F1 Dashboard

Zouheir ElHalabi, Krystal Davis,
Brian Knorr, Kashish Misbah

Choosing Formula 1

Our Reasoning:

- Two different ways in which data can be accessed
  - An API at http://ergast.com/mrd/
  - A set of CSVs from Kaggle
- More than 100 records readily available with plenty of information that can be manipulated and stored in databases
  - The records available go back to 1950 and include a large amount of data on each race, driver, and circuit
- The data is diverse enough to be displayed in various formats
  - Circuit locations can be displayed in Leaflet
  - Race information can be displayed with plot.ly

Coding Approach - Leverage the homeworks

1 **SQLAlchemy**
HW:10 - Advanced Data Storage and Retrieval: Creating an API (that we can later call upon).

2 **Web Design**
HW:11 - Web Visualization Dashboard: Using html and css to design a dashboard.

3 **Web Scraping**
HW:12 - Python Flask and mongo-db.

4 **Postgres Database**
Same concepts as Project 2. Convert CSVs, manipulate, upload, and query

5 **Visuals using plot.ly**
HW:14 - Building interactive plots

6 **JavaScript, Leaflet**
HW:15 - Using leaflet to display data.

Data Manipulation

1 SQLAlchemy

Create an API to connect to later to retrieve data from PgAdmin

```
@app.route('/avg-lap-time-per-driver/<year>/<first_name>/<last_name>')
def avg_lap_time_per_driver(year, first_name, last_name):
    session = Session(engine)
    q = f"SELECT l.race_id, l.avg_lap_time, l.driver_id, d.first_name, d.last_name, r.year, r.round \
            FROM lap_time_avg_per_driver AS l \
                JOIN races AS r ON r.race_id=l.race_id  \
                    JOIN drivers AS d ON l.driver_id=d.driver_id  \
                        WHERE (year={year} AND last_name='{last_name}' AND first_name='{first_name}')\
                            ORDER BY (r.year,r.round)"
    results = pd.read_sql(q,engine)
    results = results.drop(["race_id", "driver_id", "year", "round", "first_name", "last_name"], axis = 1)
    datadict = results.to_dict('records')
    session.close()
    return jsonify(datadict)
```

```
lap_query = "http://127.0.0.1:5000/avg-lap-time-per-driver/" + year + "/" + first_name + "/" + last_name;
d3.json(lap_query).then(function (lap_data) {

  avg_lap_query = "http://127.0.0.1:5000/avg-lap-time/" + year;
  d3.json(avg_lap_query).then(function (ave_lap_data) {

    //Get API data into usable list
    ave_lap_times = [];
    for (let i in ave_lap_data){
      ave_lap_times.push(ave_lap_data[i].avg_lap_time / 1000);
    }
```

Data Manipulation

2 Pandas

Used pandas to import scraped CSVs, edit the column names for easier SQL use, and export into PgAdmin

```
# Import CSV with drivers data
csv_file = "Resources/drivers.csv"
drivers_df = pd.read_csv(csv_file)
drivers_df.head()
```

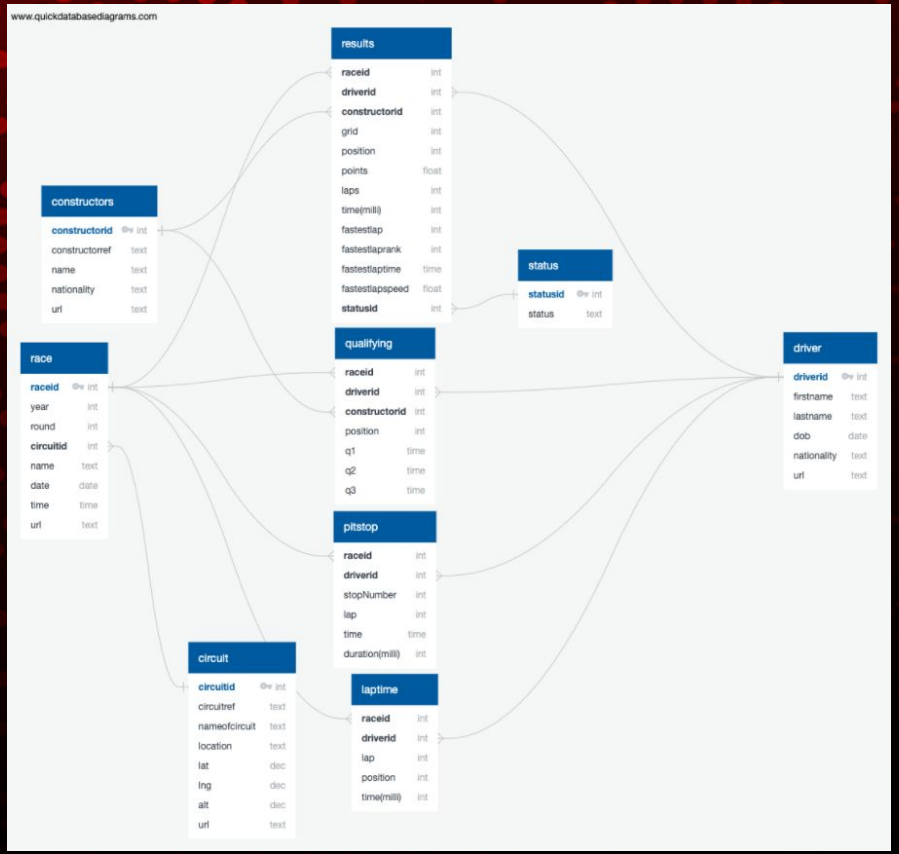
	driverId	driverRef	number	code	forename	surname	dob
0	1	hamilton	44	HAM	Lewis	Hamilton	1985-01-07
1	2	heidfeld	\N	HEI	Nick	Heidfeld	1977-05-10

```
#transform drivers csv
drivers_df = drivers_df.rename(columns={
"driverRef": "driver_ref",
"driverId": "driver_id",
"forename": "first_name",
"surname": "last_name"})
drivers_df.head()
```

	driver_id	driver_ref	number	code	first_name	last_name	dob
0	1	hamilton	44	HAM	Lewis	Hamilton	1985-01-07
1	2	heidfeld	\N	HEI	Nick	Heidfeld	1977-05-10

```
#export drivers csv
status_df.to_sql(name='drivers', con=engine, if_exists='append', index=False)
```

Data Manipulation



3 PgAdmin

Created database and tables in PgAdmin to be used in static and dynamic visualizations

10 tables in total created the relatable database

1. Constructors
2. Races
3. Results
4. Qualifying
5. Circuits
6. Pitstops
7. Laptimes
8. Status
9. Drivers
10. Constructor Standings

Data Manipulation

4 PgAdmin Con't

Used pandas to import scraped csvs, edit the column names for easier SQL use, and export into PgAdmin

```
-- Calculated which team had how many points in each round. This was used with the above table to create the below table
DROP TABLE IF EXISTS team_points_per_round CASCADE;
CREATE TABLE team_points_per_round AS
SELECT c.points, r.year, r.round, con.name FROM constructor_standings AS c
JOIN races AS r on r.race_id=c.race_id
JOIN constructors AS con on con.constructor_id=c.constructor_id;
```

Final Visualization

1 Javascript

Used to populate interactive dashboard including drop down menus, final graphs, and map.

```
let queryAPI = "https://ergast.com/api/f1/"
let queryDriver = "drivers.json?"

let dropdownMenu = d3.select("#selDataset");
let yearMenu = d3.select("#selYear");

// The earliest year in the dataset, based on researching the API
let firstDataYear = Number(1950);


// FOR REFERENCE:
// year = document.getElementById('selYear').value;
// driver = document.getElementById('selDataset').value;

//This function runs when the page is loaded, fills out the year drop down, and makes function calls
function init() {

  // Set up the year selection
  let yearList = [];
  let currentYear = new Date().getFullYear();
  currentYear = Number(currentYear);  // Convert to number just to be safe

  // Add all years to a list
  for (let y = currentYear; y >= firstDataYear; y--) {
    yearList.push(y);
  };

  // Turn list items into drop down options
  yearList.forEach((y)=>{yearMenu.append("option").text(y).property("value").code;
  });

  document.getElementById('season-info').innerHTML = `${yearList[0]} Season Information`;

  // Call on the following functions to fill out visuals for index.html
  driverList(yearList[0]);
  driverStandings(yearList[0]);
  createMarkers(yearList[0]);

};
```

```
// This function finds the driver list for a single given year
function driverList(year){
  console.log("driverList");

  // Use the year to create the query
  queryUrl = queryAPI + year + "/" + queryDriver;

  // Create Driver list
  d3.json(queryUrl).then(function (data) {

    // Navigate to the section of the JSON that has the driver information
    driversYear = data.MRData.DriverTable.Drivers;

    // Create the driver drop down menu
    let driversYear_list = [];

    for (let i in driversYear) {
      driversYear_list.push(driversYear[i].givenName + " " + driversYear[i].familyName);
    };

    driversYear_list.forEach((driver)=>{dropdownMenu.append("option").text(driver).property("value").code;
    });

    // Run the demographics function with the first driver in the list as default
    demographics(driversYear_list[0]);
  })
```

Final Visualization

```
<div id = "Visualizations">
  <div class="container">
      <select id="selYear" onchange="getYear(this.value)"></select><h3 class = "text-center" id = "season-info">Season Information
      <hr>
      <div class="container">
          <div class="row">

              <!-- Standings -->
              <div class="col-sm-12 col-md-5 graph bg-light">
                  <div id="bar"></div>
              </div>

              <div class="col-sm-12 col-md-5 graph bg-light">

              </div>

              <!-- Demographics DropDown -->
              <div class="col-sm-12 col-md-5 graph bg-light">
                  <h4>Driver Demographics</h4>
                  <div class="well" style="min-width: 225px;">
                      <h5>Driver Name:</h5>
                      <select id="selDataset" onchange="optionChanged(this.value)"></select>
                  </div>
                    <div class="panel panel-primary" style="min-width: 225px;">
                      <div class="panel-heading">
                      </div>
                      <div id="driver-metadata" class="panel-body">
            <!-- metadata here -->
                      </div>
                    </div>
              </div>
```

2 HTML

Used to position scraped articles, dashboard graphs, leaflet map, and link to a second page.

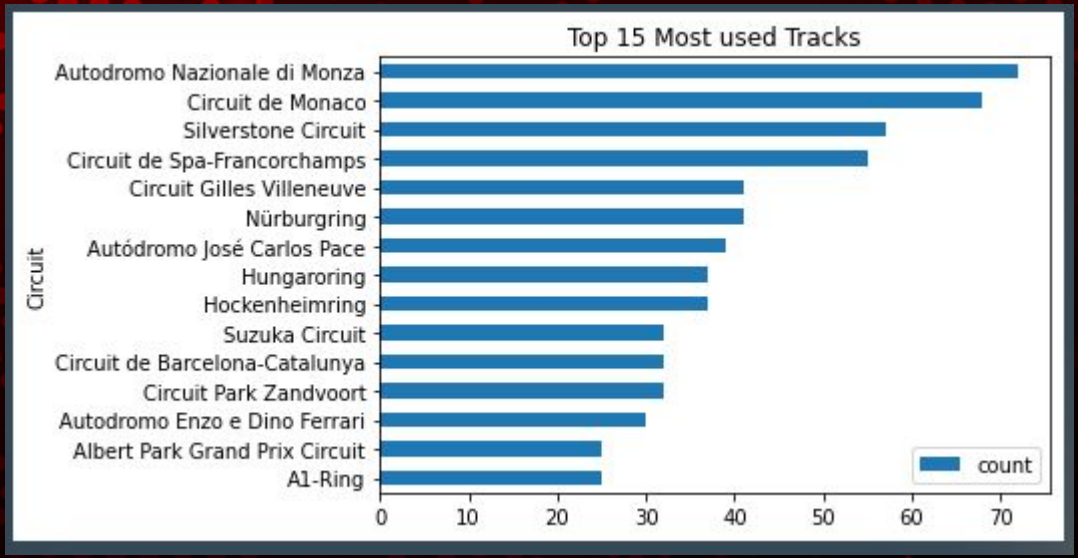
Final Visualization

```css
/* Remove the default margin and padding from the body. */
body {
  padding: 0;
  margin: 0;
}

/* Set map, body, and html to 100% of the screen size. */
#map,
body,
html {
  height: 100%;
}

.navbar-brand{
  font-family:Georgia, 'Times New Roman', Times, serif;
  padding: 1.5rem;
  background-image: linear-gradient(to bottom right, rgb(255, 24, 1), rgb(143, 13, 1));

}

#header {
  border-bottom: 3px solid black;
}

.box {
  min-height: 50px;
  padding: 5px;
  margin: 2%;
  font-weight: 800;
  color: black;
}

#Visualizations{
  margin: 2%;
  padding-top: 1.5rem;
  padding-bottom: 1.5rem;
```

3 CSS

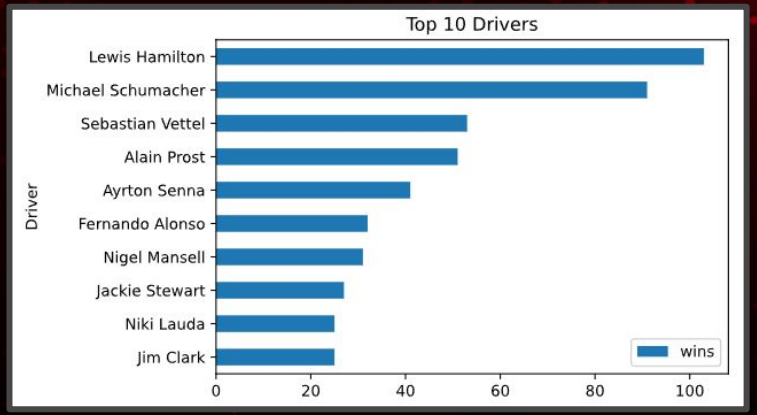
Used to format dashboard graphs

Final Visualization



4 Pandas

Created static graphs for basic analytics



Thanks

http://127.0.0.1:5000/

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik