## 1. Welcome!

.

**The Office!** What started as a British mockumentary series about office culture in 2001 has since spawned ten other variants across the world, including an Israeli version (2010-13), a Hindi version (2019-), and even a French Canadian variant (2006-2007). Of all these iterations (including the original), the American series has been the longest-running, spanning 201 episodes over nine seasons.

In this notebook, we will take a look at a dataset of The Office episodes, and try to understand how the popularity and quality of the series varied over time. To do so, we will use the following dataset: `datasets/office_episodes.csv` , which was downloaded from Kaggle here (https://www.kaggle.com/nehaprabhavalkar/the-office-dataset).

This dataset contains information on a variety of characteristics of each episode. In detail, these are:

### datasets/office_episodes.csv

- **episode_number:** Canonical episode number.
- **season:** Season in which the episode appeared.
- **episode_title:** Title of the episode.
- **description:** Description of the episode.
- **ratings:** Average IMDB rating.
- **votes:** Number of votes.
- **viewership_mil:** Number of US viewers in millions.
- **duration:** Duration in number of minutes.
- **release_date:** Airdate.
- **guest_stars:** Guest stars in the episode (if any).
- **director:** Director of the episode.
- **writers:** Writers of the episode.
- **has_guests:** True/False column for whether the episode contained guest stars.
- **scaled_ratings:** The ratings scaled from 0 (worst-reviewed) to 1 (best-reviewed).

In [35]:

```
# Use this cell to begin your analysis, and add as many as you would like!
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [36]:

```
office = pd.read_csv('datasets/office_episodes.csv')
office.head()
```

Out[36]:

| | episode_number | season | episode_title | description | ratings | votes | viewership_mil | duration |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | Pilot | The premiere episode introduces the boss and s... | 7.5 | 4936 | 11.2 | 23 |
| **1** | 1 | 1 | Diversity Day | Michael's off color remark puts a sensitivity ... | 8.3 | 4801 | 6.0 | 23 |
| **2** | 2 | 1 | Health Care | Michael leaves Dwight in charge of picking the... | 7.8 | 4024 | 5.8 | 22 |
| **3** | 3 | 1 | The Alliance | Just for a laugh, Jim agrees to an alliance wi... | 8.1 | 3915 | 5.4 | 23 |
| **4** | 4 | 1 | Basketball | Michael and his staff challenge the warehouse ... | 8.4 | 4294 | 5.0 | 23 |

In [37]:

```
def colors(x):
    if x<0.25:
        return 'red'
    elif x>=0.25 and x<0.50:
        return 'orange'
    elif x>=0.50 and x<0.75:
        return 'lightgreen'
    elif x>=0.75:
        return 'darkgreen'
```

In [38]:

```
def sizes(x):
    if x==True:
        return 250
    else:
        return 25
```

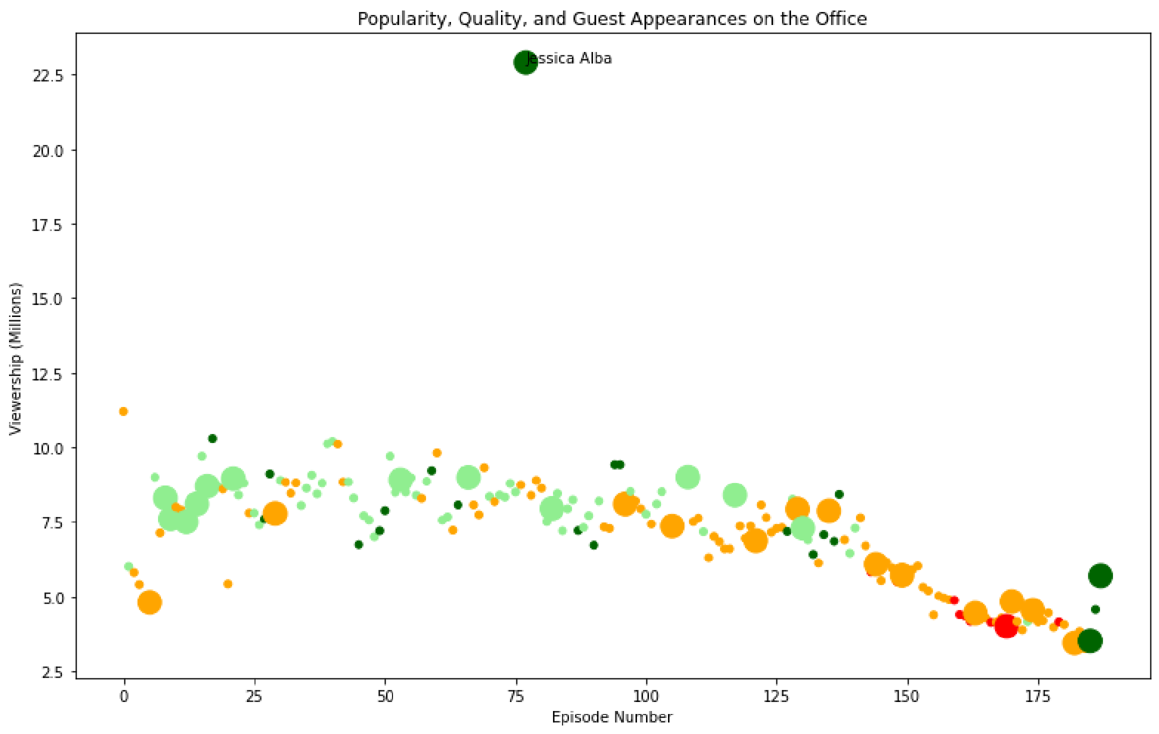
In [39]:

```
ep_no = office['episode_number']
viewership = office['viewership_mil']
colours = office['scaled_ratings'].apply(colors)
size = office['has_guests'].apply(sizes)
temp = office[office['viewership_mil']==max(office['viewership_mil'])]
print(temp)
top_star = 'Jessica Alba'
```

```
    episode_number  season  episode_title  \
77              77       5  Stress Relief

                                         description  ratings  vote
s  \
77  Dwight's too-realistic fire alarm gives Stanle...      9.7   817
0

    viewership_mil  duration release_date  \
77           22.91        60   2009-02-01

                                     guest_stars       director  \
77  Cloris Leachman, Jack Black, Jessica Alba  Jeffrey Blitz

             writers  has_guests  scaled_ratings
77  Paul Lieberstein        True         0.96875
```

In [40]:

```
fig=plt.figure(figsize=(11,7))
plt.scatter(ep_no,viewership,c=colours,marker='o',s=size)
plt.title("Popularity, Quality, and Guest Appearances on the Office")
plt.xlabel("Episode Number")
plt.ylabel("Viewership (Millions)")
plt.text(temp['episode_number'],temp['viewership_mil'],top_star)
plt.show()
plt.clf()
```



```
<Figure size 432x288 with 0 Axes>
```

```
In [41]:
```

```
%%nose
import pandas as pd
import numpy as np

color_data = np.genfromtxt('datasets/color_data.csv', delimiter=',')
bonus_color_data = np.genfromtxt('datasets/bonus_color_data.csv', delimiter=',')
bonus_color_data_2 = np.genfromtxt('datasets/bonus_color_data_2.csv', delimiter
=',')
solution_data = pd.read_csv('datasets/solution_data.csv')

x_axis_data = solution_data['x_axis'].values
y_axis_data = solution_data['y_axis'].values
size_data = solution_data['sizes'].values


# Try to retrieve student plot data, if it's been specified, otherwise set test
values to null
try:
    # Generate x and y axis containers
    stu_yaxis_cont = []
    stu_xaxis_cont = []
    stu_sizes_cont = []
    stu_colors_cont = []

    # Loop through every axis in student's figure and grab x and y data
    for col in fig.gca().collections:
        stu_yaxis_cont.append(col._offsets.data[:,1])
        stu_xaxis_cont.append(col._offsets.data[:,0])
        stu_sizes_cont.append(np.full((1, len(col._offsets.data[:,0])), col._siz
es)[0])
        stu_colors_cont.append(col._facecolors)

    # Get figure labels
    title = fig.gca()._axes.get_title()
    x_label = fig.gca()._axes.get_xlabel()
    y_label = fig.gca()._axes.get_ylabel()

    # Concatenate lists to compare to test plot
    stu_yaxis = np.concatenate(stu_yaxis_cont)
    stu_xaxis = np.concatenate(stu_xaxis_cont)
    stu_sizes = np.concatenate(stu_sizes_cont)
    stu_colors = np.concatenate(stu_colors_cont)
except:
    title = 'null'
    x_label = 'null'
    y_label = 'null'
    stu_yaxis = 'null'
    stu_xaxis = 'null'
    stu_sizes = [0, 1]
    stu_colors = [0, 1]

# Tests
def test_fig_exists():
    import matplotlib
    # Extra function to test for existence of fig to allow custom feedback
    def test_fig():
        try:
            fig
            return True
        except:
            return False
    assert (test_fig() == True), \
    'Did you correctly initalize a `fig` object using `fig = plt.figure()`?'
    assert (type(fig) == matplotlib.figure.Figure), \
    'Did you correctly initalize a `fig` object using `fig = plt.figure()`?'

def test_y_axis():
    assert (sorted(stu_yaxis) == y_axis_data).all(), \
    'Are you correctly plotting viwership in millions on the y axis? Make sure y
ou are calling your plot in the same cell that you initialize `fig`!'

def test_x_axis():
    assert (sorted(stu_xaxis) == x_axis_data).all(), \
    'Are you correctly plotting episode number on the x axis? Make sure you are
calling your plot in the same cell that you initialize `fig`!'

def test_colors():
    assert (len(stu_colors) == len(color_data)), \
    'Are you correctly setting the colors according to the rating scheme provide
d? Make sure you are calling your plot in the same cell that you initialize `fig
`!'
    assert (np.sort(color_data) == np.sort(stu_colors)).all() or \
    (np.sort(bonus_color_data) == np.sort(stu_colors)).all() or \
    (np.sort(bonus_color_data_2) == np.sort(stu_colors)).all(), \
    'Are you correctly setting the colors according to the rating scheme provide
d? Make sure you are calling your plot in the same cell that you initialize `fig
`!'

def test_size():
    assert (len(stu_sizes) == len(size_data)), \
    'Are you correctly plotting all points as size 25, except for guest-star epi
sodes which are sized at 250? Make sure you are calling your plot in the same ce
ll that you initialize `fig`!'
    assert all(size_data == np.sort(stu_sizes)), \
    'Are you correctly plotting all points as size 25, except for guest-star epi
sodes which are sized at 250? Make sure you are calling your plot in the same ce
ll that you initialize `fig`!'

def test_labels():
    assert (title.lower() == ('Popularity, Quality, and Guest Appearances on the
Office').lower()), \
    'Did you set the correct title? Make sure you are specifying your plot in th
e same cell that you initialize `fig`!'
    assert (x_label.lower() == ('Episode Number').lower()), \
    'Did you set the correct x label? Make sure you are specifying your plot in
the same cell that you initialize `fig`!'
    assert (y_label.lower() == ('Viewership (Millions)').lower()), \
    'Did you set the correct x label? Make sure you are specifying your plot in
the same cell that you initialize `fig`!'

def test_stars():
    assert (top_star == 'Cloris Leachman' or top_star == 'Jack Black' or top_sta
r == 'Jessica Alba'), \
    "Are you correctly assigning one of the guest stars of the most popular epis
ode to `top_star`?"
```

Out[41]:

```
7/7 tests passed
```