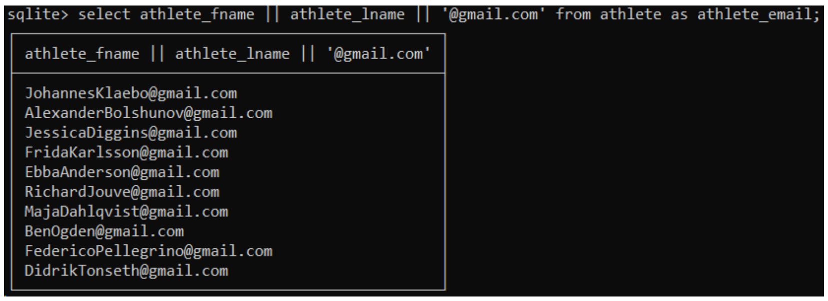
Examples of SQL Queries Run on the Database

As is the point of a database, the data can be queried in all sorts of ways to provide new ways of viewing information. In this lab, the queries and data management are conducted via a local machine running SQLite3 in a command line. Below are a series of examples of data manipulation on the database conducted in SQLite.



```
sqlite> select * from results1;
┌────────────────┬──────────┬──────┬────────────┬─────────┐
│ participant_id │ category │ time │ athlete_id │ race_id │
├────────────────┼──────────┼──────┼────────────┼─────────┤
│ 1              │ mens     │ 3.01 │ 6          │ 1001    │
│ 2              │ mens     │ 3.05 │ 9          │ 1001    │
│ 3              │ womens   │ 3.45 │ 4          │ 1001    │
│ 4              │ mens     │ 3.47 │ 8          │ 1001    │
│ 5              │ mens     │ 3.59 │ 10         │ 1001    │
│ 6              │ womens   │ 3.82 │ 5          │ 1001    │
│ 7              │ womens   │ 3.9  │ 7          │ 1001    │
│ 8              │ womens   │ 4.57 │ 3          │ 1001    │
│ 9              │ mens     │ 4.68 │ 1          │ 1001    │
│ 10             │ mens     │ 4.71 │ 2          │ 1001    │
└────────────────┴──────────┴──────┴────────────┴─────────┘
sqlite> select * from results1 order by category;
┌────────────────┬──────────┬──────┬────────────┬─────────┐
│ participant_id │ category │ time │ athlete_id │ race_id │
├────────────────┼──────────┼──────┼────────────┼─────────┤
│ 1              │ mens     │ 3.01 │ 6          │ 1001    │
│ 2              │ mens     │ 3.05 │ 9          │ 1001    │
│ 4              │ mens     │ 3.47 │ 8          │ 1001    │
│ 5              │ mens     │ 3.59 │ 10         │ 1001    │
│ 9              │ mens     │ 4.68 │ 1          │ 1001    │
│ 10             │ mens     │ 4.71 │ 2          │ 1001    │
│ 3              │ womens   │ 3.45 │ 4          │ 1001    │
│ 6              │ womens   │ 3.82 │ 5          │ 1001    │
│ 7              │ womens   │ 3.9  │ 7          │ 1001    │
│ 8              │ womens   │ 4.57 │ 3          │ 1001    │
└────────────────┴──────────┴──────┴────────────┴─────────┘
```

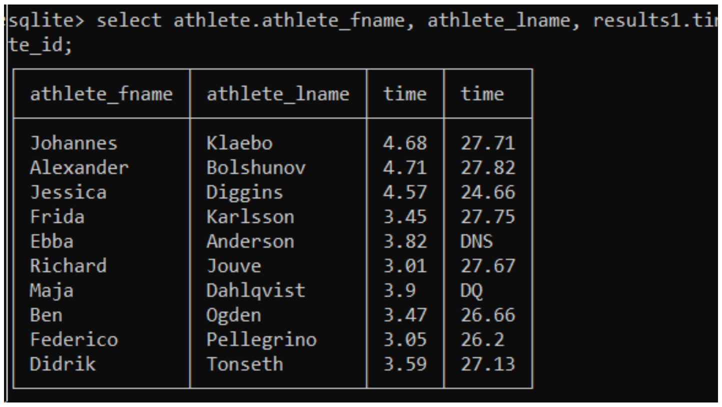
A. 1 sorting query



```
sqlite> select athlete_fname || athlete_lname || '@gmail.com' from athlete as athlete_email;
┌───────────────────────────────────────────┐
│ athlete_fname || athlete_lname || '@gmail.com' │
├───────────────────────────────────────────┤
│ JohannesKlaebo@gmail.com                  │
│ AlexanderBolshunov@gmail.com              │
│ JessicaDiggins@gmail.com                  │
│ FridaKarlsson@gmail.com                   │
│ EbbaAnderson@gmail.com                    │
│ RichardJouve@gmail.com                    │
│ MajaDahlqvist@gmail.com                   │
│ BenOgden@gmail.com                        │
│ FedericoPellegrino@gmail.com              │
│ DidrikTonseth@gmail.com                   │
└───────────────────────────────────────────┘
```

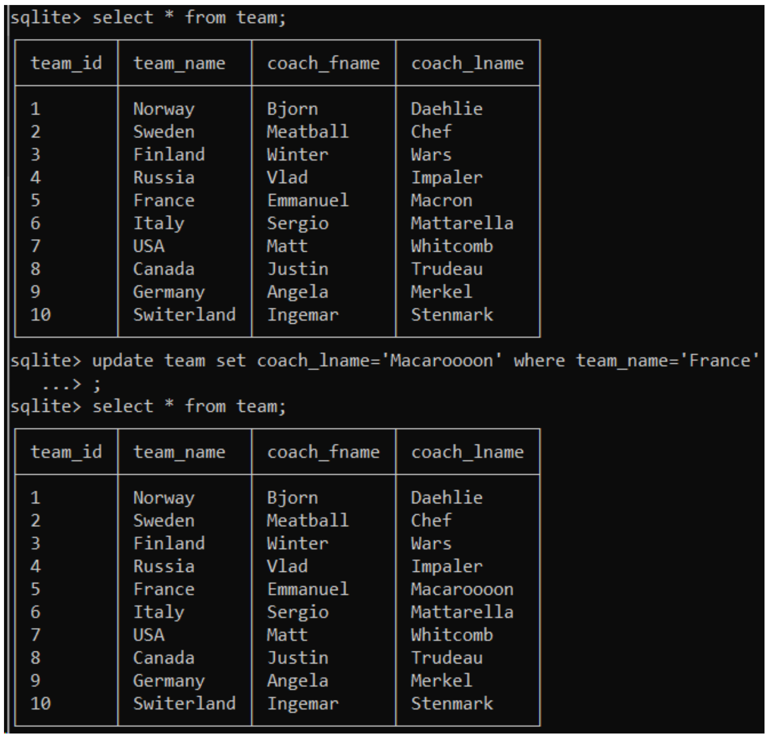
B. 1 formatting and concatenation query

```
select athlete.athlete_fname, athlete_lname, results1.time, results2.time from athlete left join
results1 on athlete.athlete_id=results1.athlete_id left join results2 on athlete.athlete_id =
results2.athlete_id;
```



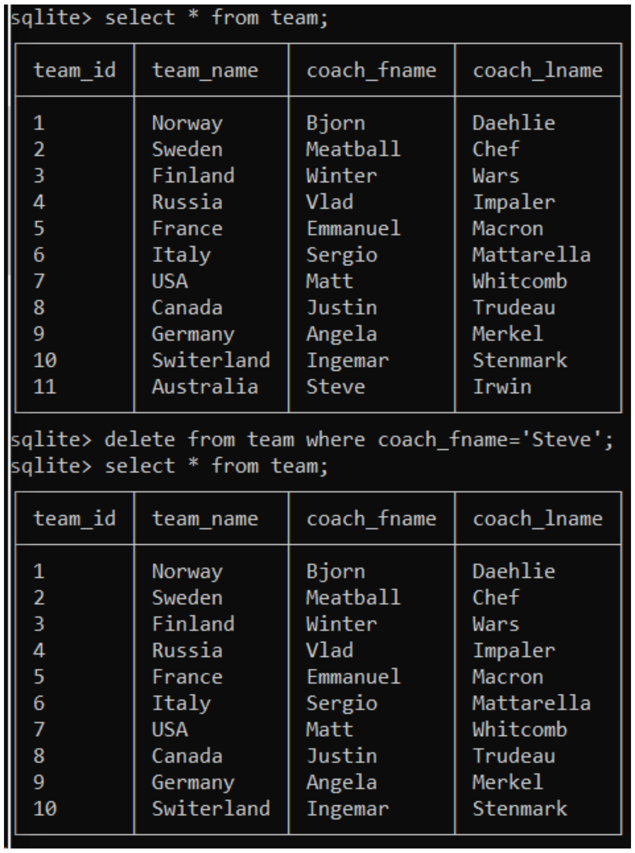
```
sqlite> select athlete.athlete_fname, athlete_lname, results1.ti
te_id;
┌───────────────┬───────────────┬──────┬───────┐
│ athlete_fname │ athlete_lname │ time │ time  │
├───────────────┼───────────────┼──────┼───────┤
│ Johannes      │ Klaebo        │ 4.68 │ 27.71 │
│ Alexander     │ Bolshunov     │ 4.71 │ 27.82 │
│ Jessica       │ Diggins       │ 4.57 │ 24.66 │
│ Frida         │ Karlsson      │ 3.45 │ 27.75 │
│ Ebba          │ Anderson      │ 3.82 │ DNS   │
│ Richard       │ Jouve         │ 3.01 │ 27.67 │
│ Maja          │ Dahlqvist     │ 3.9  │ DQ    │
│ Ben           │ Ogden         │ 3.47 │ 26.66 │
│ Federico      │ Pellegrino    │ 3.05 │ 26.2  │
│ Didrik        │ Tonseth       │ 3.59 │ 27.13 │
└───────────────┴───────────────┴──────┴───────┘
```

C. 1 left outer join query **D. 1 right outer join query- skipped**

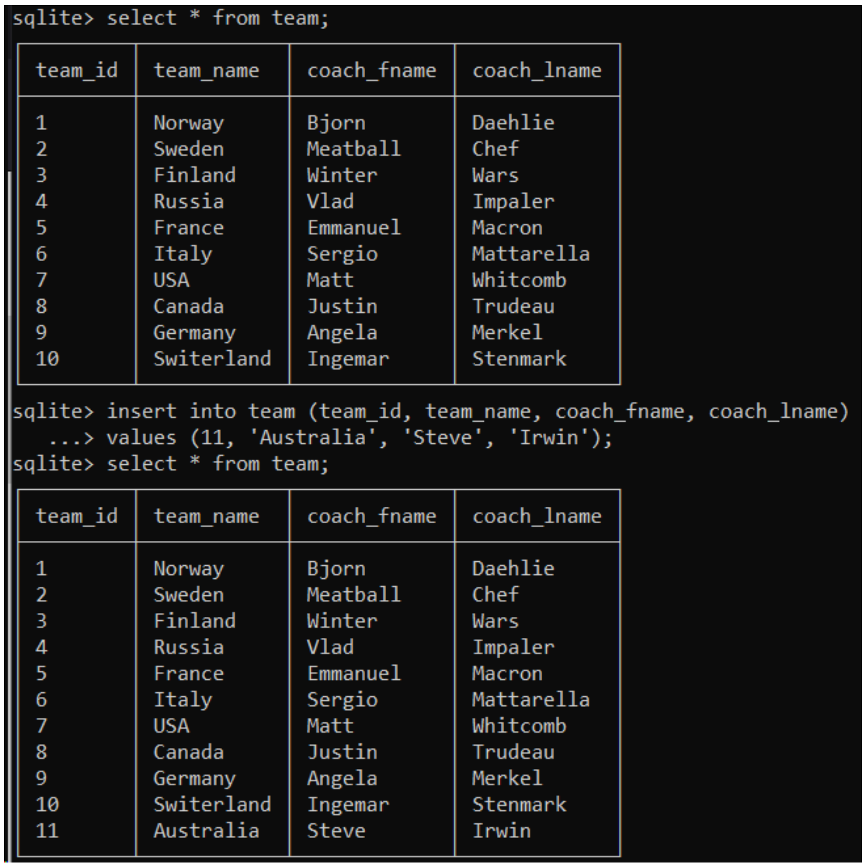


```
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macron      │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
└─────────┴───────────┴─────────────┴─────────────┘
sqlite> update team set coach_lname='Macaroooon' where team_name='France'
   ...> ;
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macaroooon  │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
└─────────┴───────────┴─────────────┴─────────────┘
```

E. 1 update query



```
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macron      │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
│ 11      │ Australia │ Steve       │ Irwin       │
└─────────┴───────────┴─────────────┴─────────────┘
sqlite> delete from team where coach_fname='Steve';
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macron      │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
└─────────┴───────────┴─────────────┴─────────────┘
```



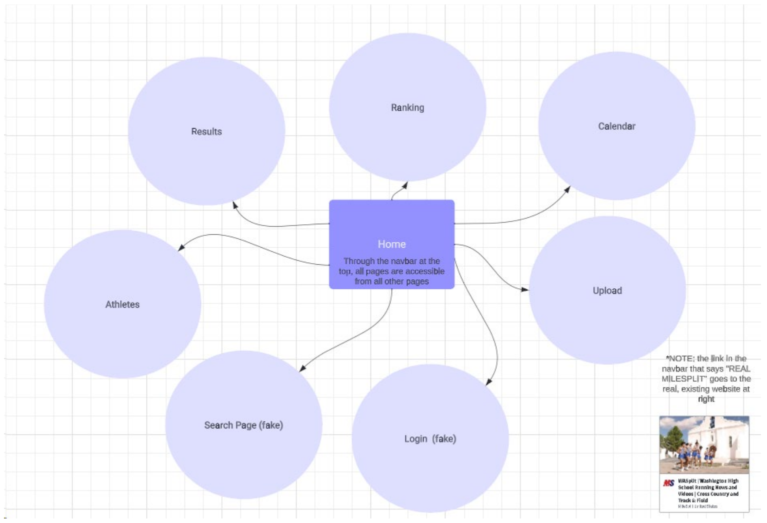
```
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macron      │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
└─────────┴───────────┴─────────────┴─────────────┘
sqlite> insert into team (team_id, team_name, coach_fname, coach_lname)
   ...> values (11, 'Australia', 'Steve', 'Irwin');
sqlite> select * from team;
┌─────────┬───────────┬─────────────┬─────────────┐
│ team_id │ team_name │ coach_fname │ coach_lname │
├─────────┼───────────┼─────────────┼─────────────┤
│ 1       │ Norway    │ Bjorn       │ Daehlie     │
│ 2       │ Sweden    │ Meatball    │ Chef        │
│ 3       │ Finland   │ Winter      │ Wars        │
│ 4       │ Russia    │ Vlad        │ Impaler     │
│ 5       │ France    │ Emmanuel    │ Macron      │
│ 6       │ Italy     │ Sergio      │ Mattarella  │
│ 7       │ USA       │ Matt        │ Whitcomb    │
│ 8       │ Canada    │ Justin      │ Trudeau     │
│ 9       │ Germany   │ Angela      │ Merkel      │
│ 10      │ Switerland│ Ingemar     │ Stenmark    │
│ 11      │ Australia │ Steve       │ Irwin       │
└─────────┴───────────┴─────────────┴─────────────┘
```

G. 1 delete query (left) **F. 1 insert query (right)**

Front End Design

After creating and practicing manipulating the data in the database, it can be connected to a series of simple HTML webpages via python accessing a terminal and running SQLite3 on a local machine. However, this means that in order to work, the HTML sites must be accessed through local ports only. While a live version of the HTML pages is available as a statically hosted GitHub Pages deployment here, they are inoperable as they are not connected to a python script, a command line, SQLite3, or the database.

The website designed to access this data is modeled as a copy of the real website MileSplit. As such, the pages are set up as in the below site map, with the home page being the initial landing page. All pages have a navigation bar at the top that allows them to access all other pages though.


Results
Ranking
Calendar
Home
Through the navbar at the top, all pages are accessible from all other pages
Athletes
Upload
Search Page (fake)
Login (fake)
*NOTE: the link in the navbar that says "REAL MILESPLIT" goes to the real, existing website at right
MS
WASplit | Washington High School Running News and Videos | Cross Country and Track & Field

Site Map

As in the Site skeleton below, most pages serve to access data in a unique way through python.

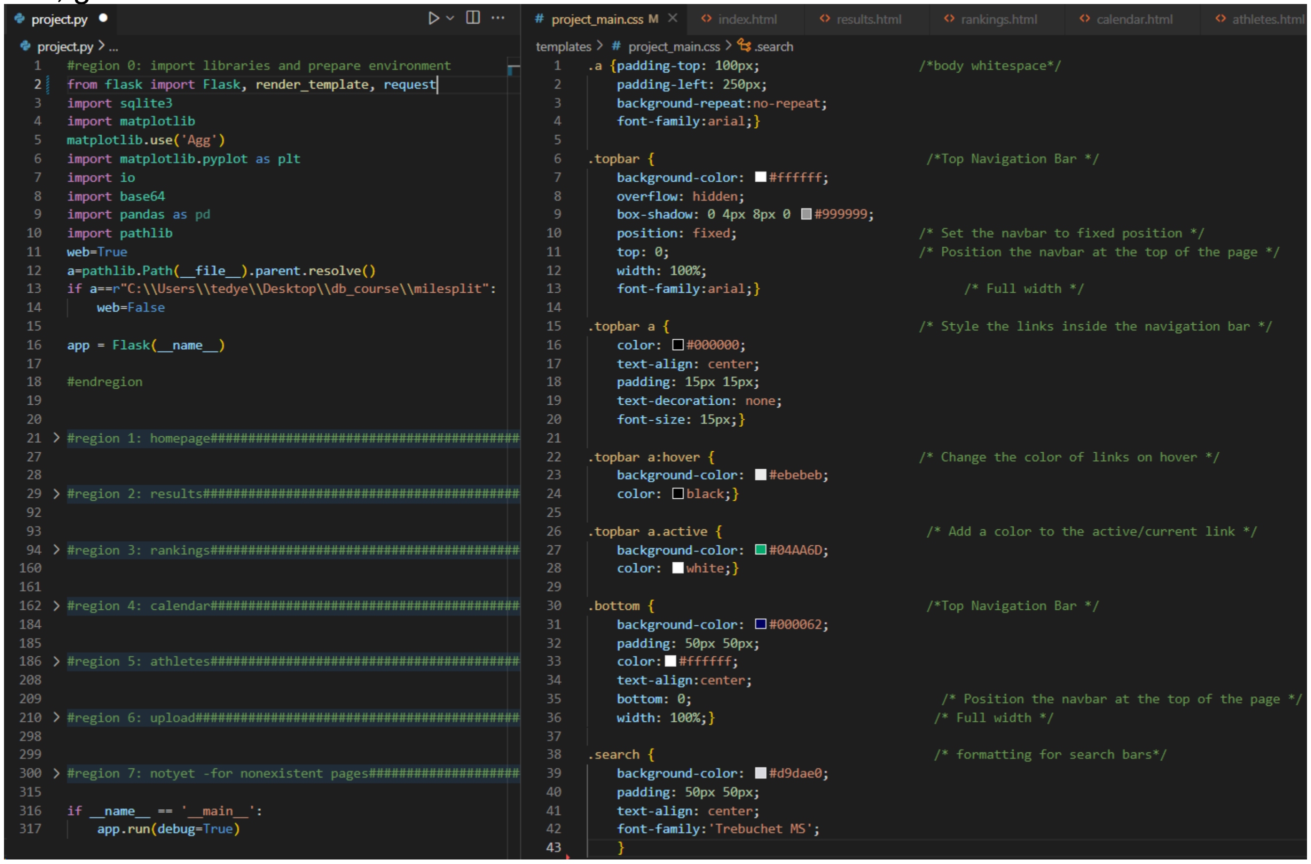
Site Skeleton

- Home page
    - Static images
- Results
    - **Report** of data from specified race result table
      Plot (bar graph) of race distances
- Rankings
    - **Extracts** athlete total place
    - **Plot** of times
- Calendar
    - **Extracts** all records from the race table (i.e., shows all listed races)
- Athletes
    - **Report** of athlete's times from all available race results
- Upload
    - **Adds** new records to race results
    - **Adds** new athletes
- Search & Login
    - Static page saying that this page has not yet been built

Code

Once again, it is recommended that the code be viewed on GitHub here:
https://github.com/tedyee114/milesplit.git
And that the live (but static) webpage be viewed here:
https://tedyee114.github.io/milesplit/templates/index

Also, please note that in order to make the code run both locally and on GitHub some syntax had to change, such as {{URLfor'page'}} just becoming 'page'. There is a lot of redundant code that depends on whether or not the HTML site is being viewed locally or on GitHub, necessary to correct filepaths to static image content. To avoid redundancies caused by that and by the repeated headers and footers, they will only be shown once and will be collapsed in the screenshots for brevity. To see the full code, go to the GitHub above.


project.py

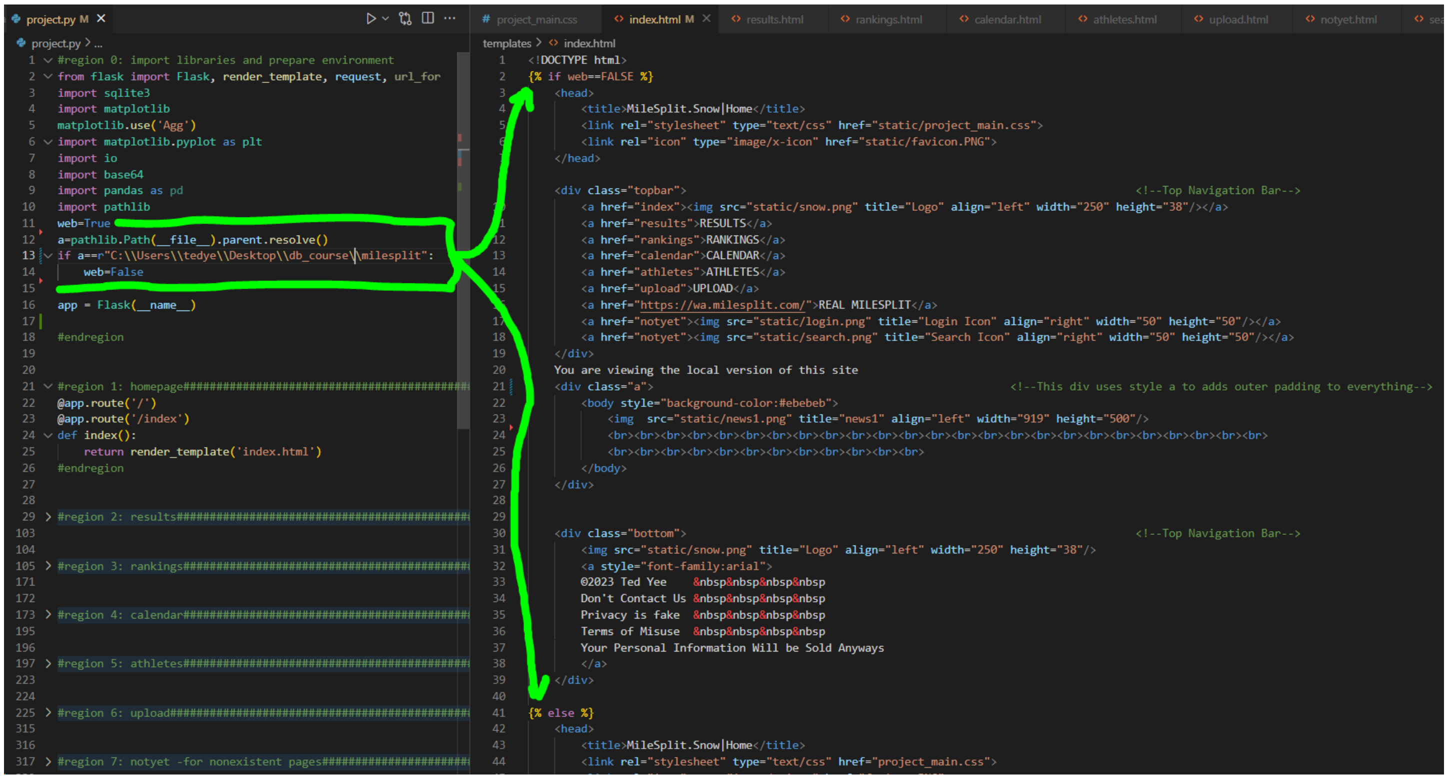
```python
#region 0: import libraries and prepare environment
from flask import Flask, render_template, request
import sqlite3
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import io
import base64
import pandas as pd
import pathlib
web=True
a=pathlib.Path(__file__).parent.resolve()
if a==r"C:\\Users\\tedye\\Desktop\\db_course\\milesplit":
    web=False

app = Flask(__name__)

#endregion


> #region 1: homepage##########################################


> #region 2: results###########################################


> #region 3: rankings##########################################


> #region 4: calendar##########################################


> #region 5: athletes##########################################


> #region 6: upload############################################


> #region 7: notyet -for nonexistent pages####################

if __name__ == '__main__':
    app.run(debug=True)
```

templates > project_main.css > .search

```css
.a {padding-top: 100px;                       /*body whitespace*/
    padding-left: 250px;
    background-repeat:no-repeat;
    font-family:arial;}

.topbar {                                      /*Top Navigation Bar */
    background-color: #ffffff;
    overflow: hidden;
    box-shadow: 0 4px 8px 0 #999999;
    position: fixed;                          /* Set the navbar to fixed position */
    top: 0;                                   /* Position the navbar at the top of the page */
    width: 100%;
    font-family:arial;}                             /* Full width */

.topbar a {                                   /* Style the links inside the navigation bar */
    color: #000000;
    text-align: center;
    padding: 15px 15px;
    text-decoration: none;
    font-size: 15px;}

.topbar a:hover {                             /* Change the color of links on hover */
    background-color: #ebebeb;
    color: black;}

.topbar a.active {                             /* Add a color to the active/current link */
    background-color: #04AA6D;
    color: white;}

.bottom {                                      /*Top Navigation Bar */
    background-color: #000062;
    padding: 50px 50px;
    color: #ffffff;
    text-align:center;
    bottom: 0;                                  /* Position the navbar at the top of the page */
    width: 100%;}                              /* Full width */

.search {                                      /* formatting for search bars*/
    background-color: #d9dae0;
    padding: 50px 50px;
    text-align: center;
    font-family:'Trebuchet MS';
    }
```

Overview of Python Code and CSS File

Homepage Python Script, HTML, rendered webpage



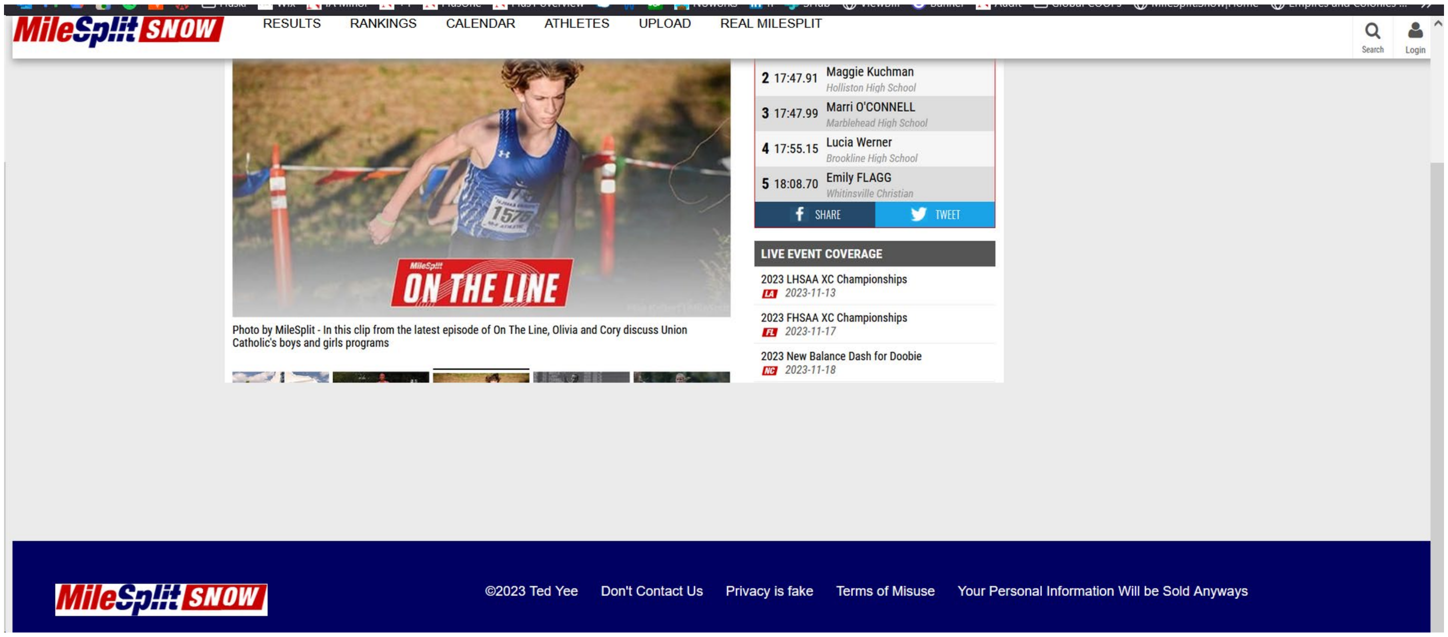
project.py

```python
#region 0: import libraries and prepare environment
from flask import Flask, render_template, request, url_for
import sqlite3
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import io
import base64
import pandas as pd
import pathlib
web=True
a=pathlib.Path(__file__).parent.resolve()
if a==r"C:\\Users\\tedye\\Desktop\\db_course\\milesplit":
    web=False

app = Flask(__name__)

#endregion


#region 1: homepage##########################################
@app.route('/')
@app.route('/index')
def index():
    return render_template('index.html')
#endregion


#region 2: results##########################################


#region 3: rankings##########################################


#region 4: calendar##########################################


#region 5: athletes##########################################


#region 6: upload##########################################


#region 7: notyet -for nonexistent pages######################
```

templates > index.html

```html
<!DOCTYPE html>
{% if web==FALSE %}
    <head>
        <title>MileSplit.Snow|Home</title>
        <link rel="stylesheet" type="text/css" href="static/project_main.css">
        <link rel="icon" type="image/x-icon" href="static/favicon.PNG">
    </head>

    <div class="topbar">                                                          <!--Top Navigation Bar-->
        <a href="index"><img src="static/snow.png" title="Logo" align="left" width="250" height="38"/></a>
        <a href="results">RESULTS</a>
        <a href="rankings">RANKINGS</a>
        <a href="calendar">CALENDAR</a>
        <a href="athletes">ATHLETES</a>
        <a href="upload">UPLOAD</a>
        <a href="https://wa.milesplit.com/">REAL MILESPLIT</a>
        <a href="notyet"><img src="static/login.png" title="Login Icon" align="right" width="50" height="50"/></a>
        <a href="notyet"><img src="static/search.png" title="Search Icon" align="right" width="50" height="50"/></a>
    </div>
    You are viewing the local version of this site
    <div class="a">                                           <!--This div uses style a to adds outer padding to everything-->
        <body style="background-color:#ebebeb">
            <img  src="static/news1.png" title="news1" align="left" width="919" height="500"/>
            <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
            <br><br><br><br><br><br><br><br><br><br><br><br>
        </body>
    </div>


    <div class="bottom">                                                          <!--Top Navigation Bar-->
        <img src="static/snow.png" title="Logo" align="left" width="250" height="38"/>
        <a style="font-family:arial">
        ©2023 Ted Yee    &nbsp&nbsp&nbsp&nbsp
        Don't Contact Us &nbsp&nbsp&nbsp&nbsp
        Privacy is fake  &nbsp&nbsp&nbsp&nbsp
        Terms of Misuse  &nbsp&nbsp&nbsp&nbsp
        Your Personal Information Will be Sold Anyways
        </a>
    </div>

{% else %}
    <head>
        <title>MileSplit.Snow|Home</title>
        <link rel="stylesheet" type="text/css" href="project_main.css">
```

Note here again that there is a clause detecting whether the site is running on the web or locally. The whole code is repeated in the 'else' statement, with only altered filepaths to the static images.



2 17:47.91 Maggie Kuchman
Holliston High School

3 17:47.99 Marri O'CONNELL
Marblehead High School

4 17:55.15 Lucia Werner
Brookline High School

5 18:08.70 Emily FLAGG
Whitinsville Christian

Photo by MileSplit - In this clip from the latest episode of On The Line, Olivia and Cory discuss Union Catholic's boys and girls programs

LIVE EVENT COVERAGE

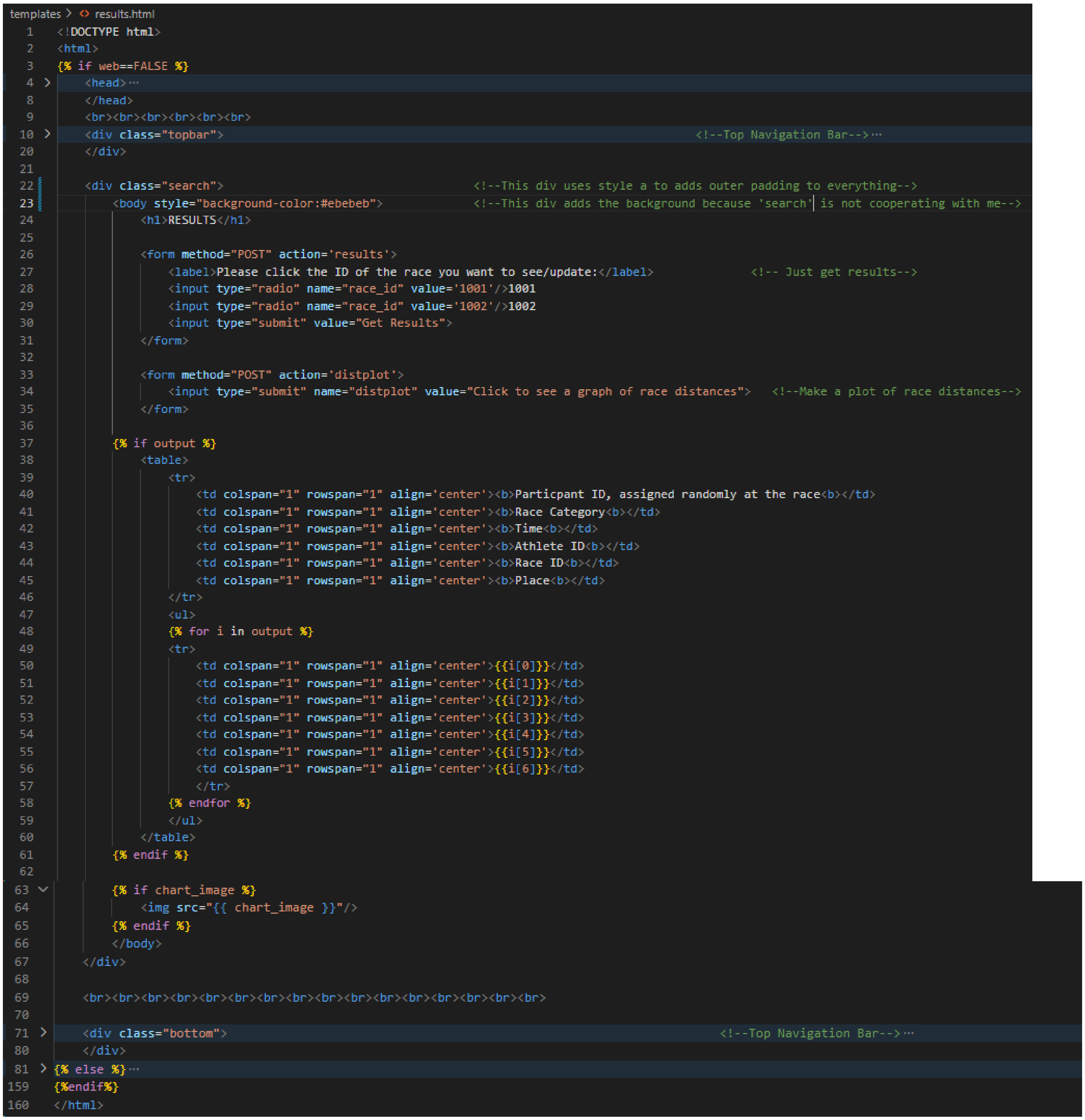
2023 LHSAA XC Championships
LA 2023-11-13

2023 FHSAA XC Championships
FL 2023-11-17

2023 New Balance Dash for Doobie
NC 2023-11-18

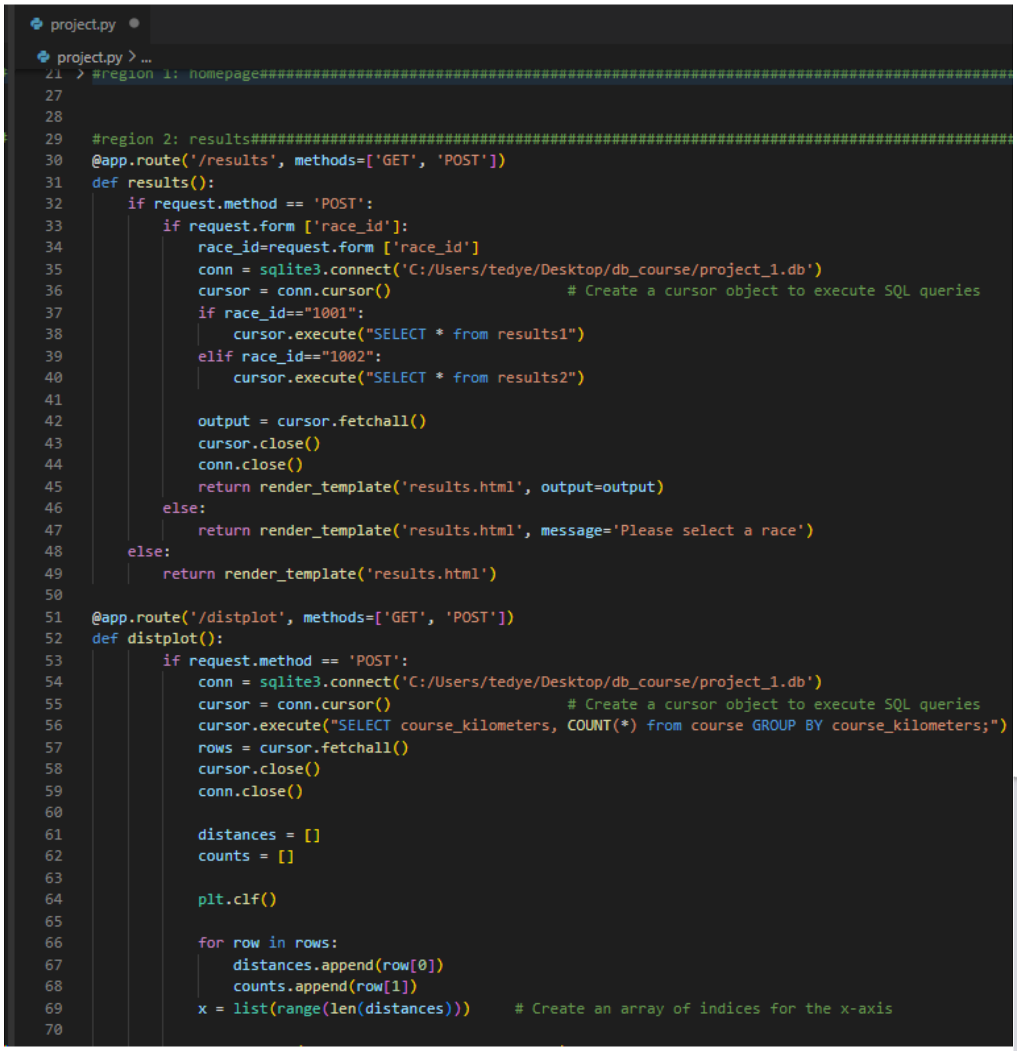
©2023 Ted Yee Don't Contact Us Privacy is fake Terms of Misuse Your Personal Information Will be Sold Anyways

Results Python Script, HTML, rendered webpage



templates > results.html

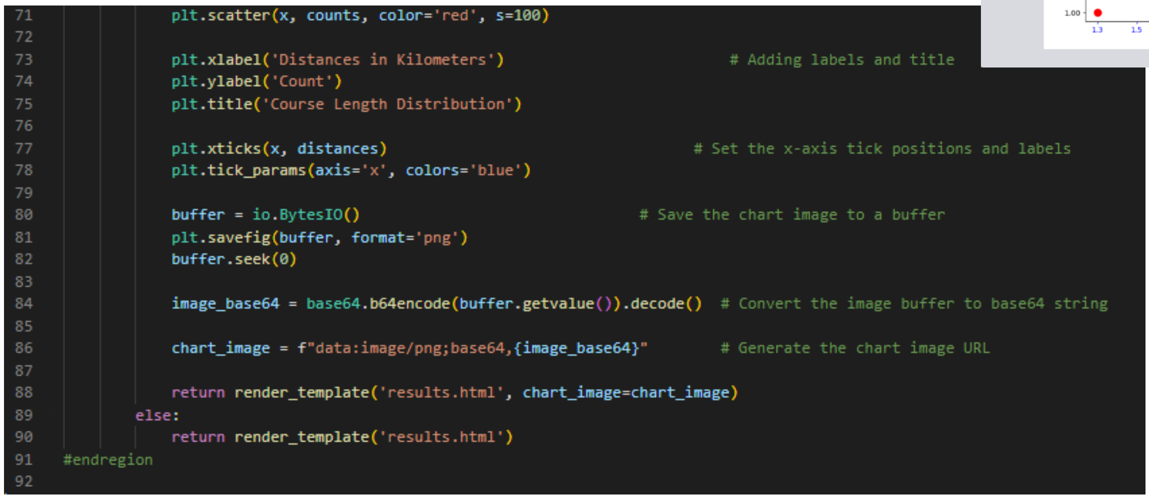
```html
<!DOCTYPE html>
<html>
{% if web==FALSE %}
    <head>⋯
    </head>
    <br><br><br><br><br><br>
    <div class="topbar">                                                    <!--Top Navigation Bar-->⋯
    </div>

    <div class="search">                                <!--This div uses style a to adds outer padding to everything-->
        <body style="background-color:#ebebeb">         <!--This div adds the background because 'search' is not cooperating with me-->
            <h1>RESULTS</h1>

            <form method="POST" action='results'>
                <label>Please click the ID of the race you want to see/update:</label>              <!-- Just get results-->
                <input type="radio" name="race_id" value='1001'/>1001
                <input type="radio" name="race_id" value='1002'/>1002
                <input type="submit" value="Get Results">
            </form>

            <form method="POST" action='distplot'>
                <input type="submit" name="distplot" value="Click to see a graph of race distances">   <!--Make a plot of race distances-->
            </form>

        {% if output %}
            <table>
                <tr>
                    <td colspan="1" rowspan="1" align='center'><b>Particpant ID, assigned randomly at the race<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race Category<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Time<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Athlete ID<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race ID<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Place<b></td>
                </tr>
                <ul>
                {% for i in output %}
                <tr>
                    <td colspan="1" rowspan="1" align='center'>{{i[0]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[1]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[2]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[3]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[4]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[5]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[6]}}</td>
                    </tr>
                {% endfor %}
                </ul>
            </table>
        {% endif %}

        {% if chart_image %}
            <img src="{{ chart_image }}"/>
        {% endif %}
        </body>
    </div>

    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

    <div class="bottom">                                                    <!--Top Navigation Bar-->⋯
    </div>
{% else %}⋯
{%endif%}
</html>
```



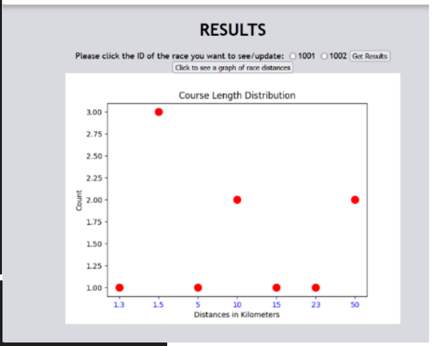
project.py

project.py > ...

```python
> #region 1: homepage################################################################################


#region 2: results###################################################################################
@app.route('/results', methods=['GET', 'POST'])
def results():
    if request.method == 'POST':
        if request.form ['race_id']:
            race_id=request.form ['race_id']
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
            if race_id=="1001":
                cursor.execute("SELECT * from results1")
            elif race_id=="1002":
                cursor.execute("SELECT * from results2")

            output = cursor.fetchall()
            cursor.close()
            conn.close()
            return render_template('results.html', output=output)
        else:
            return render_template('results.html', message='Please select a race')
    else:
        return render_template('results.html')

@app.route('/distplot', methods=['GET', 'POST'])
def distplot():
        if request.method == 'POST':
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
            cursor.execute("SELECT course_kilometers, COUNT(*) from course GROUP BY course_kilometers;")
            rows = cursor.fetchall()
            cursor.close()
            conn.close()

            distances = []
            counts = []

            plt.clf()

            for row in rows:
                distances.append(row[0])
                counts.append(row[1])
            x = list(range(len(distances)))      # Create an array of indices for the x-axis

```



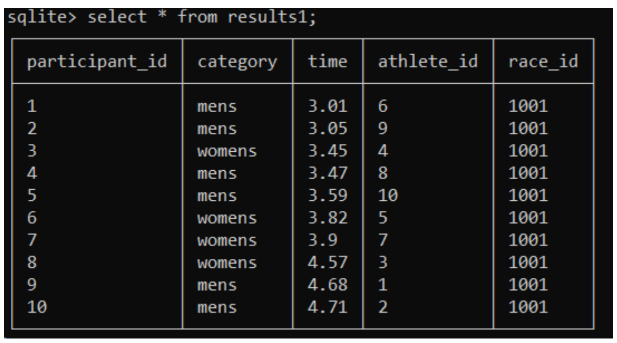
```python
            plt.scatter(x, counts, color='red', s=100)

            plt.xlabel('Distances in Kilometers')                        # Adding labels and title
            plt.ylabel('Count')
            plt.title('Course Length Distribution')

            plt.xticks(x, distances)                                # Set the x-axis tick positions and labels
            plt.tick_params(axis='x', colors='blue')

            buffer = io.BytesIO()                             # Save the chart image to a buffer
            plt.savefig(buffer, format='png')
            buffer.seek(0)

            image_base64 = base64.b64encode(buffer.getvalue()).decode()  # Convert the image buffer to base64 string

            chart_image = f"data:image/png;base64,{image_base64}"        # Generate the chart image URL

            return render_template('results.html', chart_image=chart_image)
        else:
            return render_template('results.html')
#endregion

```



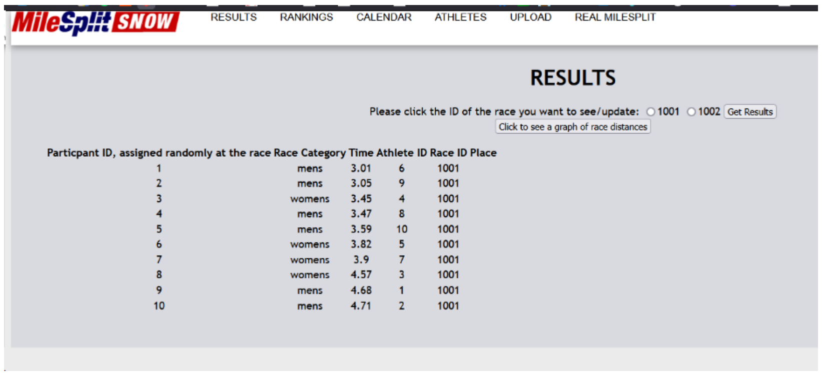
RESULTS

Please click the ID of the race you want to see/update: ○ 1001 ○ 1002 Get Results

Click to see a graph of race distances



```
sqlite> select * from results1;

 participant_id | category | time | athlete_id | race_id
 1              | mens     | 3.01 | 6          | 1001
 2              | mens     | 3.05 | 9          | 1001
 3              | womens   | 3.45 | 4          | 1001
 4              | mens     | 3.47 | 8          | 1001
 5              | mens     | 3.59 | 10         | 1001
 6              | womens   | 3.82 | 5          | 1001
 7              | womens   | 3.9  | 7          | 1001
 8              | womens   | 4.57 | 3          | 1001
 9              | mens     | 4.68 | 1          | 1001
 10             | mens     | 4.71 | 2          | 1001
```



RESULTS RANKINGS CALENDAR ATHLETES UPLOAD REAL MILESPLIT

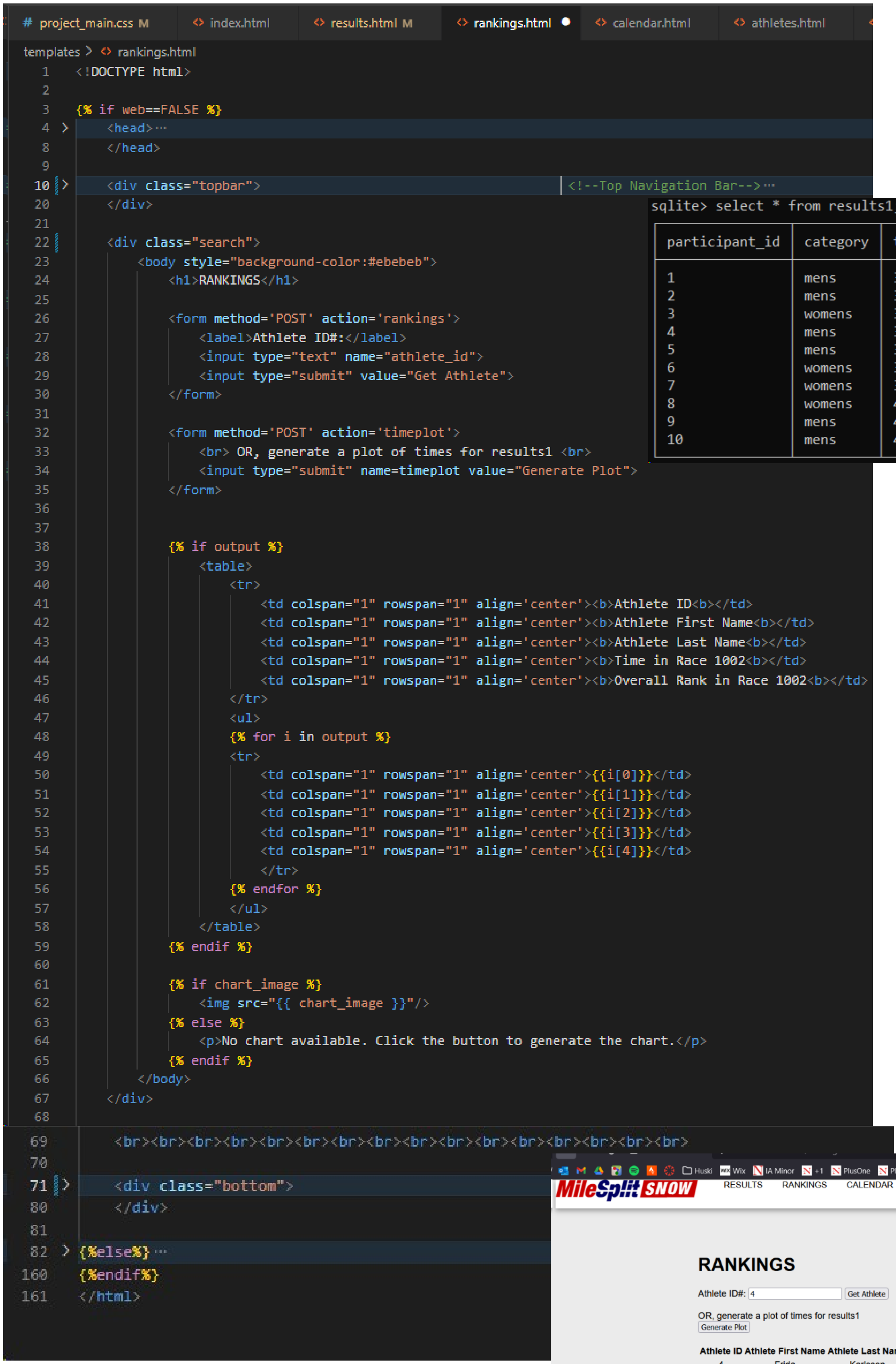
RESULTS

Please click the ID of the race you want to see/update: ○ 1001 ○ 1002 Get Results

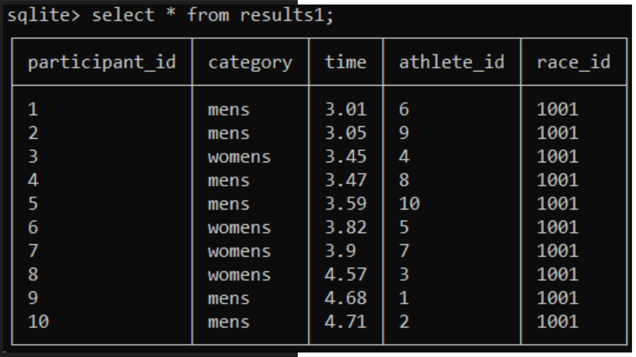
Click to see a graph of race distances

Particpant ID, assigned randomly at the race	Race Category	Time	Athlete ID	Race ID	Place
1	mens	3.01	6	1001	
2	mens	3.05	9	1001	
3	womens	3.45	4	1001	
4	mens	3.47	8	1001	
5	mens	3.59	10	1001	
6	womens	3.82	5	1001	
7	womens	3.9	7	1001	
8	womens	4.57	3	1001	
9	mens	4.68	1	1001	
10	mens	4.71	2	1001	

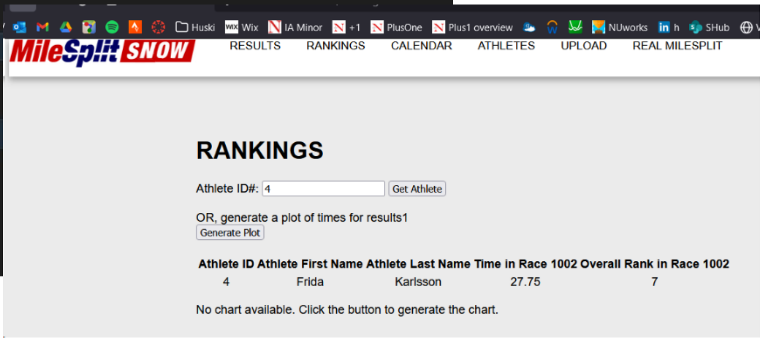
RANKINGS Python Script, HTML, rendered webpage



```html
<!DOCTYPE html>

{% if web==FALSE %}
    <head>···
    </head>

    <div class="topbar">                                        <!--Top Navigation Bar-->···
    </div>

    <div class="search">
        <body style="background-color:#ebebeb">
            <h1>RANKINGS</h1>

            <form method='POST' action='rankings'>
                <label>Athlete ID#:</label>
                <input type="text" name="athlete_id">
                <input type="submit" value="Get Athlete">
            </form>

            <form method='POST' action='timeplot'>
                <br> OR, generate a plot of times for results1 <br>
                <input type="submit" name=timeplot value="Generate Plot">
            </form>


            {% if output %}
                <table>
                    <tr>
                        <td colspan="1" rowspan="1" align='center'><b>Athlete ID<b></td>
                        <td colspan="1" rowspan="1" align='center'><b>Athlete First Name<b></td>
                        <td colspan="1" rowspan="1" align='center'><b>Athlete Last Name<b></td>
                        <td colspan="1" rowspan="1" align='center'><b>Time in Race 1002<b></td>
                        <td colspan="1" rowspan="1" align='center'><b>Overall Rank in Race 1002<b></td>
                    </tr>
                    <ul>
                    {% for i in output %}
                    <tr>
                        <td colspan="1" rowspan="1" align='center'>{{i[0]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[1]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[2]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[3]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[4]}}</td>
                        </tr>
                    {% endfor %}
                    </ul>
                </table>
            {% endif %}

            {% if chart_image %}
                <img src="{{ chart_image }}"/>
            {% else %}
                <p>No chart available. Click the button to generate the chart.</p>
            {% endif %}
        </body>
    </div>

    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

    <div class="bottom">
    </div>

{%else%}···
{%endif%}
</html>
```



```
sqlite> select * from results1;
┌────────────────┬──────────┬──────┬────────────┬─────────┐
│ participant_id │ category │ time │ athlete_id │ race_id │
├────────────────┼──────────┼──────┼────────────┼─────────┤
│ 1              │ mens     │ 3.01 │ 6          │ 1001    │
│ 2              │ mens     │ 3.05 │ 9          │ 1001    │
│ 3              │ womens   │ 3.45 │ 4          │ 1001    │
│ 4              │ mens     │ 3.47 │ 8          │ 1001    │
│ 5              │ mens     │ 3.59 │ 10         │ 1001    │
│ 6              │ womens   │ 3.82 │ 5          │ 1001    │
│ 7              │ womens   │ 3.9  │ 7          │ 1001    │
│ 8              │ womens   │ 4.57 │ 3          │ 1001    │
│ 9              │ mens     │ 4.68 │ 1          │ 1001    │
│ 10             │ mens     │ 4.71 │ 2          │ 1001    │
└────────────────┴──────────┴──────┴────────────┴─────────┘
```



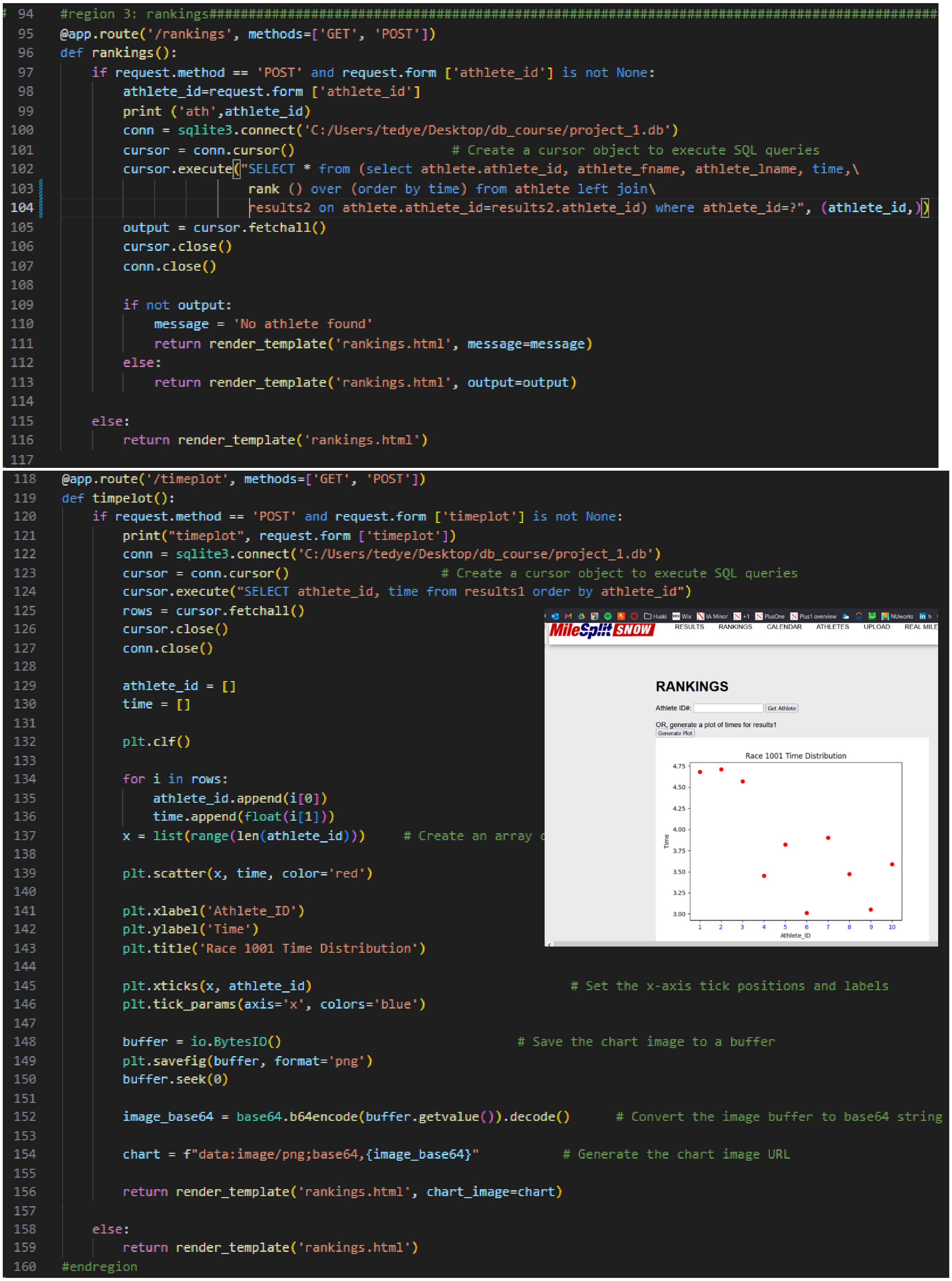
RANKINGS

Athlete ID#: 4

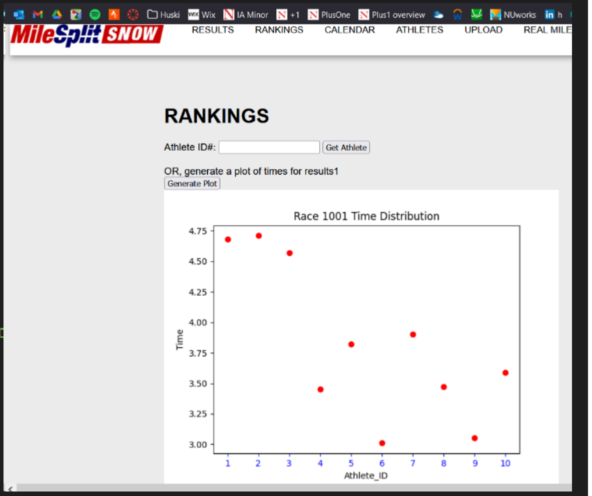
OR, generate a plot of times for results1

Athlete ID	Athlete First Name	Athlete Last Name	Time in Race 1002	Overall Rank in Race 1002
4	Frida	Karlsson	27.75	7

No chart available. Click the button to generate the chart.



```python
#region 3: rankings####################################################################################
@app.route('/rankings', methods=['GET', 'POST'])
def rankings():
    if request.method == 'POST' and request.form ['athlete_id'] is not None:
        athlete_id=request.form ['athlete_id']
        print ('ath',athlete_id)
        conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
        cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
        cursor.execute("SELECT * from (select athlete.athlete_id, athlete_fname, athlete_lname, time,\
                rank () over (order by time) from athlete left join\
                results2 on athlete.athlete_id=results2.athlete_id) where athlete_id=?", (athlete_id,))
        output = cursor.fetchall()
        cursor.close()
        conn.close()

        if not output:
            message = 'No athlete found'
            return render_template('rankings.html', message=message)
        else:
            return render_template('rankings.html', output=output)

    else:
        return render_template('rankings.html')

@app.route('/timeplot', methods=['GET', 'POST'])
def timeplot():
    if request.method == 'POST' and request.form ['timeplot'] is not None:
        print("timeplot", request.form ['timeplot'])
        conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
        cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
        cursor.execute("SELECT athlete_id, time from results1 order by athlete_id")
        rows = cursor.fetchall()
        cursor.close()
        conn.close()

        athlete_id = []
        time = []

        plt.clf()

        for i in rows:
            athlete_id.append(i[0])
            time.append(float(i[1]))
        x = list(range(len(athlete_id)))      # Create an array c

        plt.scatter(x, time, color='red')

        plt.xlabel('Athlete_ID')
        plt.ylabel('Time')
        plt.title('Race 1001 Time Distribution')

        plt.xticks(x, athlete_id)                           # Set the x-axis tick positions and labels
        plt.tick_params(axis='x', colors='blue')

        buffer = io.BytesIO()                         # Save the chart image to a buffer
        plt.savefig(buffer, format='png')
        buffer.seek(0)

        image_base64 = base64.b64encode(buffer.getvalue()).decode()      # Convert the image buffer to base64 string

        chart = f"data:image/png;base64,{image_base64}"           # Generate the chart image URL

        return render_template('rankings.html', chart_image=chart)

    else:
        return render_template('rankings.html')
#endregion
```

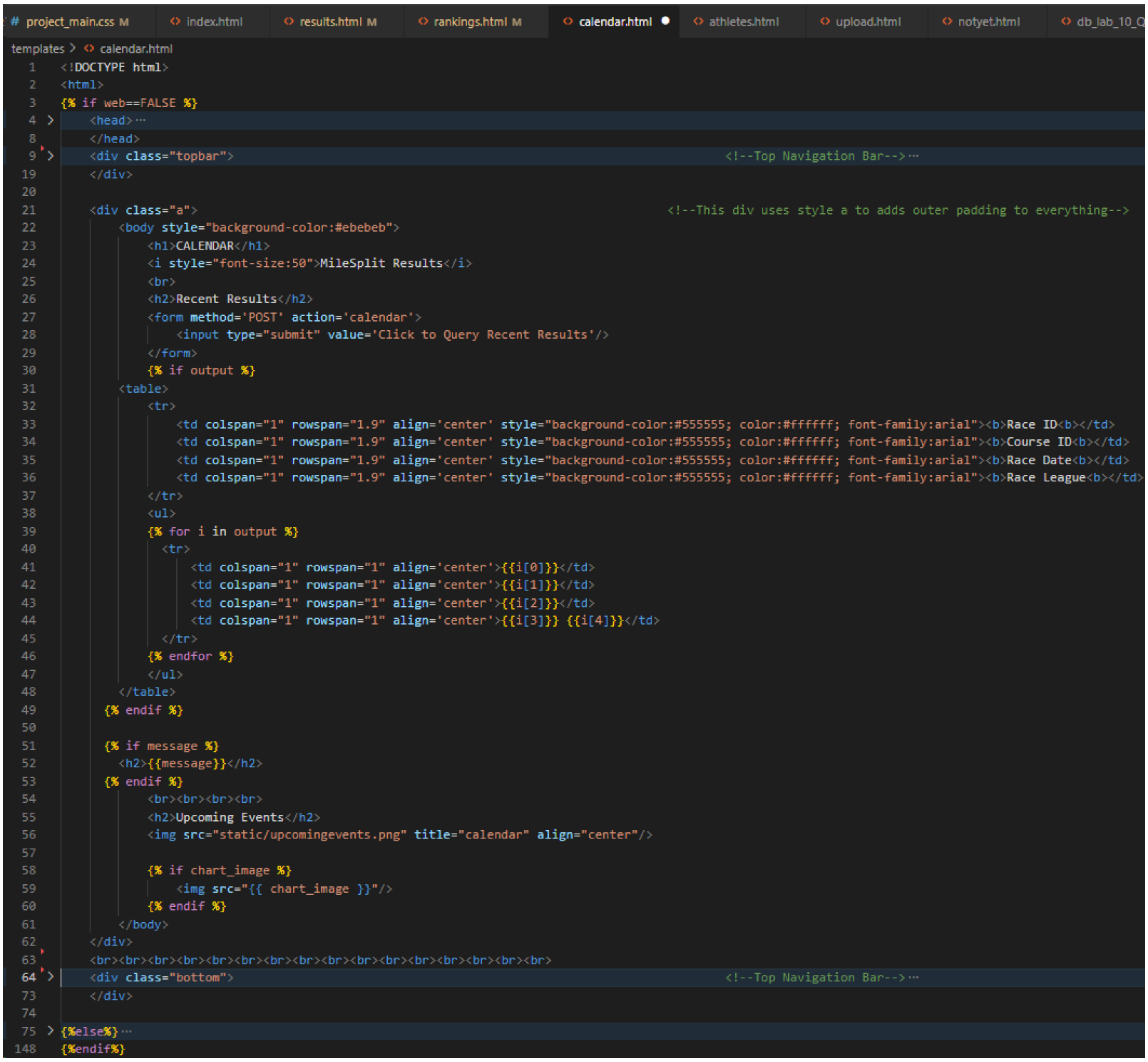


RANKINGS

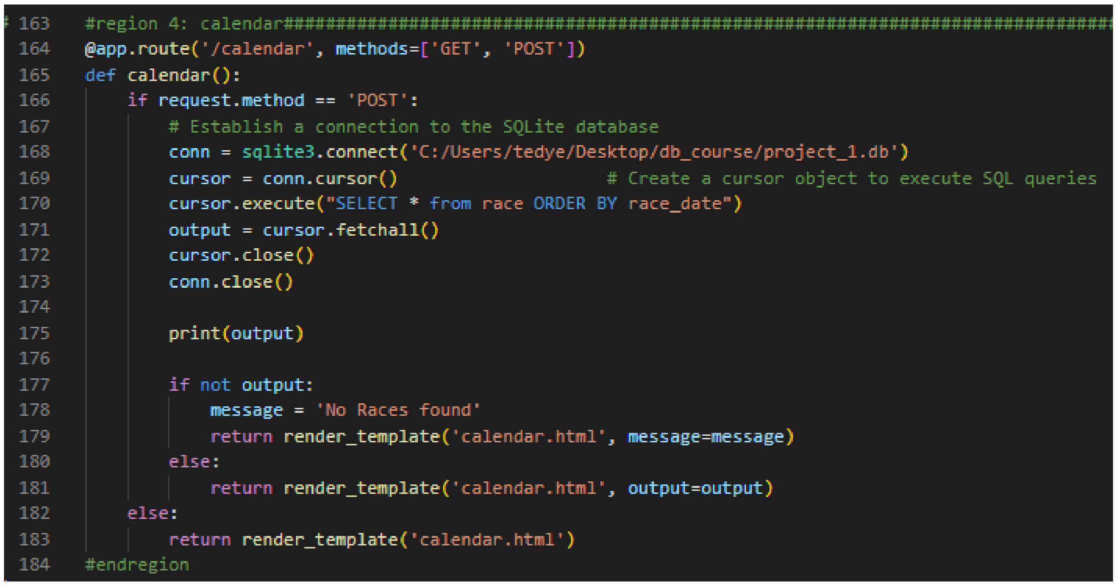
Athlete ID#:

OR, generate a plot of times for results1

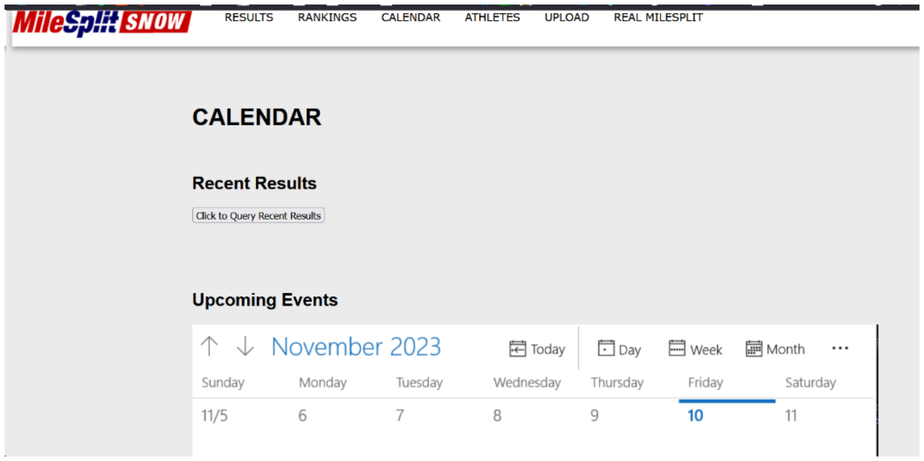
CALENDAR Python Script, HTML, rendered webpage



```
<!DOCTYPE html>
<html>
{% if web==FALSE %}
    <head>···
    </head>
    <div class="topbar">                                                                    <!--Top Navigation Bar-->···
    </div>

    <div class="a">                                                                 <!--This div uses style a to adds outer padding to everything-->
        <body style="background-color:#ebebeb">
            <h1>CALENDAR</h1>
            <i style="font-size:50">MileSplit Results</i>
            <br>
            <h2>Recent Results</h2>
            <form method='POST' action='calendar'>
                <input type="submit" value='Click to Query Recent Results'/>
            </form>
            {% if output %}
        <table>
            <tr>
                <td colspan="1" rowspan="1.9" align='center' style="background-color:#555555; color:#ffffff; font-family:arial"><b>Race ID<b></td>
                <td colspan="1" rowspan="1.9" align='center' style="background-color:#555555; color:#ffffff; font-family:arial"><b>Course ID<b></td>
                <td colspan="1" rowspan="1.9" align='center' style="background-color:#555555; color:#ffffff; font-family:arial"><b>Race Date<b></td>
                <td colspan="1" rowspan="1.9" align='center' style="background-color:#555555; color:#ffffff; font-family:arial"><b>Race League<b></td>
            </tr>
            <ul>
            {% for i in output %}
              <tr>
                  <td colspan="1" rowspan="1" align='center'>{{i[0]}}</td>
                  <td colspan="1" rowspan="1" align='center'>{{i[1]}}</td>
                  <td colspan="1" rowspan="1" align='center'>{{i[2]}}</td>
                  <td colspan="1" rowspan="1" align='center'>{{i[3]}} {{i[4]}}</td>
              </tr>
            {% endfor %}
            </ul>
        </table>
      {% endif %}

      {% if message %}
        <h2>{{message}}</h2>
      {% endif %}
            <br><br><br><br>
            <h2>Upcoming Events</h2>
            <img src="static/upcomingevents.png" title="calendar" align="center"/>

            {% if chart_image %}
                <img src="{{ chart_image }}"/>
            {% endif %}
        </body>
    </div>
    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
    <div class="bottom">                                                                    <!--Top Navigation Bar-->···
    </div>

{%else%}···
{%endif%}
```



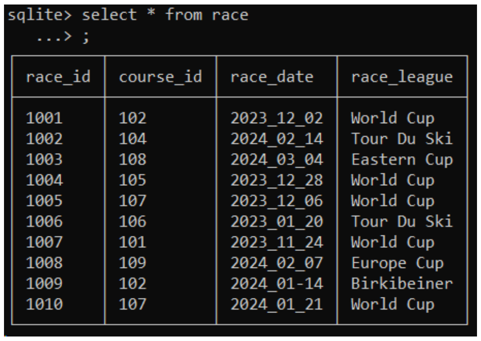
```python
#region 4: calendar########################################################################
@app.route('/calendar', methods=['GET', 'POST'])
def calendar():
    if request.method == 'POST':
        # Establish a connection to the SQLite database
        conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
        cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
        cursor.execute("SELECT * from race ORDER BY race_date")
        output = cursor.fetchall()
        cursor.close()
        conn.close()

        print(output)

        if not output:
            message = 'No Races found'
            return render_template('calendar.html', message=message)
        else:
            return render_template('calendar.html', output=output)
    else:
        return render_template('calendar.html')
#endregion
```



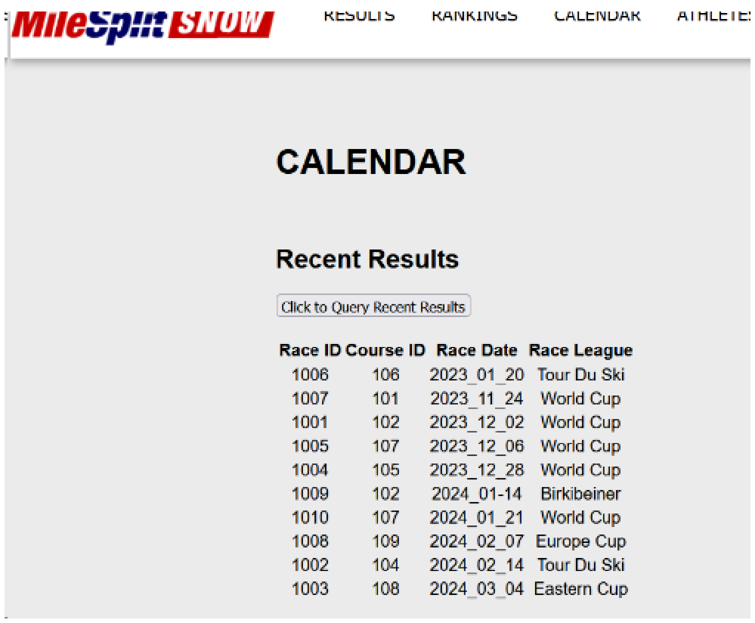
CALENDAR

Recent Results

Upcoming Events



```
sqlite> select * from race
   ...> ;

 race_id   course_id   race_date    race_league
 1001      102         2023_12_02   World Cup
 1002      104         2024_02_14   Tour Du Ski
 1003      108         2024_03_04   Eastern Cup
 1004      105         2023_12_28   World Cup
 1005      107         2023_12_06   World Cup
 1006      106         2023_01_20   Tour Du Ski
 1007      101         2023_11_24   World Cup
 1008      109         2024_02_07   Europe Cup
 1009      102         2024_01-14   Birkibeiner
 1010      107         2024_01_21   World Cup
```

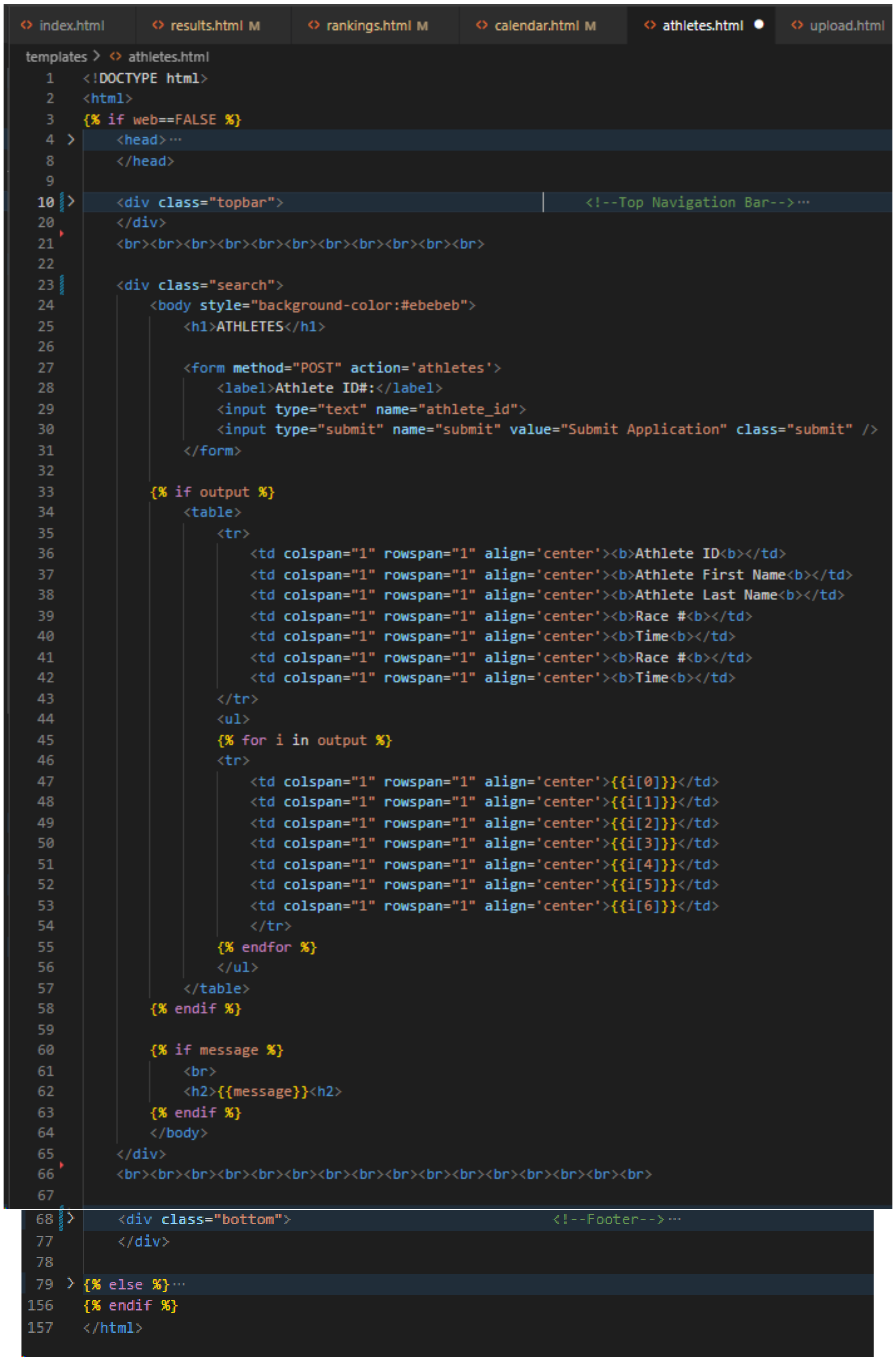


CALENDAR

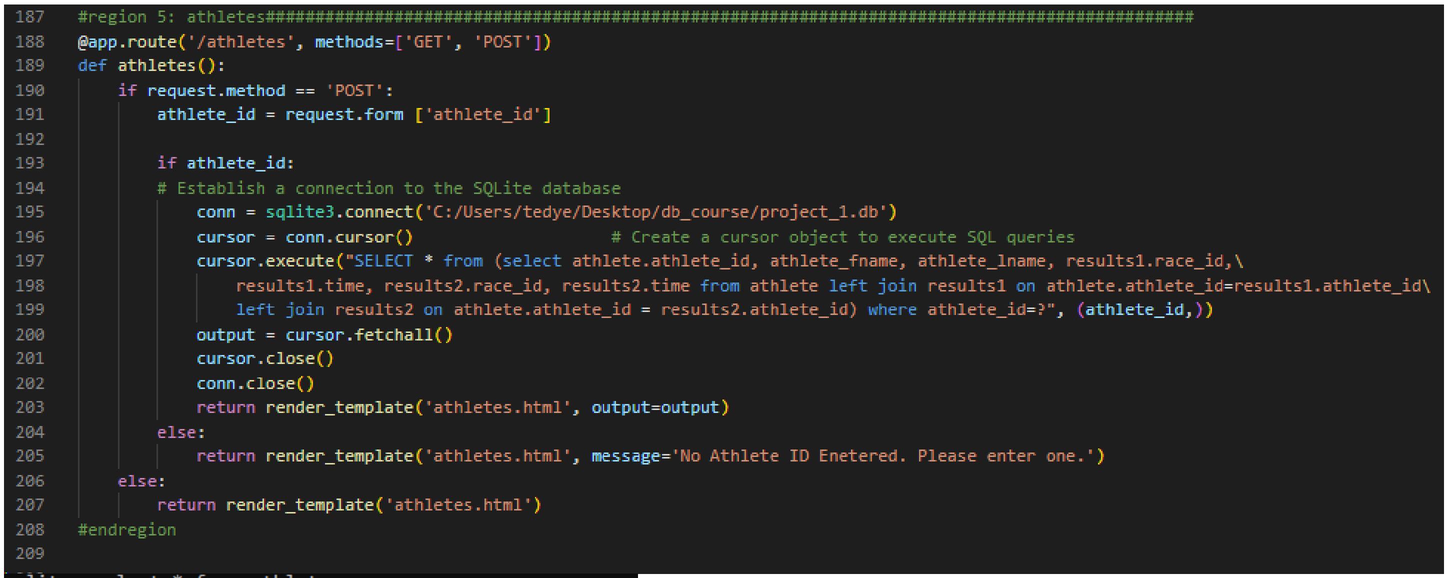
Recent Results

Race ID	Course ID	Race Date	Race League
1006	106	2023_01_20	Tour Du Ski
1007	101	2023_11_24	World Cup
1001	102	2023_12_02	World Cup
1005	107	2023_12_06	World Cup
1004	105	2023_12_28	World Cup
1009	102	2024_01-14	Birkibeiner
1010	107	2024_01_21	World Cup
1008	109	2024_02_07	Europe Cup
1002	104	2024_02_14	Tour Du Ski
1003	108	2024_03_04	Eastern Cup

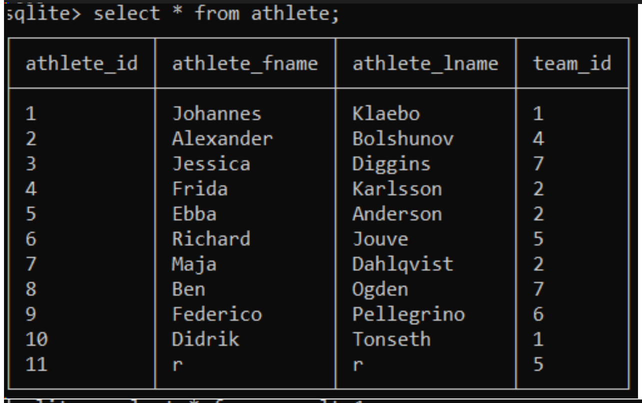
ATHLETES Python Script, HTML, rendered webpage



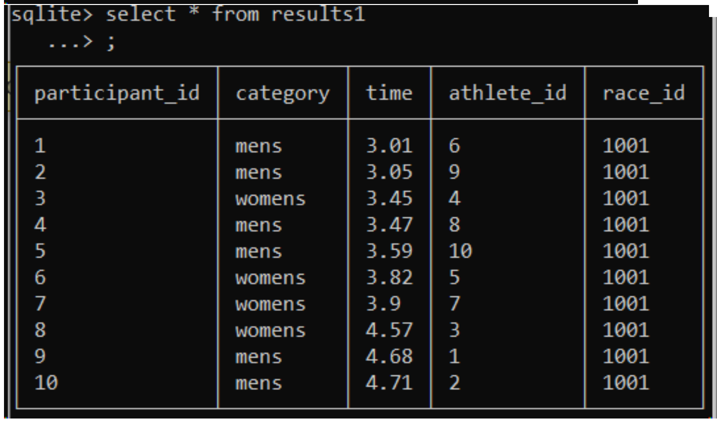
```html
<!DOCTYPE html>
<html>
{% if web==FALSE %}
    <head>⋯
    </head>

    <div class="topbar">                                    <!--Top Navigation Bar-->⋯
    </div>
    <br><br><br><br><br><br><br><br><br><br><br>

    <div class="search">
        <body style="background-color:#ebebeb">
            <h1>ATHLETES</h1>

            <form method="POST" action='athletes'>
                <label>Athlete ID#:</label>
                <input type="text" name="athlete_id">
                <input type="submit" name="submit" value="Submit Application" class="submit" />
            </form>

        {% if output %}
            <table>
                <tr>
                    <td colspan="1" rowspan="1" align='center'><b>Athlete ID<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Athlete First Name<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Athlete Last Name<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race #<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Time<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race #<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Time<b></td>
                </tr>
                <ul>
                {% for i in output %}
                <tr>
                    <td colspan="1" rowspan="1" align='center'>{{i[0]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[1]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[2]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[3]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[4]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[5]}}</td>
                    <td colspan="1" rowspan="1" align='center'>{{i[6]}}</td>
                    </tr>
                {% endfor %}
                </ul>
            </table>
        {% endif %}

        {% if message %}
            <br>
            <h2>{{message}}<h2>
        {% endif %}
        </body>
    </div>
    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

    <div class="bottom">                                <!--Footer-->⋯
    </div>

{% else %}⋯
{% endif %}
</html>
```



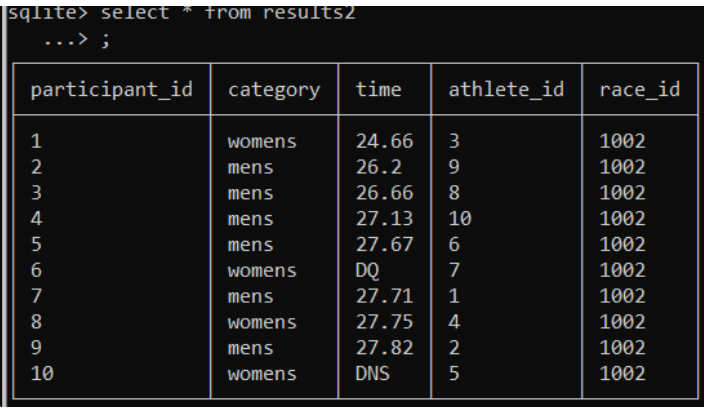
```python
#region 5: athletes###########################################################################################
@app.route('/athletes', methods=['GET', 'POST'])
def athletes():
    if request.method == 'POST':
        athlete_id = request.form ['athlete_id']

        if athlete_id:
        # Establish a connection to the SQLite database
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                    # Create a cursor object to execute SQL queries
            cursor.execute("SELECT * from (select athlete.athlete_id, athlete_fname, athlete_lname, results1.race_id,\
                results1.time, results2.race_id, results2.time from athlete left join results1 on athlete.athlete_id=results1.athlete_id\
                left join results2 on athlete.athlete_id = results2.athlete_id) where athlete_id=?", (athlete_id,))
            output = cursor.fetchall()
            cursor.close()
            conn.close()
            return render_template('athletes.html', output=output)
        else:
            return render_template('athletes.html', message='No Athlete ID Enetered. Please enter one.')
    else:
        return render_template('athletes.html')
#endregion
```



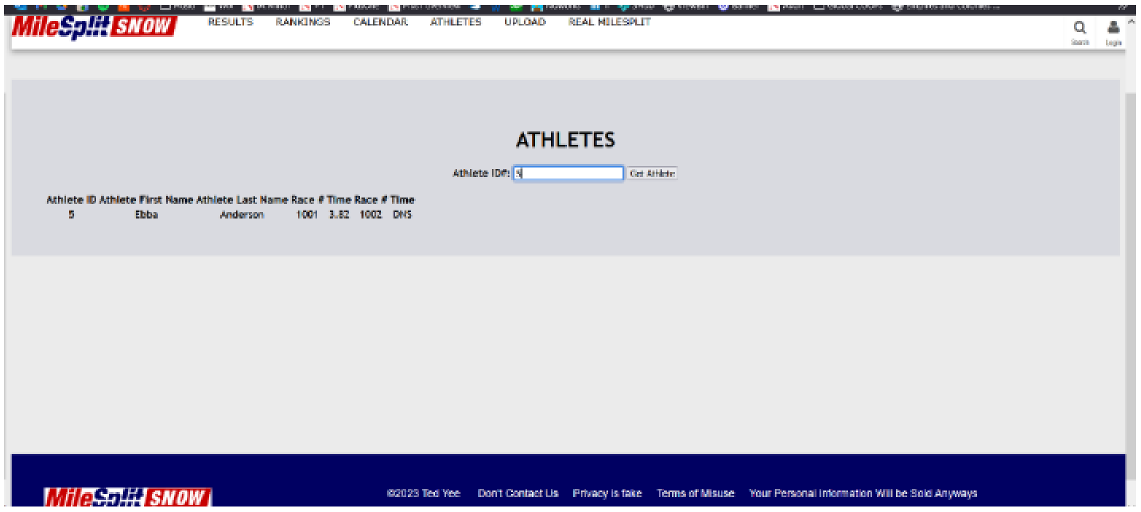
```
sqlite> select * from athlete;
+------------+---------------+---------------+---------+
| athlete_id | athlete_fname | athlete_lname | team_id |
+------------+---------------+---------------+---------+
| 1          | Johannes      | Klaebo        | 1       |
| 2          | Alexander     | Bolshunov     | 4       |
| 3          | Jessica       | Diggins       | 7       |
| 4          | Frida         | Karlsson      | 2       |
| 5          | Ebba          | Anderson      | 2       |
| 6          | Richard       | Jouve         | 5       |
| 7          | Maja          | Dahlqvist     | 2       |
| 8          | Ben           | Ogden         | 7       |
| 9          | Federico      | Pellegrino    | 6       |
| 10         | Didrik        | Tonseth       | 1       |
| 11         | r             | r             | 5       |
+------------+---------------+---------------+---------+
```



```
sqlite> select * from results1
   ...> ;
+----------------+----------+------+------------+---------+
| participant_id | category | time | athlete_id | race_id |
+----------------+----------+------+------------+---------+
| 1              | mens     | 3.01 | 6          | 1001    |
| 2              | mens     | 3.05 | 9          | 1001    |
| 3              | womens   | 3.45 | 4          | 1001    |
| 4              | mens     | 3.47 | 8          | 1001    |
| 5              | mens     | 3.59 | 10         | 1001    |
| 6              | womens   | 3.82 | 5          | 1001    |
| 7              | womens   | 3.9  | 7          | 1001    |
| 8              | womens   | 4.57 | 3          | 1001    |
| 9              | mens     | 4.68 | 1          | 1001    |
| 10             | mens     | 4.71 | 2          | 1001    |
+----------------+----------+------+------------+---------+
```



```
sqlite> select * from results2
   ...> ;
+----------------+----------+-------+------------+---------+
| participant_id | category | time  | athlete_id | race_id |
+----------------+----------+-------+------------+---------+
| 1              | womens   | 24.66 | 3          | 1002    |
| 2              | mens     | 26.2  | 9          | 1002    |
| 3              | mens     | 26.66 | 8          | 1002    |
| 4              | mens     | 27.13 | 10         | 1002    |
| 5              | mens     | 27.67 | 6          | 1002    |
| 6              | womens   | DQ    | 7          | 1002    |
| 7              | mens     | 27.71 | 1          | 1002    |
| 8              | womens   | 27.75 | 4          | 1002    |
| 9              | mens     | 27.82 | 2          | 1002    |
| 10             | womens   | DNS   | 5          | 1002    |
+----------------+----------+-------+------------+---------+
```



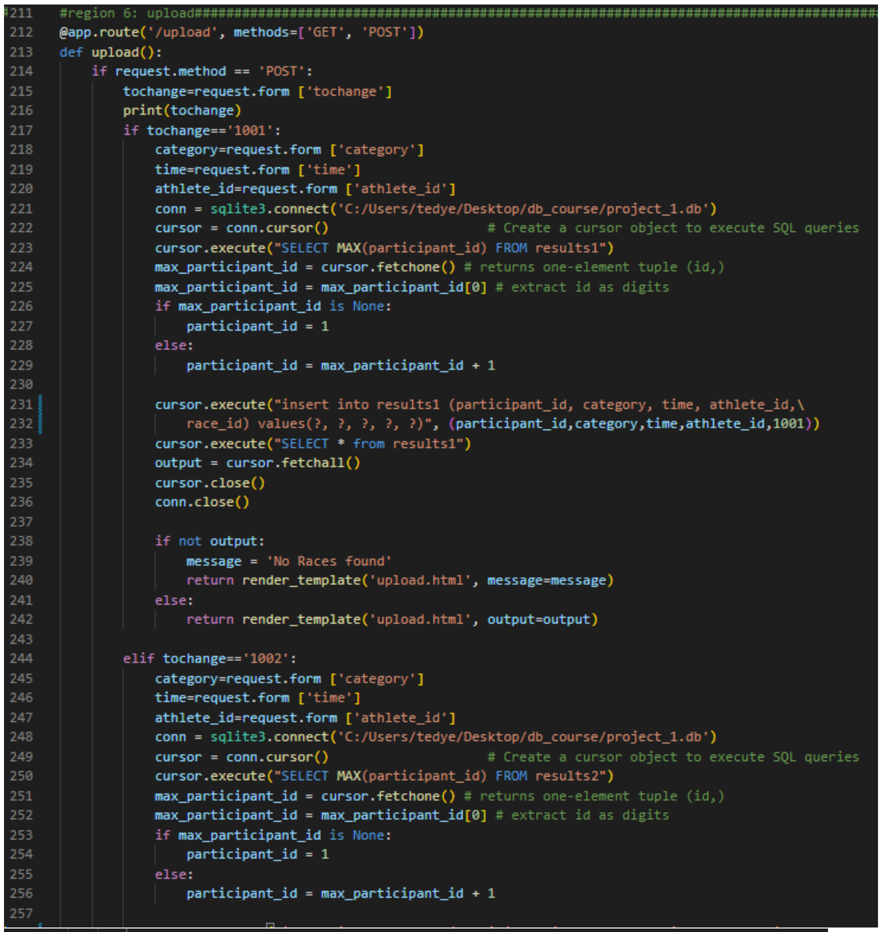
ATHLETES

Athlete ID#: 5

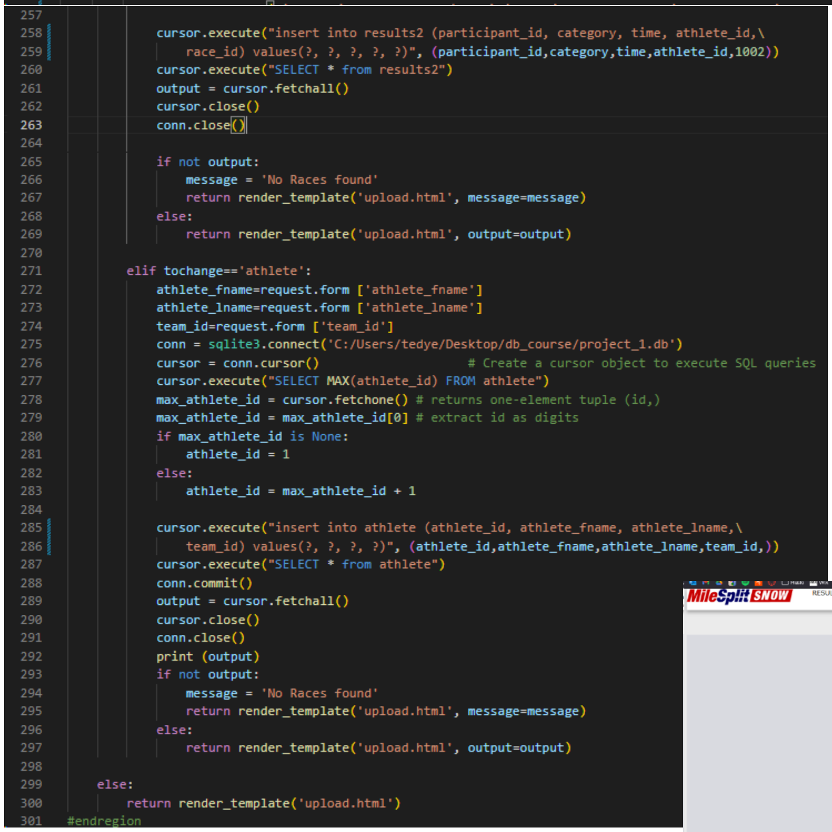
Athlete ID	Athlete First Name	Athlete Last Name	Race #	Time	Race #	Time
5	Ebba	Anderson	1001	3.82	1002	DNS

©2023 Ted Yee   Don't Contact Us   Privacy is fake   Terms of Misuse   Your Personal Information Will be Sold Anyways

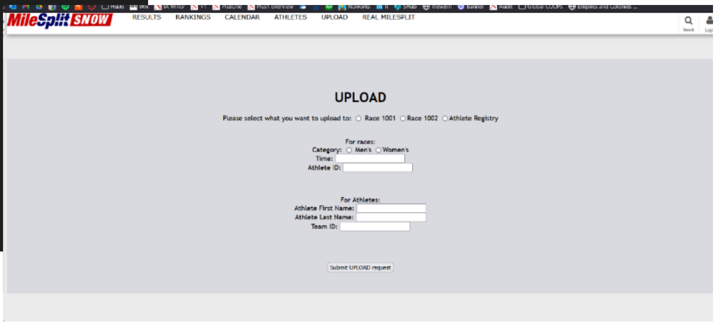
UPLOAD Python Script, HTML, rendered webpage



```python
#region 6: upload##########################################################################
@app.route('/upload', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        tochange=request.form ['tochange']
        print(tochange)
        if tochange=='1001':
            category=request.form ['category']
            time=request.form ['time']
            athlete_id=request.form ['athlete_id']
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                          # Create a cursor object to execute SQL queries
            cursor.execute("SELECT MAX(participant_id) FROM results1")
            max_participant_id = cursor.fetchone() # returns one-element tuple (id,)
            max_participant_id = max_participant_id[0] # extract id as digits
            if max_participant_id is None:
                participant_id = 1
            else:
                participant_id = max_participant_id + 1

            cursor.execute("insert into results1 (participant_id, category, time, athlete_id,\
                race_id) values(?, ?, ?, ?, ?)", (participant_id,category,time,athlete_id,1001))
            cursor.execute("SELECT * from results1")
            output = cursor.fetchall()
            cursor.close()
            conn.close()

            if not output:
                message = 'No Races found'
                return render_template('upload.html', message=message)
            else:
                return render_template('upload.html', output=output)

        elif tochange=='1002':
            category=request.form ['category']
            time=request.form ['time']
            athlete_id=request.form ['athlete_id']
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                          # Create a cursor object to execute SQL queries
            cursor.execute("SELECT MAX(participant_id) FROM results2")
            max_participant_id = cursor.fetchone() # returns one-element tuple (id,)
            max_participant_id = max_participant_id[0] # extract id as digits
            if max_participant_id is None:
                participant_id = 1
            else:
                participant_id = max_participant_id + 1

```



```python

            cursor.execute("insert into results2 (participant_id, category, time, athlete_id,\
                race_id) values(?, ?, ?, ?, ?)", (participant_id,category,time,athlete_id,1002))
            cursor.execute("SELECT * from results2")
            output = cursor.fetchall()
            cursor.close()
            conn.close()

            if not output:
                message = 'No Races found'
                return render_template('upload.html', message=message)
            else:
                return render_template('upload.html', output=output)

        elif tochange=='athlete':
            athlete_fname=request.form ['athlete_fname']
            athlete_lname=request.form ['athlete_lname']
            team_id=request.form ['team_id']
            conn = sqlite3.connect('C:/Users/tedye/Desktop/db_course/project_1.db')
            cursor = conn.cursor()                          # Create a cursor object to execute SQL queries
            cursor.execute("SELECT MAX(athlete_id) FROM athlete")
            max_athlete_id = cursor.fetchone() # returns one-element tuple (id,)
            max_athlete_id = max_athlete_id[0] # extract id as digits
            if max_athlete_id is None:
                athlete_id = 1
            else:
                athlete_id = max_athlete_id + 1

            cursor.execute("insert into athlete (athlete_id, athlete_fname, athlete_lname,\
                team_id) values(?, ?, ?, ?)", (athlete_id,athlete_fname,athlete_lname,team_id,))
            cursor.execute("SELECT * from athlete")
            conn.commit()
            output = cursor.fetchall()
            cursor.close()
            conn.close()
            print (output)
            if not output:
                message = 'No Races found'
                return render_template('upload.html', message=message)
            else:
                return render_template('upload.html', output=output)

    else:
        return render_template('upload.html')
#endregion
```



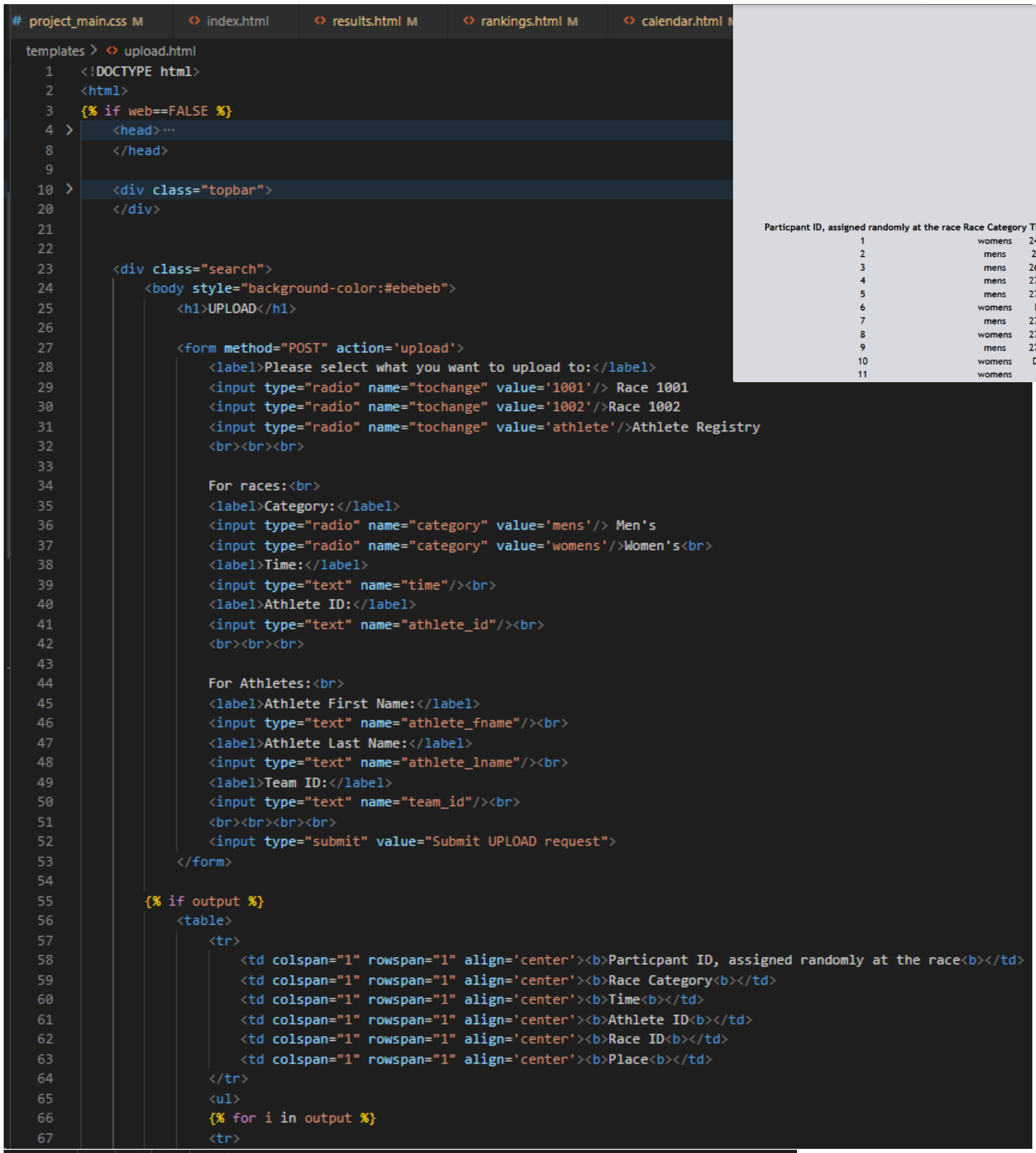
UPLOAD

Please select what you want to upload to: ○ Race 1001 ○ Race 1002 ○ Athlete Registry

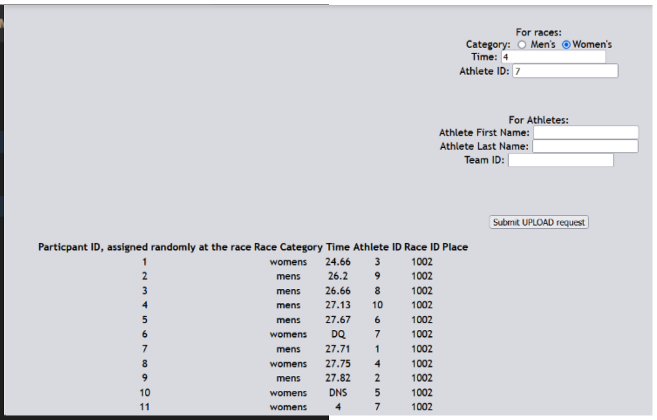
For races:
Category: ○ Men's ○ Women's
Time:
Athlete ID:

For Athletes:
Athlete First Name:
Athlete Last Name:
Team ID:

Submit UPLOAD request



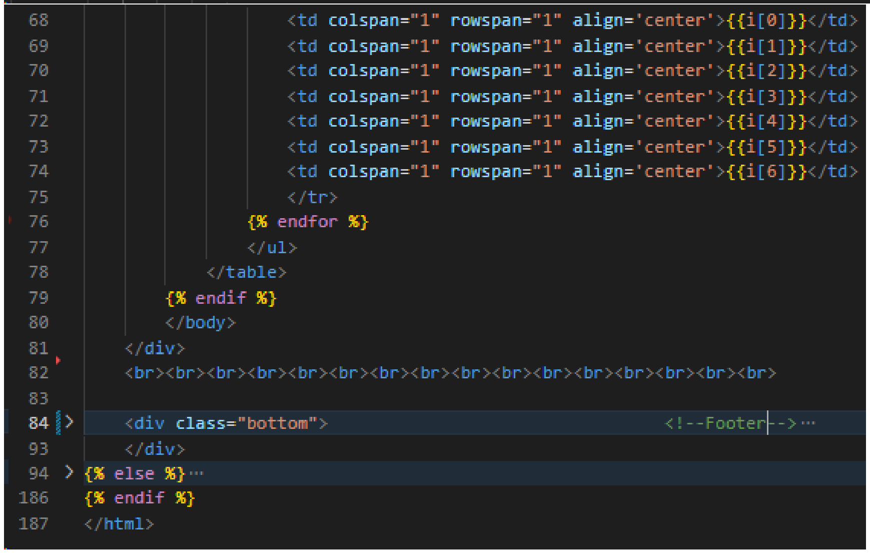
```html
<!DOCTYPE html>
<html>
{% if web==FALSE %}
    <head>···
    </head>

    <div class="topbar">
    </div>


    <div class="search">
        <body style="background-color:#ebebeb">
            <h1>UPLOAD</h1>

            <form method="POST" action='upload'>
                <label>Please select what you want to upload to:</label>
                <input type="radio" name="tochange" value='1001'/> Race 1001
                <input type="radio" name="tochange" value='1002'/>Race 1002
                <input type="radio" name="tochange" value='athlete'/>Athlete Registry
                <br><br><br>

                For races:<br>
                <label>Category:</label>
                <input type="radio" name="category" value='mens'/> Men's
                <input type="radio" name="category" value='womens'/>Women's<br>
                <label>Time:</label>
                <input type="text" name="time"/><br>
                <label>Athlete ID:</label>
                <input type="text" name="athlete_id"/><br>
                <br><br><br>

                For Athletes:<br>
                <label>Athlete First Name:</label>
                <input type="text" name="athlete_fname"/><br>
                <label>Athlete Last Name:</label>
                <input type="text" name="athlete_lname"/><br>
                <label>Team ID:</label>
                <input type="text" name="team_id"/><br>
                <br><br><br><br>
                <input type="submit" value="Submit UPLOAD request">
            </form>

        {% if output %}
            <table>
                <tr>
                    <td colspan="1" rowspan="1" align='center'><b>Particpant ID, assigned randomly at the race<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race Category<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Time<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Athlete ID<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Race ID<b></td>
                    <td colspan="1" rowspan="1" align='center'><b>Place<b></td>
                </tr>
                <ul>
                {% for i in output %}
                <tr>
```



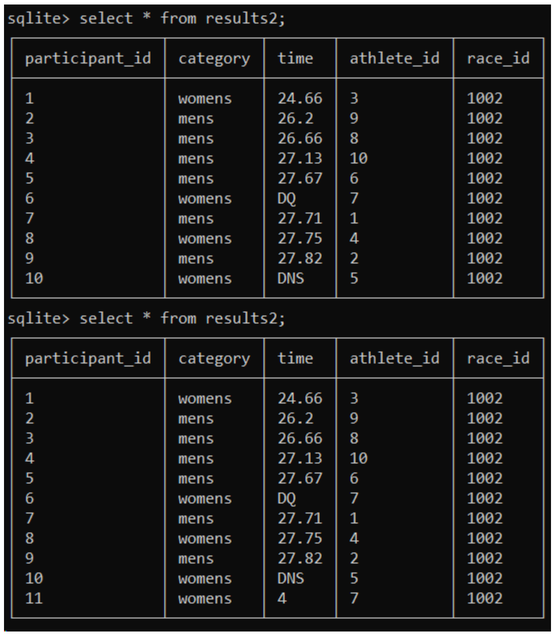
For races:
Category: ○ Men's [✓ Women's]
Time: 4
Athlete ID: 7

For Athletes:
Athlete First Name:
Athlete Last Name:
Team ID:

Particpant ID, assigned randomly at the race	Race Category	Time	Athlete ID	Race ID	Place
1	womens	24.66	3	1002	
2	mens	26.2	9	1002	
3	mens	26.66	8	1002	
4	mens	27.13	10	1002	
5	mens	27.67	6	1002	
6	womens	DQ	7	1002	
7	mens	27.71	1	1002	
8	womens	27.75	4	1002	
9	mens	27.82	2	1002	
10	womens	DNS	5	1002	
11	womens	4	7	1002	

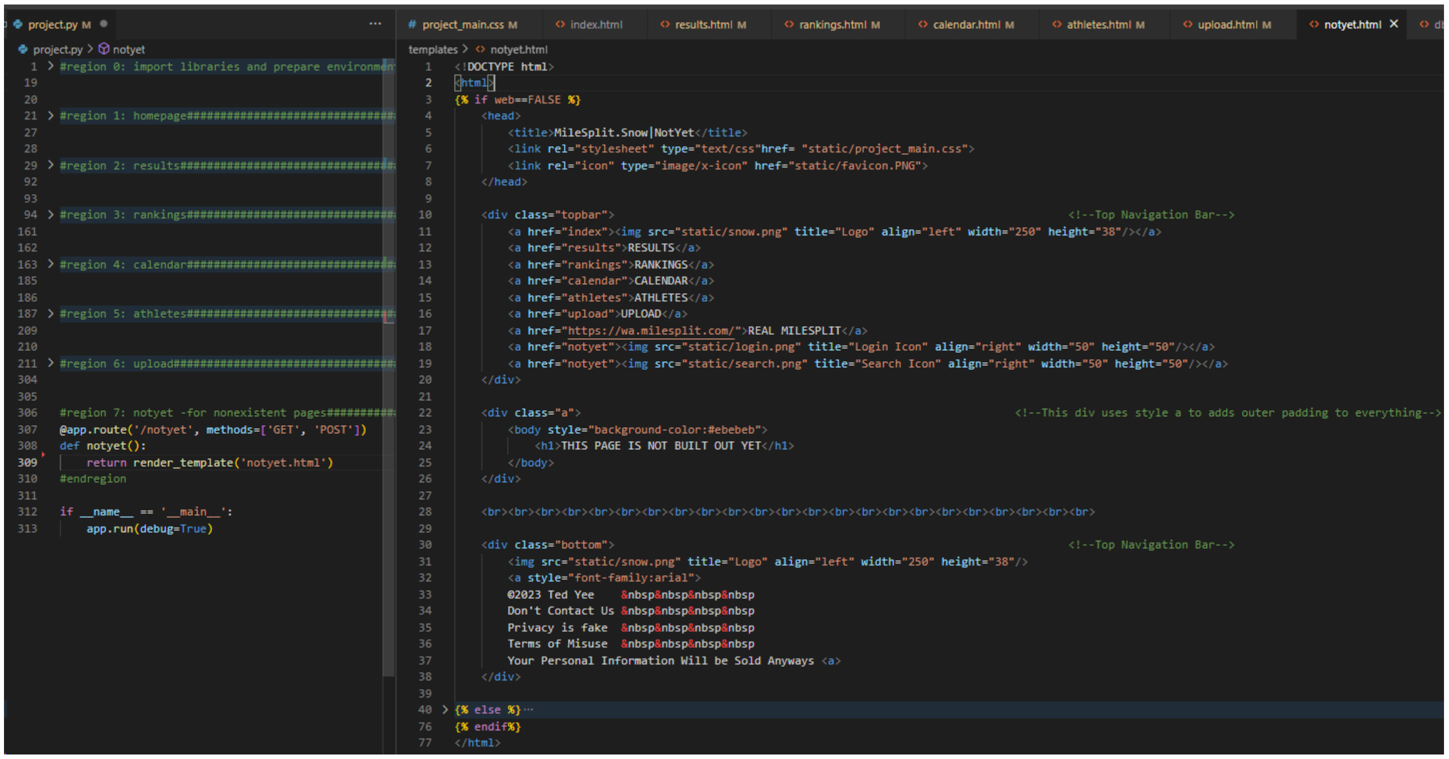


```html
                        <td colspan="1" rowspan="1" align='center'>{{i[0]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[1]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[2]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[3]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[4]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[5]}}</td>
                        <td colspan="1" rowspan="1" align='center'>{{i[6]}}</td>
                    </tr>
                {% endfor %}
                </ul>
            </table>
        {% endif %}
        </body>
    </div>
    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

    <div class="bottom">                                   <!--Footer-->···
    </div>
{% else %}···
{% endif %}
</html>
```



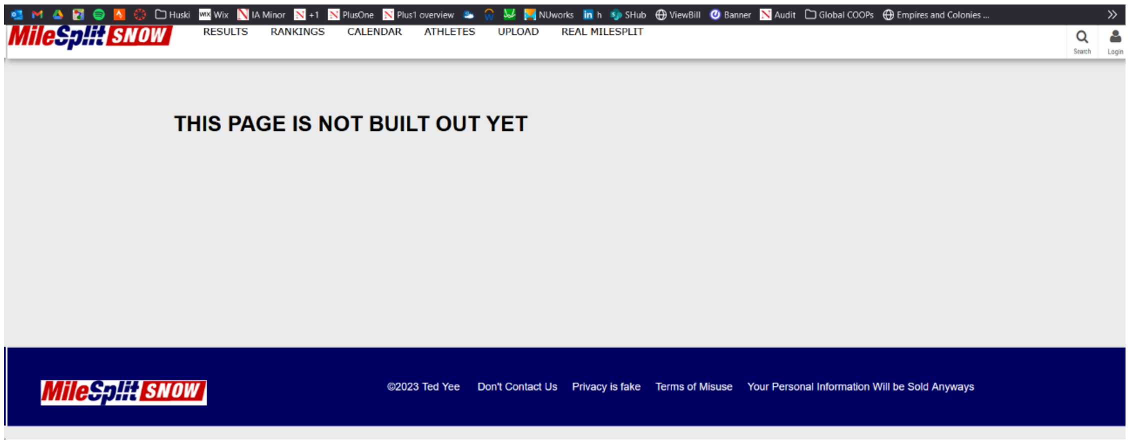
```
sqlite> select * from results2;
┌────────────────┬──────────┬───────┬────────────┬─────────┐
│ participant_id │ category │ time  │ athlete_id │ race_id │
├────────────────┼──────────┼───────┼────────────┼─────────┤
│ 1              │ womens   │ 24.66 │ 3          │ 1002    │
│ 2              │ mens     │ 26.2  │ 9          │ 1002    │
│ 3              │ mens     │ 26.66 │ 8          │ 1002    │
│ 4              │ mens     │ 27.13 │ 10         │ 1002    │
│ 5              │ mens     │ 27.67 │ 6          │ 1002    │
│ 6              │ womens   │ DQ    │ 7          │ 1002    │
│ 7              │ mens     │ 27.71 │ 1          │ 1002    │
│ 8              │ womens   │ 27.75 │ 4          │ 1002    │
│ 9              │ mens     │ 27.82 │ 2          │ 1002    │
│ 10             │ womens   │ DNS   │ 5          │ 1002    │
└────────────────┴──────────┴───────┴────────────┴─────────┘
sqlite> select * from results2;
┌────────────────┬──────────┬───────┬────────────┬─────────┐
│ participant_id │ category │ time  │ athlete_id │ race_id │
├────────────────┼──────────┼───────┼────────────┼─────────┤
│ 1              │ womens   │ 24.66 │ 3          │ 1002    │
│ 2              │ mens     │ 26.2  │ 9          │ 1002    │
│ 3              │ mens     │ 26.66 │ 8          │ 1002    │
│ 4              │ mens     │ 27.13 │ 10         │ 1002    │
│ 5              │ mens     │ 27.67 │ 6          │ 1002    │
│ 6              │ womens   │ DQ    │ 7          │ 1002    │
│ 7              │ mens     │ 27.71 │ 1          │ 1002    │
│ 8              │ womens   │ 27.75 │ 4          │ 1002    │
│ 9              │ mens     │ 27.82 │ 2          │ 1002    │
│ 10             │ womens   │ DNS   │ 5          │ 1002    │
│ 11             │ womens   │ 4     │ 7          │ 1002    │
└────────────────┴──────────┴───────┴────────────┴─────────┘
```

SEARCH and **LOGIN** pages were left intentionally blank, but I wanted them to be working links Python Script, HTML, rendered webpage.



```python
#region 0: import libraries and prepare environmen
#region 1: homepage##################################
#region 2: results###################################
#region 3: rankings##################################
#region 4: calendar##################################
#region 5: athletes##################################
#region 6: upload####################################

#region 7: notyet -for nonexistent pages###########
@app.route('/notyet', methods=['GET', 'POST'])
def notyet():
    return render_template('notyet.html')
#endregion

if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html>
{% if web==FALSE %}
    <head>
        <title>MileSplit.Snow|NotYet</title>
        <link rel="stylesheet" type="text/css"href= "static/project_main.css">
        <link rel="icon" type="image/x-icon" href="static/favicon.PNG">
    </head>

    <div class="topbar">                                                        <!--Top Navigation Bar-->
        <a href="index"><img src="static/snow.png" title="Logo" align="left" width="250" height="38"/></a>
        <a href="results">RESULTS</a>
        <a href="rankings">RANKINGS</a>
        <a href="calendar">CALENDAR</a>
        <a href="athletes">ATHLETES</a>
        <a href="upload">UPLOAD</a>
        <a href="https://wa.milesplit.com/">REAL MILESPLIT</a>
        <a href="notyet"><img src="static/login.png" title="Login Icon" align="right" width="50" height="50"/></a>
        <a href="notyet"><img src="static/search.png" title="Search Icon" align="right" width="50" height="50"/></a>
    </div>

    <div class="a">                                                    <!--This div uses style a to adds outer padding to everything-->
        <body style="background-color:#ebebeb">
            <h1>THIS PAGE IS NOT BUILT OUT YET</h1>
        </body>
    </div>

    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

    <div class="bottom">                                                        <!--Top Navigation Bar-->
        <img src="static/snow.png" title="Logo" align="left" width="250" height="38"/>
        <a style="font-family:arial">
        ©2023 Ted Yee    &nbsp&nbsp&nbsp&nbsp
        Don't Contact Us &nbsp&nbsp&nbsp&nbsp
        Privacy is fake  &nbsp&nbsp&nbsp&nbsp
        Terms of Misuse  &nbsp&nbsp&nbsp&nbsp
        Your Personal Information Will be Sold Anyways <a>
    </div>

{% else %}···
{% endif%}
</html>
```



THIS PAGE IS NOT BUILT OUT YET

©2023 Ted Yee Don't Contact Us Privacy is fake Terms of Misuse Your Personal Information Will be Sold Anyways

Conclusion and Going Forward

This project is complete but is definitely capable of being expanded much further. From simply making the HTML look nicer to adding other SQL queries, there is a lot to do that could make it a usable, practical site. The biggest change that could be made, however, would be to make the site hosted on a more robust server that allows the python commands to connect to SQL and manipulate and query the data. Then it would be a truly live database site that could live up to its ability to handle larger amounts of data.

Features that I wanted to include, but ran out of time for, were methods to upload new results in their entirety or time races directly on the website and a general search feature that would allow querying of all data. The first would be easily completed via SQL “CREATE TABLE”, then “.import” or use python to create a stopwatch and create records from it. The search feature could either provide a way to query all information in a given table using SQL “SELECT * from ____” or a real search could be conducted with a search field that controls a “SELECT * from _ WHERE ____=input”.