Find Your Destination

Solito Reyes III
Kasia Kalemba
Sandra Froonjian
Shrilekha Vijayakanthan

Motivation

Americans are currently not allowed to travel outside of the US

Solution

WHY NOT FIND YOUR PERFECT DESTINATION IN YOUR OWN COUNTRY?

OUR INTERACTIVE PAGE WILL HELP YOU DECIDE WHERE TO GO!

Behind the scenes

- Interactive Webpage
  - Data
    - Inspecting, Scraping & Cleaning
      - Python in Jupyter notebook
    - Storing
      - Postgres Database
        - Heroku
    - Retrieving
      - Flask
  - Visualizations
    - Web Page Design
      - HTML & CSS
      - Bootstrap
    - Interaction
      - Javascript
      - D3
    - Graphs
      - MapBox & Leaflet
      - Frappe Charts

kaggle

WIKIPEDIA The Free Encyclopedia

data.world

LET'S GO THROUGH OUR PROCESS

Step 1: Get data, clean it, upload it to a database

- We gathered data for the following categories for each US state:
  - Average airfare & Count of passengers: *Bureau of Transportation Statistics*
  - Hotel reviews: *Kaggle*
  - Various attractions number and location: *Wikipedia*
  - Campsite locations: *DataWorld*
  - Value of your $: *US News*
- **We wanted to focus on the things that make us choose a destination: things to do and value of money**

Step 1: Get data, clean it, upload it to a database

- Using **Pandas** in **Jupyter Notebook,** we inspected the data, cleaned it up and scraped using **Chromedriver** from sources that did not provide downloadable files

Scraping with Pandas

In [6]:
```
import pandas as pd
from sqlalchemy import create_engine
from sqlalchemy.dialects.postgresql import insert
from sqlalchemy import table, column
```

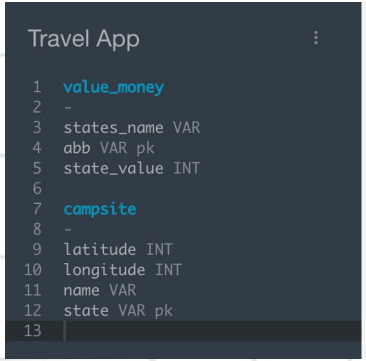
In [7]:
```
url1 = 'https://en.wikipedia.org/wiki/List_of_casinos_in_the_United_States#State'
tables = pd.read_html(url1)
casinos_df = tables[0]
casinos_df=casinos_df[["Casino","State"]]
casinos_df=casinos_df.rename(columns = {'Casino':'casino', 'State':'state'}, inplace = False)
casinos_df.index.name='casino_id'
casinos_df['state']=casinos_df['state'].str.strip()
casinos_df
```

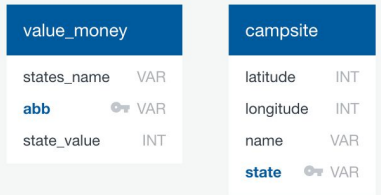
Out[7]:

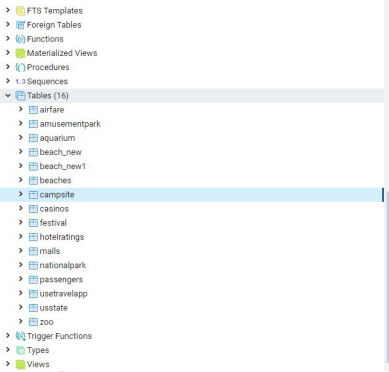
casino_id	casino	state
0	Victoryland	Alabama
1	Wind Creek Casino & Hotel Atmore	Alabama
2	Wind Creek Casino & Hotel Montgomery	Alabama
3	Wind Creek Casino & Hotel Wetumpka	Alabama
4	Apache Gold Casino Resort	Arizona
...	...	...
1006	St. Croix Casino	Wisconsin

Step 1: Get data, clean it, upload it to a database

- Using **ERDs**, set up the schema for individual tables
- Uploaded data from data frame into Heroku **Postgres** database using **SQL-Alchemy 16 Tables, 10000 rows of data**








```
1  -- Table: public.campsite
2
3  -- DROP TABLE public.campsite;
4
5  CREATE TABLE public.campsite
6  (
7      camp_id bigint NOT NULL,
8      name character varying(1000) COLLATE pg_catalog."default",
9      state character varying(200) COLLATE pg_catalog."default",
10     CONSTRAINT campsite_pkey PRIMARY KEY (camp_id)
11 )
12 WITH (
13     OIDS = FALSE
14 )
15 TABLESPACE pg_default;
16
17 ALTER TABLE public.campsite
18     OWNER to postgres;
```

Our ranking equation for the best state to visit

- Created a consolidated table with all the data in order to make an overall ranking system for the best state to visit:
  - Flights: 1
  - Passenger volume: 1
  - Value of Dollar: 1
  - Hotel Ratings: 1
  - Casinos: 1
  - National Parks: 1
  - Aquariums: 1
  - Zoos: 1
  - Festivals: 1
  - Amusement Parks: 1
  - Beaches: 1
  - Malls: 0.5
  - Campsites: 0.1

Data output Explain Messages Notifications

	state [PK] character varying (100)	abbr text	states_value double precision	Latitude double precision	Longitude double precision	count_amusement_park bigint	count_aquarium bigint	count_beach bigint	count_casino bigint	count_festival bigint	count_hotelratings integer	count_mall bigint	count_parks bigint	count_campsite bigint	count_zoo bigint	total_count numeric	airfare_rank bigint	passenger_rank bigint	rank_number bigint
1	California	ca	0.88	36.7783	-119.4179	36	22	46	157	18	12	87	8	1301	25	638.1	44	53	1
2	Florida	fl	1.01	27.9944	-81.7603	31	20	85	37	5	16	59	3	180	31	403.0	46	52	2
3	Nevada	nv	1.02	39.876	-117.2241	5	4	5	240	1	4	12	1	98	2	362.8	34	27	3
4	New Jersey	nj	0.88	39.8329	-74.8718	18	6	50	18	4	50	53	0	26	7	300.6	53	42	4
5	New York	ny	0.87	43	-75	19	12	18	25	14	3	72	0	149	19	282.9	38	50	5
6	Washington	wa	0.95	47.75	-120.74	7	8	4	52	3	13	26	3	576	8	269.6	41	47	6
7	Texas	tx	1.03	31	-100	23	16	5	3	11	7	51	2	486	26	265.6	22	51	7
8	Illinois	il	1	40	-89	11	2	0	10	5	15	73	0	162	14	237.3	42	49	8
9	Ohio	oh	1.12	40.3675	-82.9962	18	3	3	11	19	3	54	1	114	15	226.4	49	40	9
10	Massachusetts	ma	0.94	42.4072	-71.3824	5	0	22	3	3	28	27	0	36	13	185.5	29	44	10
11	North Carolina	nc	1.1	35.7822	-80.7935	8	0	30	3	10	3	36	1	105	13	195.5	39	43	11
12	Michigan	mi	1.07	44.1822	-84.5368	7	3	7	25	7	2	25	1	488	12	186.8	11	38	12
13	Oregon	or	1.01	44	-120.5	2	5	12	11	5	3	0	1	720	8	183.0	31	33	13
14	Colorado	co	0.97	39.113	-105.3589	12	0	0	42	1	3	14	4	507	0	180.7	6	48	14
15	Arizona	az	1.04	34.0489	-111.0937	5	4	0	28	1	3	33	3	259	9	177.9	37	41	15
16	Minnesota	mn	1.03	46.3924	-94.6363	5	4	0	20	5	2	19	1	392	9	166.2	26	36	16
17	Missouri	mo	1.12	37.9	-91.83	9	5	0	14	3	2	30	0	189	9	165.9	40	35	17
18	Maryland	md	0.91	39.0458	-76.6413	9	10	2	6	3	2	26	0	48	5	159.6	50	32	18
19	Utah	ut	1.03	39.4192	-111.9507	4	3	0	0	7	5	15	5	260	4	157.0	48	30	19
20	Hawaii	hi	0.84	19.7618	-155.5828	1	4	29	0	1	18	11	2	36	2	154.8	52	31	20
21	Pennsylvania	pa	1.02	41.2033	-77.1945	17	3	1	12	7	5	0	0	112	20	154.2	22	45	21
22	Oklahoma	ok	1.11	36.0616	-96.9214	4	2	0	63	0	2	13	0	281	3	154.1	20	23	22
23	Wisconsin	wi	1.07	44.5	-89.5	8	2	0	22	2	2	22	0	309	22	151.9	14	27	23
24	Tennessee	tn	1.11	35.8601	-86.6602	16	5	0	0	4	2	30	1	135	7	148.5	30	34	24
25	Georgia	ga	1.08	33.3479	-83.4013	10	3	3	0	2	12	33	0	179	8	141.9	7	46	25
26	Indiana	in	1.1	40.2735	-86.127	8	1	0	14	8	2	29	0	111	9	140.1	33	28	26
27	Louisiana	la	1.1	30.3916	-92.3291	3	3	4	28	6	23	16	0	89	9	139.8	10	29	27
28	Virginia	va	0.98	37.9269	-78.0249	6	1	6	0	2	2	41	1	105	16	130.5	12	26	28
29	Iowa	ia	1.11	42.032	-93.5815	4	2	0	30	0	2	15	0	497	2	129.7	16	15	29
30	South Carolina	sc	1.11	33.8361	-81.1637	7	2	11	0	0	4	18	1	71	9	119.1	25	25	30
31	Kentucky	ky	1.13	37.8393	-85.27	3	1	0	0	3	2	21	1	136	5	110.8	28	18	31
32	New Mexico	nm	1.06	34.3071	-106.0181	2	1	1	28	1	3	10	1	212	9	107.2	13	17	32
33	Idaho	id	1.07	44.0682	-114.742	4	2	6	5	0	2	8	1	487	5	102.7	4	16	33
34	Montana	mt	1.05	46.9653	-109.5337	1	0	0	0	0	2	3	2	481	6	94.1	19	13	34
35	Connecticut	ct	0.92	41.6	-72.7	2	0	8	2	1	0	13	0	18	2	91.8	36	22	35

Step 2: Making a flask app

- Once the data was ready to use, we created a **Flask App** so our page could retrieve data from APIs we created for the web page

- Creating a couple routes, we were able to achieve a dynamic page

```
@app.route("/rank")
def rank():
    session = Session(engine)

    ranking = session.query(combined_table.state, combined_table.rank_number).all()
    session.close()

    rank_list = []
    for state_name, rank in ranking:
        rank_list_dict = {}
        rank_list_dict["name"] = state_name
        rank_list_dict["rank"] = rank
        rank_list.append(rank_list_dict)

    return jsonify(rank_list)


# dynamic state page
@app.route("/<state>")
def dynamic(state):
    # Create a session
    session = Session(engine)

    travel_num = session.query(combined_table.state, combined_table.abbr, combined_ta
        filter(combined_table.state == state).all()
    session.close()
```

Step 3: Setting up the webpage

- For the basic front page design, we used **Bootstrap,** an **HTML** library for easy website building
- Added **Javascript** components for an interactive experience
- We used the **Flask App** for data retrieval

```html
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <title>Travel App</title>

  <!-- Leaflet CSS and Script -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css" />
  <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>



  <!-- Our CSS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
  bootstrap.min.css">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/
  font-awesome.min.css">

  <!-- Stylesheets -->
  <link rel="stylesheet" href="../static/css/bootstrap.min.css">
  <link rel="stylesheet" href="../static/css/style.css">


 <!-- Javascript -->
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
    integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
    crossorigin="anonymous"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
    integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
    crossorigin="anonymous"></script>

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
    integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
    crossorigin="anonymous"></script>

      <!-- d3 JavaScript -->
  <script src="https://d3js.org/d3.v4.min.js"></script>

</head>
```

Step 4: Making an interactive map

```
L.geoJson(states, {
  style: function (feature) {
    return {
      fillColor: getColor(feature.properties.name),
      weight: 2,
      opacity: 1,
      color: 'black',
      fillOpacity: 1.0
    };
  },
  onEachFeature: function (feature, layer) {
    layer.on({

      mouseover: function (event) {
        layer = event.target;
        layer.setStyle({
          fillOpacity: .7,
          weight: 3.5
        });
```

```
d3.json(rank_url, function (state_ranks) {
  function getColor(r) {
    var color = '';
    for (var i = 0; i < state_ranks.length; i++) {
      var state_index = state_ranks[i];

      if (r === state_index.name) {
        var rank = +state_index.rank;
        break;
      }

    }

    if (rank > 45) {
      color = color_scale[9]
    } else if (rank > 40) {
      color = color_scale[8]
    } else if (rank > 35) {
      color = color_scale[7]
```

- Our map was set up using **Mapbox**
- Using **Leaflet,** we created a US choropleth based on our ranking of states

```
  mouseout: function (event) {
    layer = event.target;
    layer.setStyle({
      fillOpacity: 1,
      weight: 2
    });
  },

  click: function (event) {
    map.fitBounds(event.target.getBounds()),
      attraction_lists(feature.properties.name);
  }

});
layer.bindPopup("<h1>" + feature.properties.name +
```

Step 5: Making a user interactive dashboard

- For graphs that show types of attractions and miscellaneous data we used a new JS library, **Frappe Charts.**

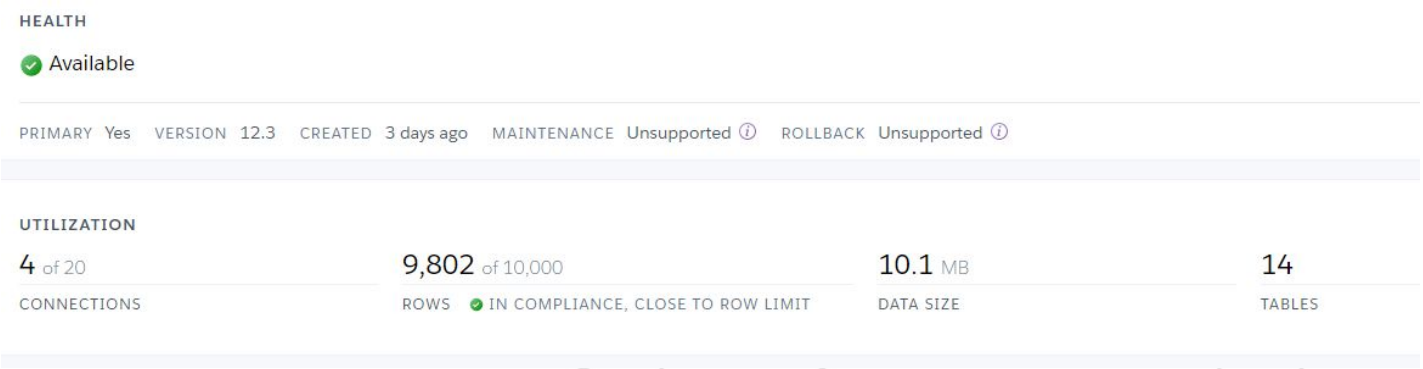
```
amusement_num = Object.values(state_info[0])[2];
aquarium_num = Object.values(state_info[0])[3];
beach_num = Object.values(state_info[0])[4];
campsite_num = Object.values(state_info[0])[5];
casino_num = Object.values(state_info[0])[6];
festival_num = Object.values(state_info[0])[8];
mall_num = Object.values(state_info[0])[10];
park_num = Object.values(state_info[0])[11];
zoo_num = Object.values(state_info[0])[15];

const attraction_data = {
  labels: ["Amusements", "Aquariums", "Beaches", "Campsites",
  datasets: [
    {
      name: "Attractions", type: "pie",
      values: [amusement_num, aquarium_num, beach_num, campsit
    }
  ]
}

const pie_chart = new frappe.Chart(".pie-chart", {
  title: "Distribution of Attractions",
  data: attraction_data,
  type: 'pie',
  height: 500,
})
```

Step 6: Deployment to Heroku

- Lastly, we used the connection between **Postgres** Database to **Heroku** for the final webpage



SO, WHERE DO YOU WANT TO GO?

LET'S SEE THE FINAL RESULT

https://findyourdestinations.herokuapp.com/