

# GoSports Release 2 Individual Report - Sam DeFrancisco (sjdefran)

Python 3, Django, sqLite3, HTML

4/17/2022

SE/COMS 319

## Successful Implemented Story Cards

### 1. Integrating nba database into Django's settings.py

**Solution:** Connecting the database to.djangos settings wasn't super hard. The only issue came with getting Django to recognize the path, using an r string which represents a windows path in python did the trick.

**Code:**

```
BASE_DIR = Path(__file__).resolve().parent.parent
DB_DIR = BASE_DIR.parent

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': DB_DIR / r'backend//db_scripts//gosports_nba_api.db',
    }
}
```

### 2. First round-trip to display backend data in Django Html template

**Solution:** The full roundtrip consists of multiple different files working together. Within views.py defining a 'Request' function that access's the django models needed for the template (page) that you are working on and returns a rendered html doc. Next within urls.py you have to set it up so that this function is called when that url is requested. Finally within the actualy html template you have to format the tables using Djangos Template scripting language

- Included example of views.py function, urls.py, and standings.html a Django Template

**Code:**

```
# The Standings Page
def standings_tables(request):
    # Standings returns the whole Standings Table
    # each row is an object .filter() allows us to search for rows by their
    → value in the column given
```

```

# in this case using conference
# all fields available can be found in GoSports.models
east = Standings.objects.filter(conference='East')
west = Standings.objects.filter(conference='West')
# returning a render (request given, template you want to use,
# locals just refers to the local variables defined within this
→ function
# this makes it so you can access east and west within the html
→ template!
return render(request, 'standings.html', locals())

urlpatterns = [
    path("", HomePageView.as_view(), name="home"),
    path("standings/", standings_tables, name="standings"),

<h1 style="text-align: center"><u>Eastern Standings</u></h1><br>
<table class="table" id="tableHeaders1">
    <tr>
        <th>Rank</th>
        <th>Team Name</th>
        <th>Games</th>
        <th>Wins</th>
        <th>Losses</th>
        <th>Win Pct</th>
        <th>Home Record</th>
        <th>Away Record</th>
    </tr>
    <!-- jinja2 Technique -->
    <!-- So we can print only east first, east coming from locals() in
→ views.py -->
    <!-- Can now loop through each row and access fields given by the
→ Standings model -->
    {% if east %}
    {% for team in east %}
    <tr>
        <td>{{team.rank}}</td>
        <td>{{team.team}}</td>

```

```

        <td>{{team.g}}</td>
        <td>{{team.w}}</td>
        <td>{{team.l}}</td>
        <td>{{team.w_pct}}</td>
        <td>{{team.home_record}}</td>
        <td>{{team.road_record}}</td>
    </tr>
{% endfor %}
{% endif %}
</table>

```

### 3. Refomattting DB to use relational Models using Django's ORM

**Solution:** Teams originally each had there own DB tables (one for the celtics, one for the cavs etc...) But that was very redudant. I decided to reformat how I was storing data within the database to use far less tables, now only has (Teams, Seasons, Players, GamesThisWeek, GamesToday, Standings) Using each teams 'id' as their primary key I was able to connect the other models to their given team using a foreign key

- Included example of models.py including what the Teams model looks like, and how I connect each player to their specific team

**Code:**

```

class Teams(models.Model):
    index = models.IntegerField(blank=True, null=True)
    name = models.TextField(db_column='TEAM_NAME', blank=True, null=True) #
    ↪ 'Boston Celtics'
    team_id = models.IntegerField(primary_key=True, db_column='TEAM_ID',
    ↪ blank=True, null=False) # '1610612737'

    class Meta:
        managed = True
        db_table = 'Teams'

class Players(models.Model):
    name = models.TextField(blank=True, null=True) # 'Jayson Tatum'

```

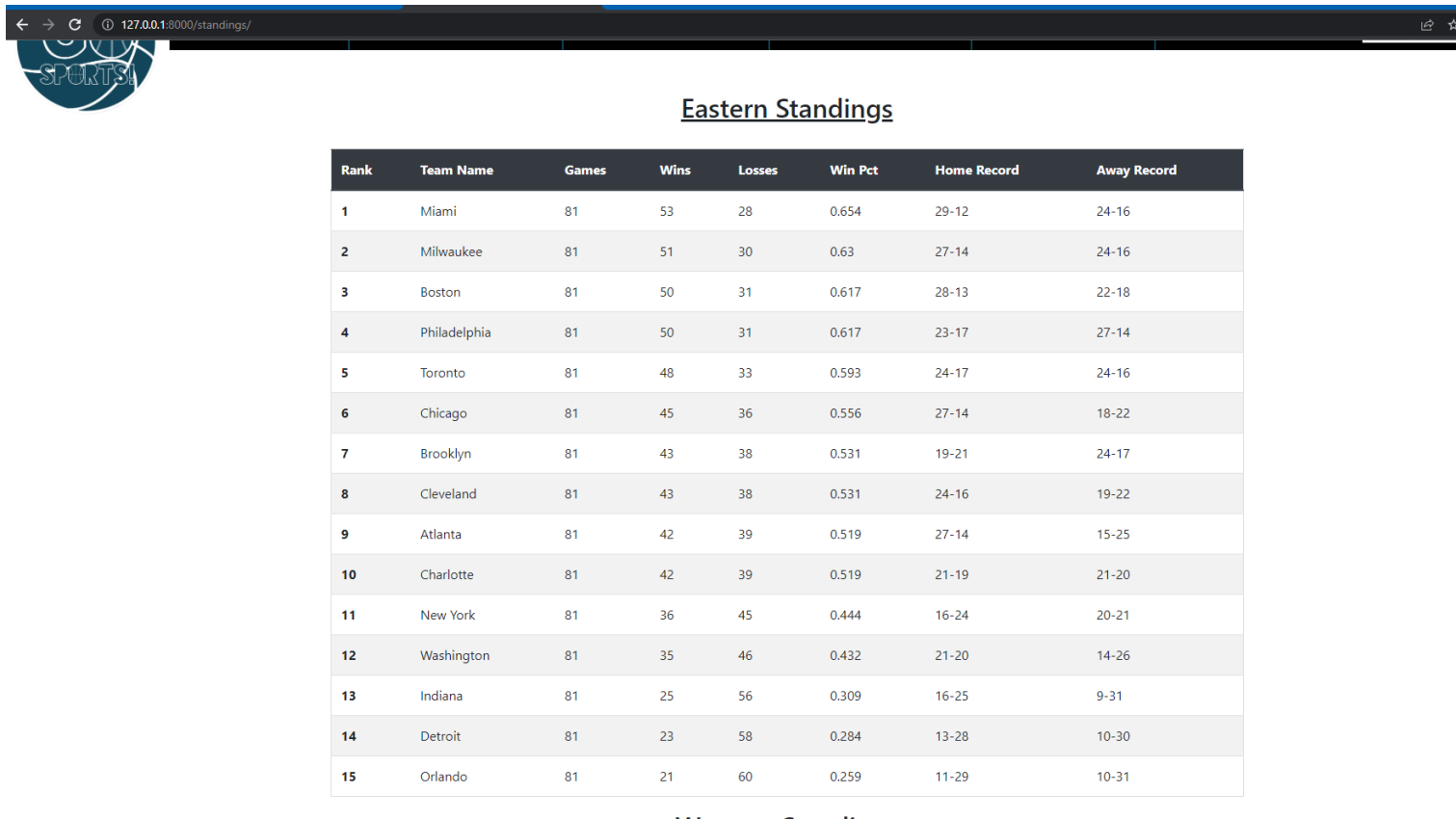
```

player_id = models.TextField(primary_key = True, db_column='PLAYER_ID',
    ↳ blank=True, null=False) # '1626153'
season_id = models.TextField(db_column='SEASON_ID', blank=True,
    ↳ null=True) # '2021-2022'
team = models.ForeignKey(Teams, on_delete=models.CASCADE) # uses teamID
    ↳ to connect to Teams

```

## Screen Shots

Figure 1: What the first round trip produced after implantation's above



Rank	Team Name	Games	Wins	Losses	Win Pct	Home Record	Away Record
1	Miami	81	53	28	0.654	29-12	24-16
2	Milwaukee	81	51	30	0.63	27-14	24-16
3	Boston	81	50	31	0.617	28-13	22-18
4	Philadelphia	81	50	31	0.617	23-17	27-14
5	Toronto	81	48	33	0.593	24-17	24-16
6	Chicago	81	45	36	0.556	27-14	18-22
7	Brooklyn	81	43	38	0.531	19-21	24-17
8	Cleveland	81	43	38	0.531	24-16	19-22
9	Atlanta	81	42	39	0.519	27-14	15-25
10	Charlotte	81	42	39	0.519	21-19	21-20
11	New York	81	36	45	0.444	16-24	20-21
12	Washington	81	35	46	0.432	21-20	14-26
13	Indiana	81	25	56	0.309	16-25	9-31
14	Detroit	81	23	58	0.284	13-28	10-30
15	Orlando	81	21	60	0.259	11-29	10-31

\*\*Level of prof. and contr. on next page\*\*

## Level of Proficiency

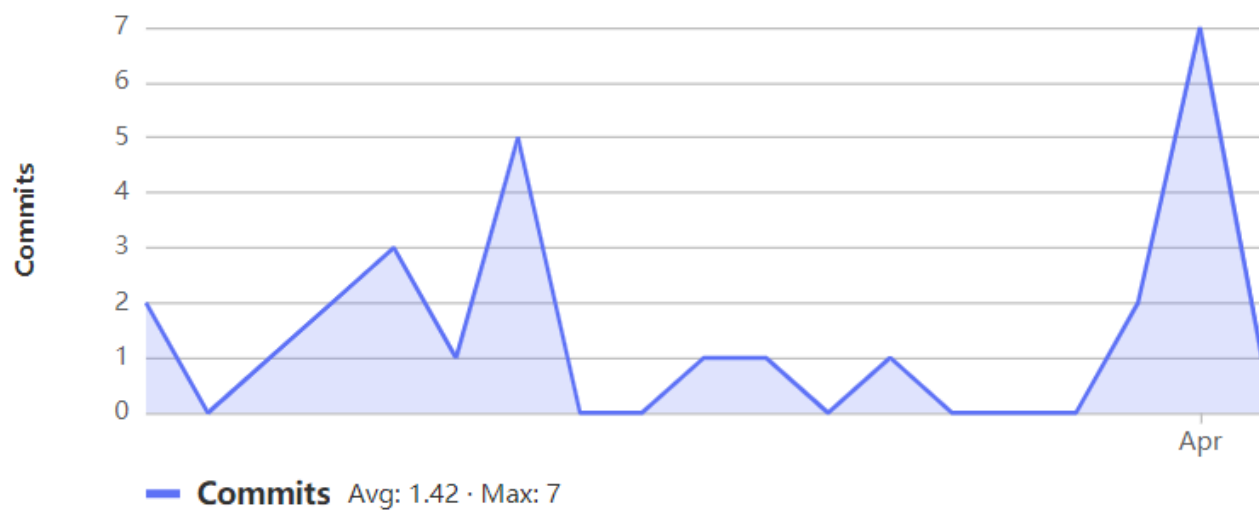
This is not my first project working with python as I have created a few personal project using it. As for sql I had very limited experieence before this working with it but discovered how well pandas works with sql. Over this last sprint I have become much more experienced working with Django as a backend service and feel comfortable.

## Contributions

*Figure 2: Number of commits to gitlab*

### Sam

27 commits (sjdefran@iastate.edu)



Repository Analytics

Programming languages used in this repository

Measured in bytes of code. Excludes generated and vendored code.

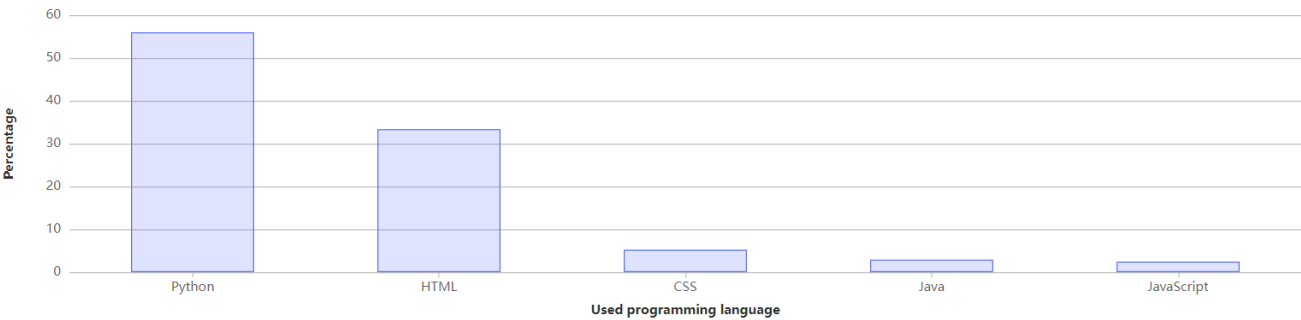


Figure 3: Languages Used

- Python - 56%
- HTML - 34%
- CSS - 5%
- Java - 3%
- JavaScript - 4%