CA270 Project Data Gathering

October 27, 2018

```
In [1]: import requests
        import pandas as pd
        import json

        import warnings
        warnings.filterwarnings('ignore')
```

```
In [2]: def get_req(url):
            resp = requests.get(url)
            return json.loads(resp.text)
```

```
In [3]: def urlify(s):
            return s.replace(" ", "%20")
```

These small csv's were filled in by hand and exported from my mySQL db.

```
In [4]: df_cs = pd.read_csv("class_specs.csv")
        df_affixes = pd.read_csv("affixes.csv")
        df_dungeons = pd.read_csv("dungeons.csv")
```

```
In [5]: df_affixes.head()
```

Out[5]:

	id	level_2	level_4	level_7	level_10
0	FO_BO_SK_IN	Fortified	Bolstering	Skittish	Infested
1	FO_BU_QU_IN	Fortified	Bursting	Quaking	Infested
2	FO_SA_GR_IN	Fortified	Sanguine	Grievous	Infested
3	FO_TE_EX_IN	Fortified	Teeming	Explosive	Infested
4	FO_TE_QU_IN	Fortified	Teeming	Quaking	Infested

```
In [6]: df_cs.head()
```

Out[6]:

	id	base_class	specialisation	category
0	DH_HA	Demon Hunter	Havoc	Melee Damage
1	DH_VE	Demon Hunter	Vengeance	Tank
2	DK_BL	Death Knight	Blood	Tank
3	DK_FR	Death Knight	Frost	Melee Damage
4	DK_UN	Death Knight	Unholy	Melee Damage

```
In [7]: df_dungeons.head()
```

```
Out[7]:    id              name  timer
        0  AD        Atal'dazar   1800
        1  FH          Freehold   2160
        2  KR       Kings' Rest   2340
        3  ML  The MOTHERLODE!!   2340
        4  SB  Siege of Boralus   2160
```

Here is an example of the data I received from an API call. It holds data about: * The dungeon (name, difficulty, etc) * The players (names, classes, specs, etc.) * The run (Time completed, affixes, deaths, etc.)

```
In [8]: tmp_url = ("https://raider.io/api/v1/mythic-plus/runs"
                   "?season=season-bfa-1&region=world&dungeon=all&affixes=fortified")
        tmp_rsp = get_req(tmp_url)
        tmp_rsp["rankings"][0]["run"].keys()
```

```
Out[8]: dict_keys(['season', 'dungeon', 'keystone_run_id', 'keystone_team_id', 'keystone_plato
```

Here I converted each result from the API call into 5 separate entries. * Extract relevant data from the json * Convert dungeon names into their IDs * Convert affix combinations into their IDs * Convert class & spec information into IDs * Create 5 separate entries with the above information in each

```
In [9]: def json_to_entries(js):
            entries = []
            for run in js["rankings"]:
                run = run["run"]

                # Extract data from json
                du_name = run["dungeon"]["name"]
                di_level = run["mythic_level"]
                affixes_json = run["weekly_modifiers"]
                affixes = [j["name"] for j in affixes_json]
                class_specs = [(c["character"]["class"]["name"],
                                c["character"]["spec"]["name"]) for c in run["roster"]]


                # "Atal'dazar" -> "AD"
                du_id = df_dungeons[df_dungeons.name==du_name]["id"].values[0]

                # ['Tyrannical', 'Teeming', 'Volcanic', 'Infested'] -> "TY_TE_VO_IN"
                affix_id = df_affixes[df_affixes.level_2==affixes[0]
                                      ][df_affixes.level_4==affixes[1]
                                       ][df_affixes.level_7==affixes[2]
                                        ][df_affixes.level_10==affixes[3]]["id"].values[0]

                # ('Rogue', 'Subtlety') -> "RO_SU"
```

```
            cs_ids = []
            for cls, spec in class_specs:
                cs_ids.append(df_cs[df_cs.base_class==cls
                                    ][df_cs.specialisation==spec]["id"].values[0])


            # Convert to single entries
            for cs_id in cs_ids:
                entries.append((cs_id, du_id, di_level, affix_id))

        return entries
```

Here I iterate over all regions, dungeons and affixes.
First the API call urls are created, then the data is retrieved and converted into entries.
(this part of the API is outdated, so I am only able to filter for 2 affixes)

```
In [10]: regions = ["eu", "us", "tw", "kr"]
         dungeons = df_dungeons.name.values
         affixes_2 = ["Tyrannical", "Fortified"]
         url_format = ("https://raider.io/api/v1/mythic-plus/runs"
                       "?season=season-bfa-1&region={}&dungeon={}&affixes={}")
         urls = [url_format.format(r, urlify(d), a
                                   ) for r in regions for d in dungeons for a in affixes_2]
```

```
In [11]: entries = []
         for url in urls:
             resp_json = get_req(url)
             entries += json_to_entries(resp_json)
```

Finally, I convert the list of entries into a pandas DataFrame.
This allows me to aggregate duplicate values into a count.

```
In [12]: df_entries = pd.DataFrame.from_records(entries,
                                 columns=["class_id", "dungeon_id", "difficulty_id", "affix_id"])
         df_entries.head()
```

Out[12]:

	class_id	dungeon_id	difficulty_id	affix_id
0	RO_SU	AD	19	TY_TE_VO_IN
1	HU_BM	AD	19	TY_TE_VO_IN
2	DK_BL	AD	19	TY_TE_VO_IN
3	MO_MW	AD	19	TY_TE_VO_IN
4	DH_HA	AD	19	TY_TE_VO_IN

```
In [13]: cols = df_entries.columns.tolist()
         agg_entries = df_entries.groupby(cols,as_index=False).size()
         agg_entries = agg_entries.reset_index().rename(columns={0:'num_completed'})
         agg_entries.head()
```

Out[13]:

	class_id	dungeon_id	difficulty_id	affix_id	num_completed
0	DH_HA	AD	15	FO_BO_SK_IN	3
1	DH_HA	AD	15	FO_SA_GR_IN	3
2	DH_HA	AD	15	TY_RA_NE_IN	1
3	DH_HA	AD	16	FO_BO_SK_IN	2
4	DH_HA	AD	16	FO_SA_GR_IN	4

After aggregation the data is in the correct format to be import into the database. I saved it as a csv file and imported it manually through mySQL workbench.

```
In [14]: agg_entries.to_csv("ca270_entries.csv", header=True, index=False)
```