UDACITY

PROJECT

Logs Analysis

A part of the Full Stack Web Developer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 11

NOTES

reporting.py 8

```
1 import psycopg2
```

REQUIRED

Be sure to include a shebang line as the very first line of your Python code file.

From http://stanford.edu/~mayai/basics.py:

When you execute a file from the shell, and the first line in this file is a "shebang" line, the shell tries to run the file using the command specified on the shebang line. The ! is called the "bang". Sometimes the "shebang" line is also called the "hashbang". The hash character (#) is used because it defines a comment in most scripting languages, so the shebang line will be ignored by the scripting language by default.

The shebang line was invented because scripts are not compiled, so they are not executable files, but people still want to "run" them. The shebang line specifies exactly how to run a script. Suppose this is your shebang line:

```
#!/usr/bin/env python3
```

What it means is that, when I type in ./reporting.py, the shell invokes the Python 3 interpreter and will actually run the following command:

```
/usr/bin/env python3 reporting.py
```

```
2
3 DB_NAME = 'news'
4 DB_USER = 'vagrant'
5
6 def get_most_popular_articles():
7     """gets the most popular three articles of all time, as a sorted list with the most popular article at the top."""
8     # Connect to an existing database
9     conn = psycopg2.connect("dbname=%s user=%s" % (DB_NAME, DB_USER))
10    # Open a cursor to perform database operations
11    cursor = conn.cursor()
12    # Query the database and obtain data as Python objects
13    cursor.execute("select articles.title, count(*) as views from articles left join log"
14                   " on concat('/article/',articles.slug)=log.path where path like '/article/%'"
15                   " group by log.path, articles.title order by views desc limit 3;")
```

REQUIRED

Because the pep8 style tool issues an error message when it encounters a line longer than 79 characters, you will need to break this SELECT statement into more, shorter lines. There are several ways of doing this, here is one example that can replace lines 13 - 15:

```
cursor.execute("""
    select articles.title, count(*) as views
    from articles left join log
    on concat('/article/',articles.slug)=log.path
    where path like '/article/%'
    group by log.path, articles.title order by views desc limit 3;
    """)
```

```
16    popular_articles = cursor.fetchall()
17    # Close communication with the database
18    cursor.close()
19    conn.close()
20    return popular_articles
21
22
23 def get_most_popular_authors():
```

```
24      """gets the authors with the most page views, when you sum up all of the articles each author has written.
25      givess a sorted list with the most popular author at the top."""
26      # Connect to an existing database
27      conn = psycopg2.connect("dbname=%s user=%s" % (DB_NAME, DB_USER))
28      # Open a cursor to perform database operations
29      cursor = conn.cursor()
30      # Query the database and obtain data as Python objects
31      cursor.execute("select authors.name, authors_score.views from authors left join authors_score"
32                     " on authors.id=authors_score.author_id order by views desc;")
33      popular_authors = cursor.fetchall()
34      # Close communication with the database
35      cursor.close()
36      conn.close()
37      return popular_authors
38
39
40  def get_most_popular_authors():
```

REQUIRED

Looks like you've repeated the get_most_popular_authors() function definition. Be sure to remove the duplicate.

```
41      """gets the authors with the most page views, when you sum up all of the articles each author has written.
42      givess a sorted list with the most popular author at the top."""
43      # Connect to an existing database
44      conn = psycopg2.connect("dbname=%s user=%s" % (DB_NAME, DB_USER))
45      # Open a cursor to perform database operations
46      cursor = conn.cursor()
47      # Query the database and obtain data as Python objects
48      cursor.execute("select authors.name, authors_score.views from authors left join authors_score"
49                     " on authors.id=authors_score.author_id order by views desc;")
50      popular_authors = cursor.fetchall()
51      # Close communication with the database
52      cursor.close()
53      conn.close()
54      return popular_authors
55
56
57  def get_down_times():
58      """gets the days when more than 1% of requests lead to errors"""
59      # Connect to an existing database
60      conn = psycopg2.connect("dbname=%s user=%s" % (DB_NAME, DB_USER))
61      # Open a cursor to perform database operations
62      cursor = conn.cursor()
63      # Query the database and obtain data as Python objects
64      cursor.execute("select * from error_rates where error_rate>1.0")
65      down_times = cursor.fetchall()
66      # Close communication with the database
67      cursor.close()
68      conn.close()
69      return down_times
```

AWESOME

All of the SQL statements used in your project run efficiently and return the correct results.

```
70
71
72  def write_article_info(articles):
73      """writes article-related information in report-file.txt"""
74      report_file = open("report-file.txt", "a")
```

SUGGESTION

I notice that all of the calls to open() are in append mode ("a"). I was going to suggest that here on line 74 you change the mode to "w" to overwrite any file that may already be there, but then I was thinking you may have written your code to be imported, so that you could call any of the functions separate from the others. Since the project specifications don't discuss this, I'll leave it up to you whether to change this or not.

```
75      report_file.write("MOST POPULAR ARTICLES:\n")
76
77      for article in articles:
78          report_file.write("\"")
79          report_file.write(article[0])
80          report_file.write("\" : ")
81          report_file.write(str(article[1]))
82          report_file.write(" views\n")
83
84      report_file.write("\n\n\n")
85      report_file.close()
86
87
88  def write_author_info(authors):
89      """writes author-related information in report-file.txt"""
90      report_file = open("report-file.txt", "a")
91      report_file.write("MOST POPULAR AUTHORS:\n")
92
93      for author in authors:
94          report_file.write(author[0])
95          report_file.write(" : ")
```

```
 96         report_file.write(str(author[1]))
 97         report_file.write(" views\n")
 98
 99     report_file.write("\n\n\n")
100     report_file.close()
101
102 def write_down_time_info(down_times):
103     """writes error-rate information in report-file.txt"""
104     report_file = open("report-file.txt", "a")
105     report_file.write("DOWN TIME INFO:\n")
106
107     for down_time in down_times:
108         report_file.write(down_time[0])
109         report_file.write(" : ")
110         report_file.write(str(down_time[1]))
111         report_file.write("% errors\n")
112
113     report_file.write("\n\n\n")
114     report_file.close()
115
116
117 popular_articles = get_most_popular_articles()
118 write_article_info(popular_articles)
```

SUGGESTION

Or, you could combine the two lines into one:

```
write_article_info(get_most_popular_articles())
```

This would also apply to lines 120 - 124 below.

```
119
120 popular_authors = get_most_popular_authors()
121 write_author_info(popular_authors)
122
123 down_times = get_down_times()
124 write_down_time_info(down_times)
```

SUGGESTION

In fact, you could write lines 117 - 124 as follows:

```
if __name__ == "__main__":
    write_article_info(get_most_popular_articles())
    write_author_info(get_most_popular_authors())
    write_down_time_info(get_down_times())
```

This would help permit your code to be imported by another program. One of the other reviewers gets credit for this suggestion, also for informing me about a good StackOverflow discussion on the subject:

https://stackoverflow.com/questions/419163/what-does-if-name-main-do

With that in place, from a Python prompt you could enter the following:

```
>>> import reporting as r
>>> r.write_article_info(r.get_most_popular_authors())
>>> quit()
```

and a file containing the information for this section will be created (or appended to an existing file).

```
125
126
127 # create view authors_score as select articles.author as author_id, count(*) as views from articles left join log on concat('
128 # create view request_counts as select date_trunc('day', time) as date, count(*) as request_count from log group by date;
129 # create view error_counts as select date_trunc('day', time) as date, count(*) as error_count from log where status not like
130 # create view error_rates as select to_char(request_counts.date, 'FMMonth FMDDth, YYYY') as date, round((error_counts.error_c
```

REQUIRED

Since you've included the CREATE VIEW definitions in your README file, you don't need to list them here, even as comments. You can assume that the user will be responsible for creating the views by extracting their definitions from the README file, before attempting to run your Python code.

report-file.txt 2

README.md 1

RETURN TO PATH

Rate this review

Student FAQ Reviewer Agreement