Project - 1 Reflection

PostGRESQL Reflection :

- **What is the need for Add Ons in Heroku?**

  Add-ons are cloud services that extend Heroku apps with useful features and services, such as data storage, monitoring, analytics, data processing, and more.
  These are fully maintained for you by either a third-party provider or by Heroku.
  Add-ons exist so that developers can focus on their own application logic, and not the additional complexity of keeping supporting services running at full production capacity.
  An add-on interacts with a Heroku app in one or more of the following ways:
  - It sets one or more config vars in the app with values necessary to communicate with the add-on. These values typically include the cloud service's URL and any credentials necessary to access that URL.
  - It reads or writes to the app's logs.
  - It uses the Heroku Platform API for Partners to perform app management actions (such as dyno scaling) on behalf of the app's developer.

- **What exactly happens when you click on provision while configuring the Postgres addon?**

  When we click on Provision while configuring Postgres addon, the addon will be installed and you will be having a working Postgres database. It provides us with 10000 rows which are free to use. We

can get credentials from the settings and these are used to login on Adminer.

- **What is the use of Adminer? How does it work?**

  Adminer is a tool similar to phpMyAdmin. It is used to run SQL commands on a GUI interface. We can access the databases via Adminer by logging into it using credentials from the Heroku Credentials page.

Python and Flask Reflection :

- **How do I manage to use python 3.6 if I already have python 2.7?**

  While Installation of Python3, if Python 2.7 is already on the Working system, Select custom installation and choose a different location to install it.
  Add path to environmental variables in the System.
  In order to execute both python versions change the python.exe to python2.exe where python2 is located and python.exe to python3.exe where python3 is located.
  Run Python2 –V and Python3 –V to get respective python versions.
  If the above commands run without problems python2 and python3 were successfully installed on the environment.

- **What is the role of pip and how does it work?**

  PIP is a package manager for Python packages, or modules if you like. PIP helps you in downloading, removing and listing all the packages.

- **What is the role of requirements.txt and how does it work with pip?**

Requirements are very simple: lists of packages to install.
Instead of running something like pip install MyApp and getting whatever libraries come along, you can create a requirements file.
If you save this in requirements.txt, then you can pip install -r requirements.txt
The requirements file is a way to get pip to install specific packages to make up an environment. This document describes that format.

- **Which packages are installed and why are they required?**

  The packages that are installed are : Flask, Flask-Session, psycopg2-binary, SQLAlchemy.
  Flask is a web framework. This means flask provides you with tools, libraries, and technologies that allow you to build a web application.

  Session is the time interval when a client logs into a server and logs out of it. The data, which is needed to be held across this session, is stored in the client browser. A session with each client is assigned a Session ID.

  Psycopg is the most popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several threads can share the same connection).

  SQLAlchemy is a library that facilitates the communication between Python programs and databases. Most of the times, this library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements

- **Which environment variables set for Flask to work? What is the purpose of each variable?**

  We use the export command to set environment variables.
  - We set the FLASK_APP variable to tell the start point of the application i.e what file to run after the run command is used.
  - We set the FLASK_DEBUG to 1 to automatically apply a change to the web app when the file is changed or modified.
  - Set up the DATABASE_URL to establish the communication between the flask application and database.

- **What happens when the Flask run command is issued on the terminal?**

  When Flask run Command is issued on the terminal, it launches a very simple builtin server, which is good enough for testing but probably not what you want to use in production.

- **On which port is Flask running and can it be changed?**

  Flask generally runs on the port number 5000. Yes, it can be changed, but it is not recommended.

- **How is Flask different from the tiny web server?**

  Flask is a micro web application framework. That means it is basically a set of tools and libraries that make it easier to build web applications in Python.
  Flask does however include a web server that can be used for testing and development. But when you're ready to host your app or put it into production, you should choose a different web server to use with it. Flask also enables you to choose the URI, whereas in webserver you have to parse the URI.

Goodreads API - Reflection :

- **What are the various categories of web APIs available on good reads?**

  The various categories of web APIs available on good reads are :
  - book.isbn_to_id — Get Goodreads book IDs given ISBNs.
  - book.id_to_work_id — Get Goodreads work IDs given Goodreads book IDs.
  - book.review_counts — Get review statistics given a list of ISBNs.
  - book.show — Get the reviews for a book given a Goodreads book id.
  - book.show_by_isbn — Get the reviews for a book given an ISBN.
  - book.title — Get the reviews for a book given a title string.

- **Is there a limit on the use of the web API? What are the limits?**

  Yes, GoodReads has a limit to Not request any method more than once a second.
  By default, the number of requests is set to 100 requests per 100 seconds per user and can be adjusted to a maximum value of 1,000. But the number of requests to the API is restricted to a maximum of 10 requests per second per user.