UFC Fighter's Rating Percentage Index

Overview:

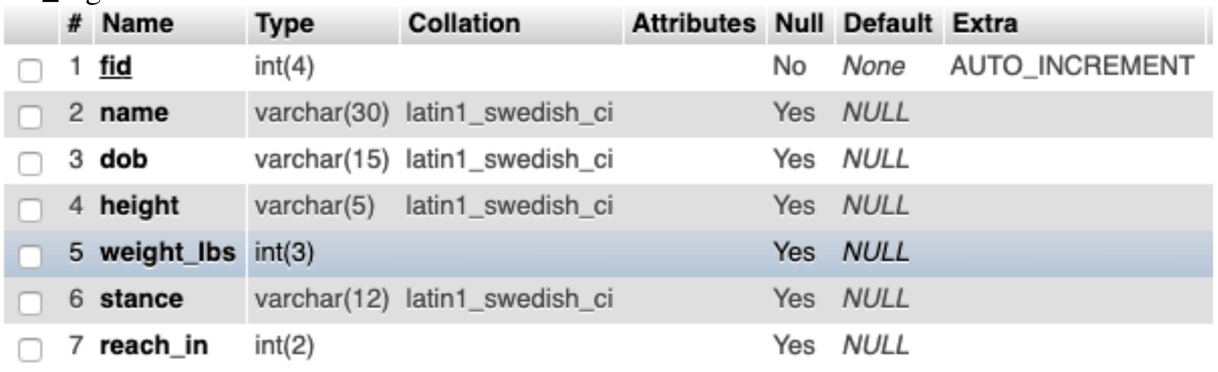
The purpose of the application is to take a pair of datasets on UFC fighters and UFC Match history to be able for a user to select two names, recall basic information about the fighters, and determine who is the most likely to be the victor if they were to partake in a match against one another.

I: Data Set (Cleaning and Database Creation)

The dataset that is being used is a public set on Kaggle titled, "UFC-Fight historical data from 1993 to 2019". The user web scraped the data from UFC Stats and created multiple csv files. I've taken two of the files, one on historical events and one on details of each fighter, to then clean and throw into two separate tables in phpMyAdmin.

Kaggle Dataset: https://www.kaggle.com/rajeevw/ufcdata
UFC STATS: http://ufcstats.com/statistics/events/completed

ufc_Fighters Table:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	fid	int(4)			No	None	AUTO_INCREMENT
2	name	varchar(30)	latin1_swedish_ci		Yes	NULL	
3	dob	varchar(15)	latin1_swedish_ci		Yes	NULL	
4	height	varchar(5)	latin1_swedish_ci		Yes	NULL	
5	weight_lbs	int(3)			Yes	NULL	
6	stance	varchar(12)	latin1_swedish_ci		Yes	NULL	
7	reach_in	int(2)			Yes	NULL	

Figure 1.1 – shows table structure

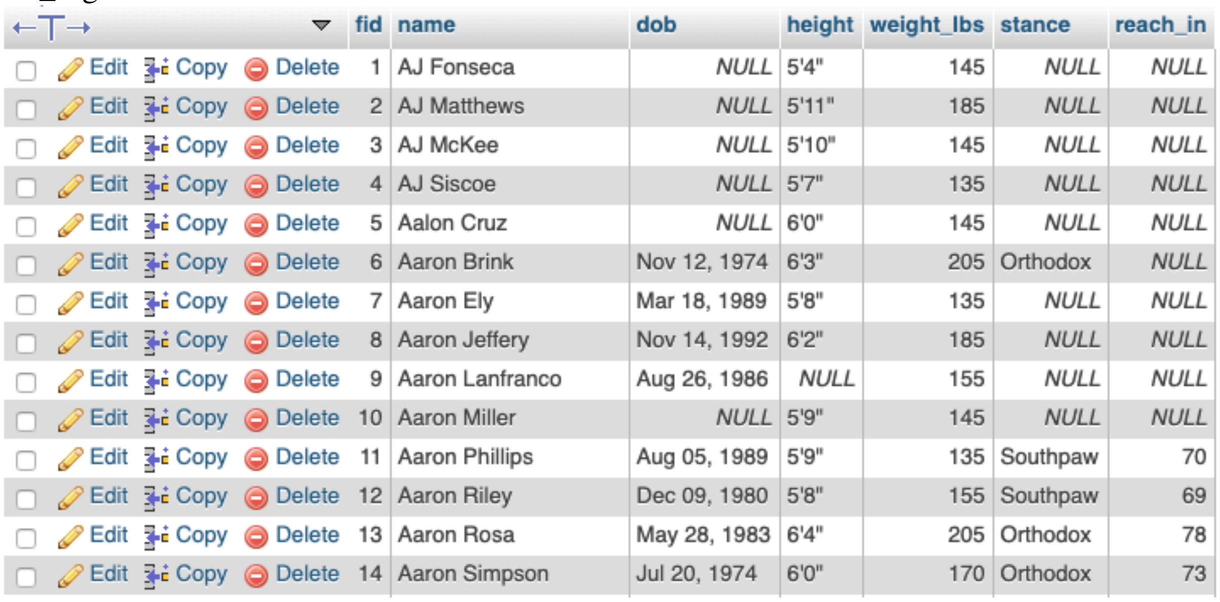
ufc_Fighters Table:

- fid
    - Fighter id – auto_incremented id number for each fighter
- name
    - Both first and last name
    - For purpose of the application there was no need to separate into fname and lname columns
- dob
    - date of birth in a string
    - for application the only purpose is to return a string and append in json, no need to date type, varchar works perfectly fine
- height
    - shows in 0'0'' format for recall as string purposes
    - no need to separate into multiple columns or change formatting to int type

- weight_lbs
    - shows weight in pounds
- stance
    - there are different fighting stances styles for ufc

Not all information is known, unknown fields are replaced with NULL.

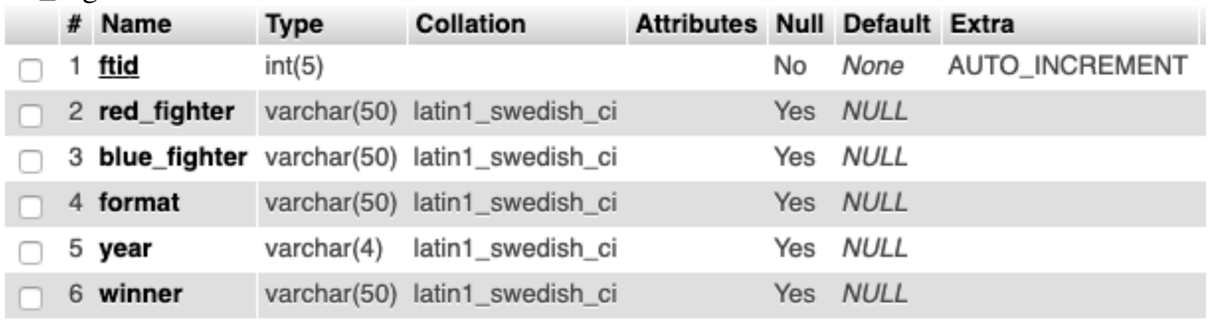
ufc_Fighters Table:



←T→	fid	name	dob	height	weight_lbs	stance	reach_in
Edit Copy Delete	1	AJ Fonseca	NULL	5'4"	145	NULL	NULL
Edit Copy Delete	2	AJ Matthews	NULL	5'11"	185	NULL	NULL
Edit Copy Delete	3	AJ McKee	NULL	5'10"	145	NULL	NULL
Edit Copy Delete	4	AJ Siscoe	NULL	5'7"	135	NULL	NULL
Edit Copy Delete	5	Aalon Cruz	NULL	6'0"	145	NULL	NULL
Edit Copy Delete	6	Aaron Brink	Nov 12, 1974	6'3"	205	Orthodox	NULL
Edit Copy Delete	7	Aaron Ely	Mar 18, 1989	5'8"	135	NULL	NULL
Edit Copy Delete	8	Aaron Jeffery	Nov 14, 1992	6'2"	185	NULL	NULL
Edit Copy Delete	9	Aaron Lanfranco	Aug 26, 1986	NULL	155	NULL	NULL
Edit Copy Delete	10	Aaron Miller	NULL	5'9"	145	NULL	NULL
Edit Copy Delete	11	Aaron Phillips	Aug 05, 1989	5'9"	135	Southpaw	70
Edit Copy Delete	12	Aaron Riley	Dec 09, 1980	5'8"	155	Southpaw	69
Edit Copy Delete	13	Aaron Rosa	May 28, 1983	6'4"	205	Orthodox	78
Edit Copy Delete	14	Aaron Simpson	Jul 20, 1974	6'0"	170	Orthodox	73

Figure 1.2 – shows selection of table after data insert

ufc_Fight Table:



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	ftid	int(5)			No	None	AUTO_INCREMENT
2	red_fighter	varchar(50)	latin1_swedish_ci		Yes	NULL	
3	blue_fighter	varchar(50)	latin1_swedish_ci		Yes	NULL	
4	format	varchar(50)	latin1_swedish_ci		Yes	NULL	
5	year	varchar(4)	latin1_swedish_ci		Yes	NULL	
6	winner	varchar(50)	latin1_swedish_ci		Yes	NULL	

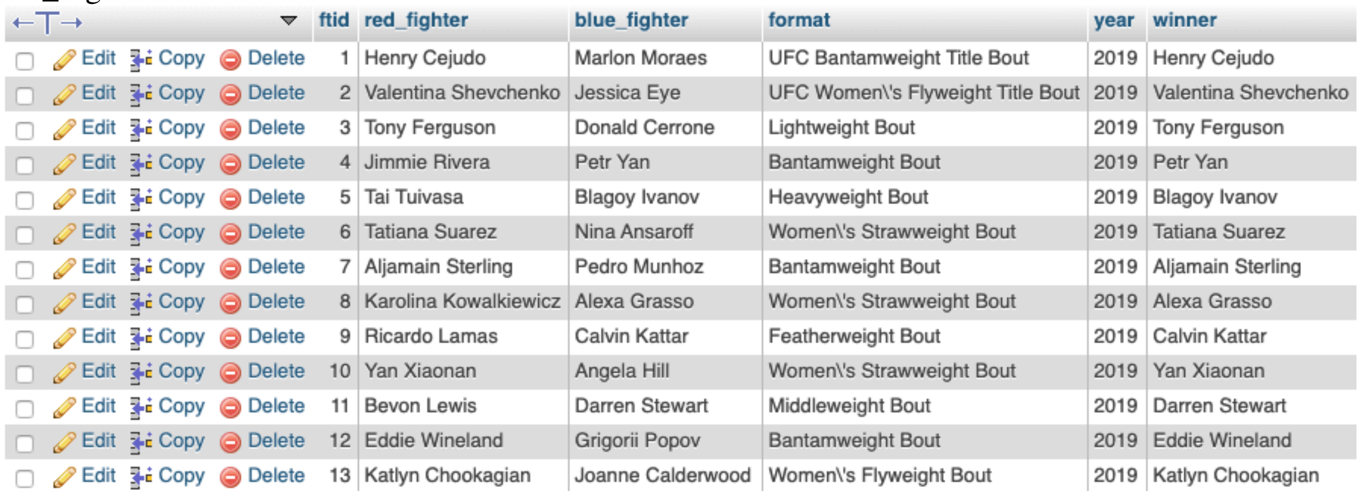
Figure 1.3 – shows table structure

ufc_Fight Table:

- ftid
    - fight id - auto_incremented id number for each individual event
- red_fighter
    - fighter on the red side
    - same format as ‘name’ in ufc_Fighters table

- blue_fighter
    - fighter on the blue side
    - same format as ‘name’ in ufc_Fighters table
- format
    - title of the fight event
    - weight division
    - men’s/women’s
    - could be a normal fight ‘Bout’ or a title fight
    - for purposes of application, no need to normalize
- year
    - year of event
    - for purposes of application, no need for int or date type
- winner
    - who won the event
    - same format as ‘name’ in ufc_Fighters table

ufc_Fight Table:

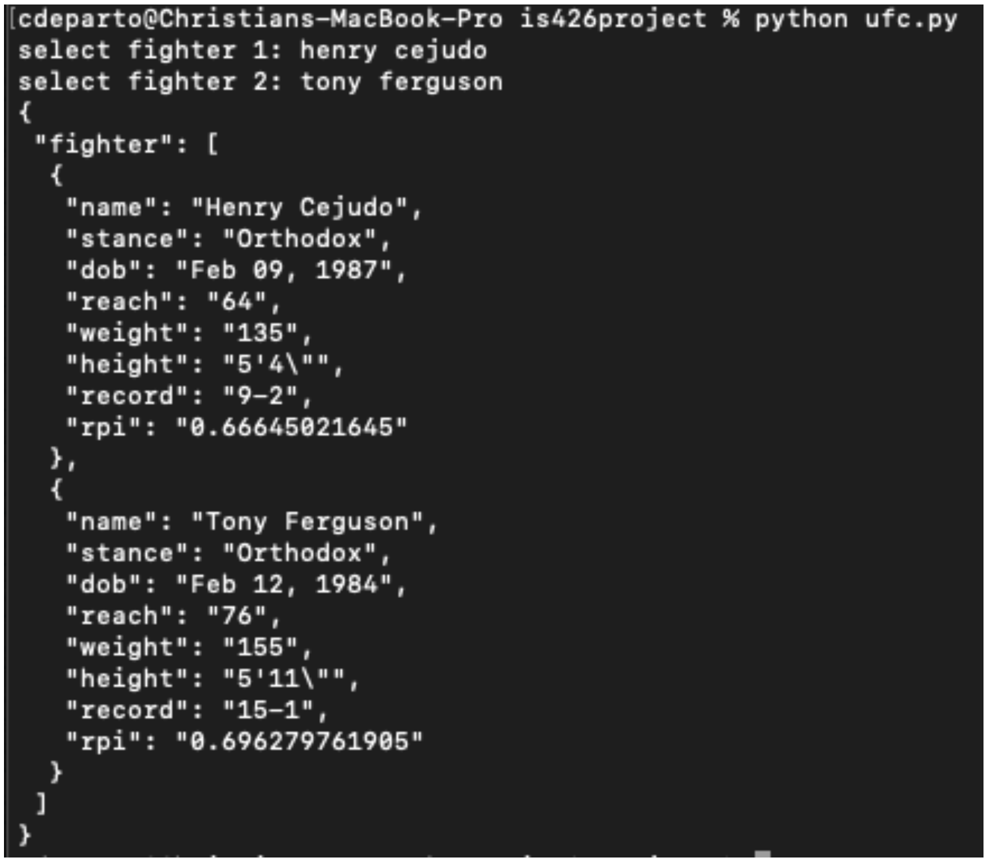


←T→	ftid	red_fighter	blue_fighter	format	year	winner
Edit Copy Delete	1	Henry Cejudo	Marlon Moraes	UFC Bantamweight Title Bout	2019	Henry Cejudo
Edit Copy Delete	2	Valentina Shevchenko	Jessica Eye	UFC Women\'s Flyweight Title Bout	2019	Valentina Shevchenko
Edit Copy Delete	3	Tony Ferguson	Donald Cerrone	Lightweight Bout	2019	Tony Ferguson
Edit Copy Delete	4	Jimmie Rivera	Petr Yan	Bantamweight Bout	2019	Petr Yan
Edit Copy Delete	5	Tai Tuivasa	Blagoy Ivanov	Heavyweight Bout	2019	Blagoy Ivanov
Edit Copy Delete	6	Tatiana Suarez	Nina Ansaroff	Women\'s Strawweight Bout	2019	Tatiana Suarez
Edit Copy Delete	7	Aljamain Sterling	Pedro Munhoz	Bantamweight Bout	2019	Aljamain Sterling
Edit Copy Delete	8	Karolina Kowalkiewicz	Alexa Grasso	Women\'s Strawweight Bout	2019	Alexa Grasso
Edit Copy Delete	9	Ricardo Lamas	Calvin Kattar	Featherweight Bout	2019	Calvin Kattar
Edit Copy Delete	10	Yan Xiaonan	Angela Hill	Women\'s Strawweight Bout	2019	Yan Xiaonan
Edit Copy Delete	11	Bevon Lewis	Darren Stewart	Middleweight Bout	2019	Darren Stewart
Edit Copy Delete	12	Eddie Wineland	Grigorii Popov	Bantamweight Bout	2019	Eddie Wineland
Edit Copy Delete	13	Katlyn Chookagian	Joanne Calderwood	Women\'s Flyweight Bout	2019	Katlyn Chookagian

Figure1.4 - shows selection of table after data insert

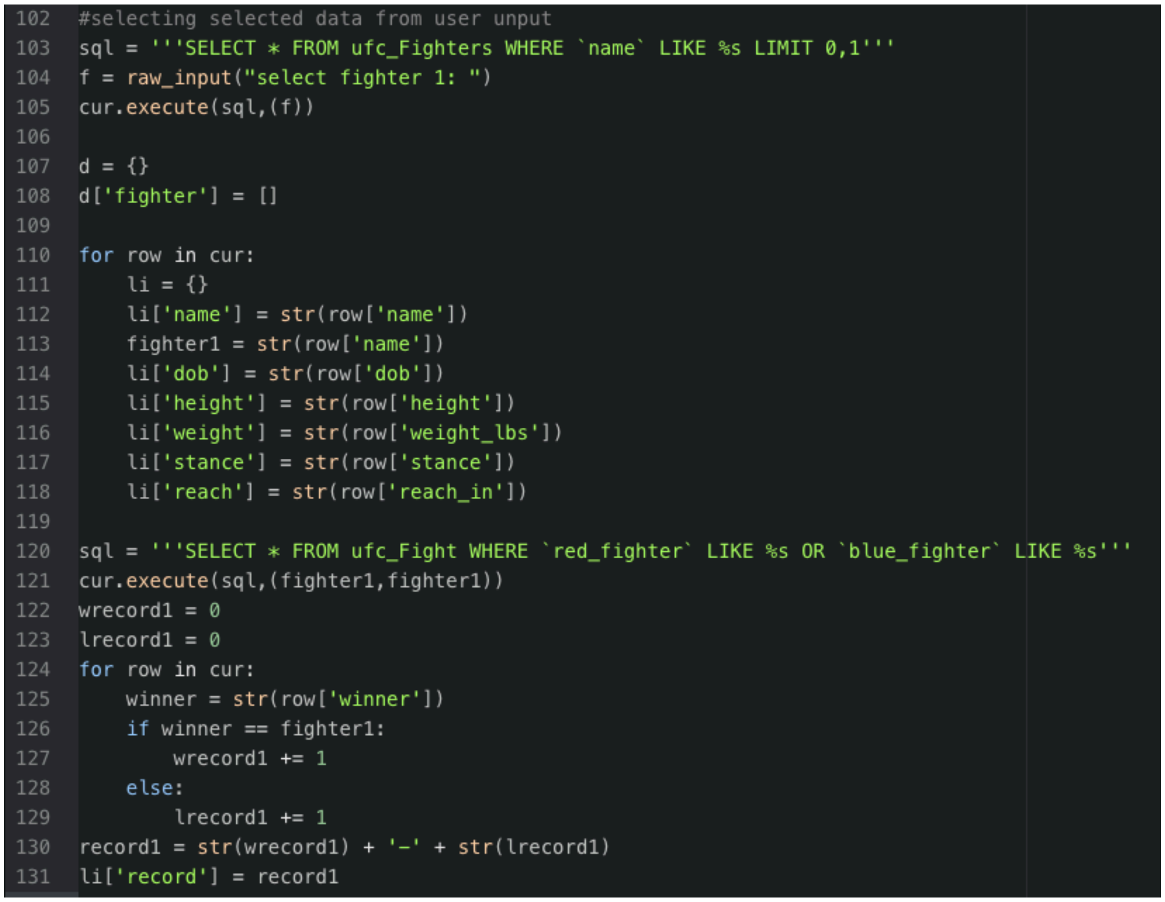
II: Formatting (JSON)

Once the tables are created and the data is inserted, the information can be recalled for the purpose of the application. The user would type in two names and use that raw input as tokens for sql statements in order to gather the wanted information. The output of the data is in JSON. Figure 2.1 below shows an example of the output of the application.



```
[cdeparto@Christians-MacBook-Pro is426project % python ufc.py
select fighter 1: henry cejudo
select fighter 2: tony ferguson
{
 "fighter": [
  {
   "name": "Henry Cejudo",
   "stance": "Orthodox",
   "dob": "Feb 09, 1987",
   "reach": "64",
   "weight": "135",
   "height": "5'4\"",
   "record": "9-2",
   "rpi": "0.66645021645"
  },
  {
   "name": "Tony Ferguson",
   "stance": "Orthodox",
   "dob": "Feb 12, 1984",
   "reach": "76",
   "weight": "155",
   "height": "5'11\"",
   "record": "15-1",
   "rpi": "0.696279761905"
  }
 ]
}
```

Figure 2.1 – sample output of application



```
102 #selecting selected data from user unput
103 sql = '''SELECT * FROM ufc_Fighters WHERE `name` LIKE %s LIMIT 0,1'''
104 f = raw_input("select fighter 1: ")
105 cur.execute(sql,(f))
106 
107 d = {}
108 d['fighter'] = []
109 
110 for row in cur:
111     li = {}
112     li['name'] = str(row['name'])
113     fighter1 = str(row['name'])
114     li['dob'] = str(row['dob'])
115     li['height'] = str(row['height'])
116     li['weight'] = str(row['weight_lbs'])
117     li['stance'] = str(row['stance'])
118     li['reach'] = str(row['reach_in'])
119 
120 sql = '''SELECT * FROM ufc_Fight WHERE `red_fighter` LIKE %s OR `blue_fighter` LIKE %s'''
121 cur.execute(sql,(fighter1,fighter1))
122 wrecord1 = 0
123 lrecord1 = 0
124 for row in cur:
125     winner = str(row['winner'])
126     if winner == fighter1:
127         wrecord1 += 1
128     else:
129         lrecord1 += 1
130 record1 = str(wrecord1) + '-' + str(lrecord1)
131 li['record'] = record1
```

Figure 2.2 – code for acquiring data for every field except for 'rpi'

Code Figure 2.2:

- Lines 102 – 105 is selecting data from the ufc_fighters table where the name is what the user types in
- Lines 107-118 is converting the string of our selected data and transferring it to our new dictionary
- Lines 120-131 to create our win record we need to select every fight our selected fighter partaken in and if the winner column matches our fighter name we add +1 to win and if not then +1 is added to loss.

III: Calculating RPI

RPI stands for "rating percentage index" and is used to rank a sports team based on a team's wins, losses, and the strength of the schedule. This is going to be the indicator of which fighter is most likely to be victor if two were to face against each other (whoever has a higher rpi). It is broken down into three parts:

$$RPI = (WP * 0.25) + (OWP * 0.50) + (OOWP * 0.25)$$

WP: winning percentage (wins / games played)
OWP: the average of the WP for each of the selected team's opponents
OOWP: the average of each opponent's OWP

For the program, WP can be calculated simply by our created record field. We take our 'wrecord#' variable divided by the combination of our wins and losses to determine a percentage.

For OWP, we select each opponent by creating two sql statements; if our fighter is in the red_fighter column then we select the opponent fighter in the blue_fighter column, and vice versa. Then we determine their records and find the win percentage.

For OOWP, we do the same thing as OWP except for each of the opponents of our originally selected fighter.

To solve and find these variable, tiers of loops needed to be made to acquire all of the records and you had to account for both the red_fighter and blue_figher columns. The following structure shows how it would fuction:

- If original fighter (red_fighter)
    - If opponent (red_fighter)
        - If opponent opponent (red_fighter)
        - If opponent opponent (blue_fighter
    - If opponent(blue_fighter)
        - If opponent opponent (red_fighter)
        - If opponent opponent (blue_fighter

This entire process you would have to do twice since there are two fighters selected through raw_input.

IV: Notes

- The application could be broken into two; one for the data transformation of the csv files into the database tables (along with creation of tables if choose to), and for the recall of data from selecting two fighters, doing the calculations, and presenting the data in a json format.
- OWP and OOWP are not 100% accurate. They call for the removal of events where the team in question is involved in for calculation, in this case they are not. Also, both are averages of each individual win percentage, in this case it is the average of the overall record. For purposes of this application since all available data such as type of win (knockout, decision), % attacks landed vs attempted, how many rounds lasted, etc. are not considered, this is okay since the purpose is to find a rough estimate of who would be more likely the victor in a matchup.
- The application allows for both men and women to be compared against one another which would also mean the RPI would not be accurate due to differences in physicality.
- This application is a good starting point to expand from to really spend time and go into significantly more detail and can do a lot of fun things with, such as creating a flask application where we can use html, css, and js to create a ui for people to experience the data in a much more appealing way.