eDreams ODIGEO

Hello,

On the following pages, you will find 3 sections with some exercises related to different types of work that this job position may require.

- The first section is a case study where we will validate your skills on developing a software application. This exercise is optional, we will positively evaluate if you perform it ;
- The second section combines a set of quick exercises focused on SQL ;
- The third section is a case study where you should perform a deep analysis on two different data sources.

We encourage you to start with the sections that you feel the most comfortable with.

Please submit your solution in a zipped file via email to sem.caps@edreamsodigeo.com.

Good luck!

CASE STUDY 1 - Movies Data

Description

We want you to implement a REST API that, given a movie or series title, returns the results available in the OMDb API, which is a RESTful web service to obtain that information.

We will positively evaluate it if you develop it in Python.

Data structure

We want you to get the following information of the obtained results:

- Movies: title, year, released date, runtime, genre, plot, box office, IMDb rating and IMDb votes.
- Series: title, year, released date, runtime, genre, plot, total seasons, IMDb rating and IMDb votes.

Constraints

- The amount of displayed results should be equal or less than 10. If not, your REST API should return an error. Either way, keep in mind that this constraint could be changed or removed in the future.

- The displayed results must be ordered by IMDb rating, IMDb votes and title.

- We don't know if the OMDb API will be free in the future, so we would like you to implement a cache layer of these movies and/or series in order to avoid hitting it twice for the same title on the same date.

Example

- Request: “Dragon Ball”.
- Response:

```
{
        "series": {
                "title": "Dragon Ball Z",
                "year": "1996-2003",
                "released": "13 Sep 1996",
                "runtime": "24 min",
                "genre": "[...]",
                "plot": "[...]",
                "totalSeasons": 15,
                "imdbRating": 8.7,
                "imdbVotes": 111597
        },
        [...],
        "movie": {
                "title": "Dragon Ball Z: Battle of Gods",
                "year": "2013",
                "released": "05 Aug 2014",
```

```
                "runtime": "85 min",
                "genre": "[...]",
                "plot": "[...]",
                "boxOffice": "$2,553,002",
                "imdbRating": 7.2,
                "imdbVotes": 26163
        }
}
```


Notes

- We can not tell you how much will grow this application in the future, so its design is very important and you must assume that you're going to maintain and evolve it for years.
- Also, we would like you to separate the application in layers. For example, that would allow you to quickly change its access (HTTP server, command line, etc.).
- You should consider implementing some automated testing.

Instructions

- OMDb API reference: http://www.omdbapi.com/.
- Please generate your API Key through this form (choose the free account type).
- You can test it through HTTP access. For example:
    - http://www.omdbapi.com/?apikey=YOUR_API_KEY&s=dragon%20ball
    - http://www.omdbapi.com/?apikey=YOUR_API_KEY&i=tt0214341
- Send us back your solution in any format you wish (a zip file via email or cloud service, a hosting link for software development as GitHub, etc.).
- Please share with us any comments we would need to know in order to run your application.

CASE STUDY 2 - Customers Insights

Datasets

You will need to use the file "SQL data.xlsx" that has been shared with you.

That file contains 2 different datasets:

- *FunnelTable*
- *BookingTable*

The **FunnelTable** dataset represents some example navigation on our website on a given day. Each row is a pageview and its columns are the following ones:

- **userid**: a random string that uniquely identifies each user.
- **sessionid**: a random string that uniquely identifies each of a user's sessions. It's unique within a user but not unique across all of them.
- **timestamp**: the timestamp (YYYY-MM-DD hh:mm:ss) of the moment when the user viewed the page.
- **page**: the name of the page.
- **channel**: how the user arrived at our website. For instance, "Email" means the user clicked on a link in a promotional email.
- **device**: the device type used by the customer (desktop, mobile or tablet).
- **browser**: the browser used by the customer.
- **country**: the country of the customer.
- **transactionid**: a universally unique identifier for each transaction.

The **BookingTable** dataset only contains information about the final transactions performed by the users that come to our website. Its columns are the following ones:

- **transactionid**: a universally unique identifier for each transaction.
- **product**: the product that the customer bought.
- **revenue**: the amount of money (EUR) that the company made with the transaction.

Notes

The navigation flow on our website is:

- A user reaches our website. The first page can be:
    - "**home**": the homepage.
    - "**black friday**": a special web page containing some Black Friday offers.
    - "**christmas offers**": a special web page containing some Christmas offers.

- After that, the user launches a search, and reaches the "**results**" page. If they find a search result of their interest, they select it and progress forward.

- An optional page, called "**upsell**", appears, which tries to convince the user to add a business class seat to their purchase. This page does not always appear.

- Then comes the "**payment**" page, where the user input their details and payment info.

- Finally, if the payment succeeds, the transaction completes and the "**confirmation**" page appears.

Summarizing: home/black friday/christmas offers → results → (upsell) → payment → confirmation.

Constraints

A purchase is completed when the customer reaches the "confirmation" page - thus, the transactionid field is only defined there and empty elsewhere.

Questions

For each of the questions below, provide a **SQL query** that answers them:

1. What is the total revenue generated per each country AND device the 2nd of November?
2. What product from Paid Search channel has the highest average revenue per transaction?
3. How many users visited us between 21:00 and 22:00 on the 2nd of November?
4. What country has the most pageviews per session on average?
5. How many users made more than one transaction?
6. How many sessions have seen the "home" page AND the "payment" page?
7. We define "continuance rate" as the percentage of sessions that have progressed from payment to confirmation, i.e. (sessions that have seen confirmation) / (sessions that have seen payment). Compute it by country AND device.
8. We define "landing page" as the first page in a session. What is the most abundant landing page?
9. We define conversion rate as (total transactions) / (total sessions). What is the conversion rate per landing page AND device?
10. We define "Exit rate" as the percentage of times a page was the last one in a session, out of all the times the page was viewed. What is the exit rate of "results"?

CASE STUDY 3 - Airline Popularity on Twitter

Datasets

You will need to use the following two files that we also shared with you:

- *Tweets.csv*
- *Delay_Airlines.csv*

The first dataset contains Twitter data of 6 major US Airline companies: American, Delta, Virgin America, Unaited, US Airways and SouthWest. That data corresponds to February 2015 and contains a sentiment classification based on the tweet content. These are the columns of the dataset:

- **tweet_id**: a identifier for each tweet.
- **airline_sentiment**: the sentiment nature of the tweet (positive, neutral or negative).
- **negativereason**: the reason why the tweet is negative (i.e. flight delay).
- **airline**: the airline about the tweet was written.
- **name**: the name of the user.
- **retweet_count**: the number of retweets.
- **text**: the full text of the tweet.
- **tweet_created**: timestamp creation of the tweet.
- **tweet_location**: the location from which the tweet was sent.

The second dataset comes from the database *Bureau of Transportation Statistics*. In this link you will find detailed information of the columns' content.

Questions

1. Drive a descriptive analysis of the *Tweets* dataset that would allow us to:
    - Understand and compare the popularity of the different airlines. Could you extract any conclusion?
    - Analyze the tweets distribution based on the type of sentiment and the day of the week. Could you extract any conclusion?
    - Analyze the tweets distribution based on the type of sentiment and the hour of the day. Could you extract any conclusion?

2. Merge the *Tweets* dataset with the *Airline_Delay* one by the date and time of the tweets creation, and analyze if there is any correlation between tweets and cancelled flights.

Please send us the code that you've done in order to extract your conclusions and a short report explaining the methodology used and the obtained results.

We will positively evaluate it if you perform it in Python.