In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
%matplotlib inline
```

In [2]:

```
df_raw = pd.read_csv('pokemon.csv')
```

In [3]:

```
df_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Data columns (total 41 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   abilities          801 non-null    object 
 1   against_bug        801 non-null    float64
 2   against_dark       801 non-null    float64
 3   against_dragon     801 non-null    float64
 4   against_electric   801 non-null    float64
 5   against_fairy      801 non-null    float64
 6   against_fight      801 non-null    float64
 7   against_fire       801 non-null    float64
 8   against_flying     801 non-null    float64
 9   against_ghost      801 non-null    float64
 10  against_grass      801 non-null    float64
 11  against_ground     801 non-null    float64
 12  against_ice        801 non-null    float64
 13  against_normal     801 non-null    float64
 14  against_poison     801 non-null    float64
 15  against_psychic    801 non-null    float64
 16  against_rock       801 non-null    float64
 17  against_steel      801 non-null    float64
 18  against_water      801 non-null    float64
 19  attack             801 non-null    int64  
 20  base_egg_steps     801 non-null    int64  
 21  base_happiness     801 non-null    int64  
 22  base_total         801 non-null    int64  
 23  capture_rate       801 non-null    object 
 24  classfication      801 non-null    object 
 25  defense            801 non-null    int64  
 26  experience_growth  801 non-null    int64  
 27  height_m           781 non-null    float64
 28  hp                 801 non-null    int64  
 29  japanese_name      801 non-null    object 
 30  name               801 non-null    object 
 31  percentage_male    703 non-null    float64
 32  pokedex_number     801 non-null    int64  
 33  sp_attack          801 non-null    int64  
 34  sp_defense         801 non-null    int64  
 35  speed              801 non-null    int64  
 36  type1              801 non-null    object 
 37  type2              417 non-null    object 
 38  weight_kg          781 non-null    float64
 39  generation         801 non-null    int64  
 40  is_legendary       801 non-null    int64  
dtypes: float64(21), int64(13), object(7)
memory usage: 256.7+ KB
```

In [4]:

```
df_raw.describe()
```

Out[4]:

	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_figh
count	801.000000	801.000000	801.000000	801.000000	801.000000	801.00000
mean	0.996255	1.057116	0.968789	1.073970	1.068976	1.06554
std	0.597248	0.438142	0.353058	0.654962	0.522167	0.71725
min	0.250000	0.250000	0.000000	0.000000	0.250000	0.00000
25%	0.500000	1.000000	1.000000	0.500000	1.000000	0.50000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
max	4.000000	4.000000	2.000000	4.000000	4.000000	4.00000

8 rows × 34 columns

In [5]:

```
df_raw.head()
```

Out[5]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	ag
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	

5 rows × 41 columns

In [6]:

```
df_raw.isnull().sum()
```

Out[6]:

```
abilities              0
against_bug            0
against_dark           0
against_dragon         0
against_electric       0
against_fairy          0
against_fight          0
against_fire           0
against_flying         0
against_ghost          0
against_grass          0
against_ground         0
against_ice            0
against_normal         0
against_poison         0
against_psychic        0
against_rock           0
against_steel          0
against_water          0
attack                 0
base_egg_steps         0
base_happiness         0
base_total             0
capture_rate           0
classfication          0
defense                0
experience_growth      0
height_m              20
hp                     0
japanese_name          0
name                   0
percentage_male       98
pokedex_number         0
sp_attack              0
sp_defense             0
speed                  0
type1                  0
type2                384
weight_kg             20
generation             0
is_legendary           0
dtype: int64
```

In [83]:

```
df = df_raw[['pokedex_number','name','percentage_male','type1','type2','height_m','weight_kg','capture_rate','abilities','base_happiness','hp','attack','defense','sp_attack','sp_defense','speed','generation','base_egg_steps','experience_growth','is_legendary']]
df.head()
```

Out[83]:

	pokedex_number	name	percentage_male	type1	type2	height_m	weight_kg	capt
0	1	Bulbasaur	88.1	grass	poison	0.7	6.9	
1	2	Ivysaur	88.1	grass	poison	1.0	13.0	
2	3	Venusaur	88.1	grass	poison	2.0	100.0	
3	4	Charmander	88.1	fire	NaN	0.6	8.5	
4	5	Charmeleon	88.1	fire	NaN	1.1	19.0	

In [72]:

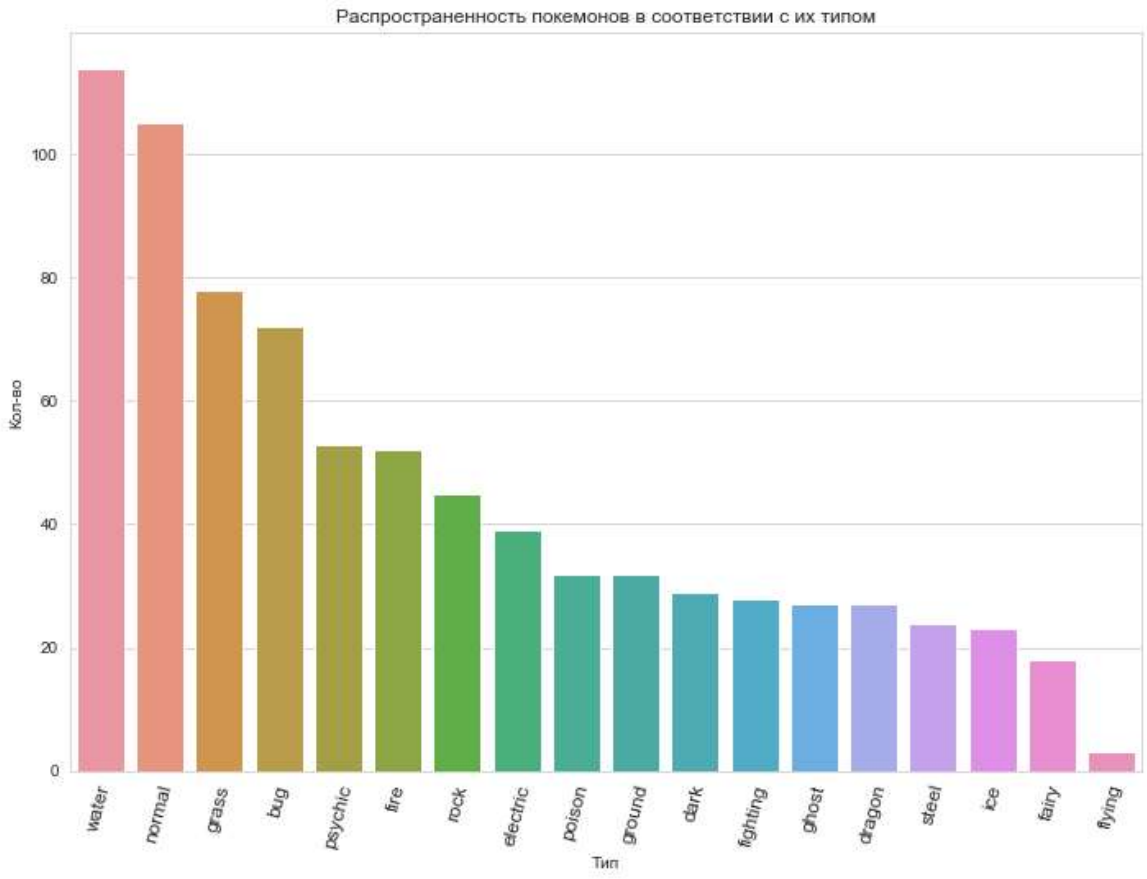
```
yy = pd.value_counts(df['type1'])

fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.set_style("whitegrid")

ax = sns.barplot(x=yy.index, y=yy, data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation = 75, fontsize = 12)
ax.set(xlabel='Тип', ylabel='Кол-во')
ax.set_title('Распространенность покемонов в соответствии с их основным типом')
```

Out[72]:

```
Text(0.5, 1.0, 'Распространенность покемонов в соответствии с их типом')
```



In [73]:

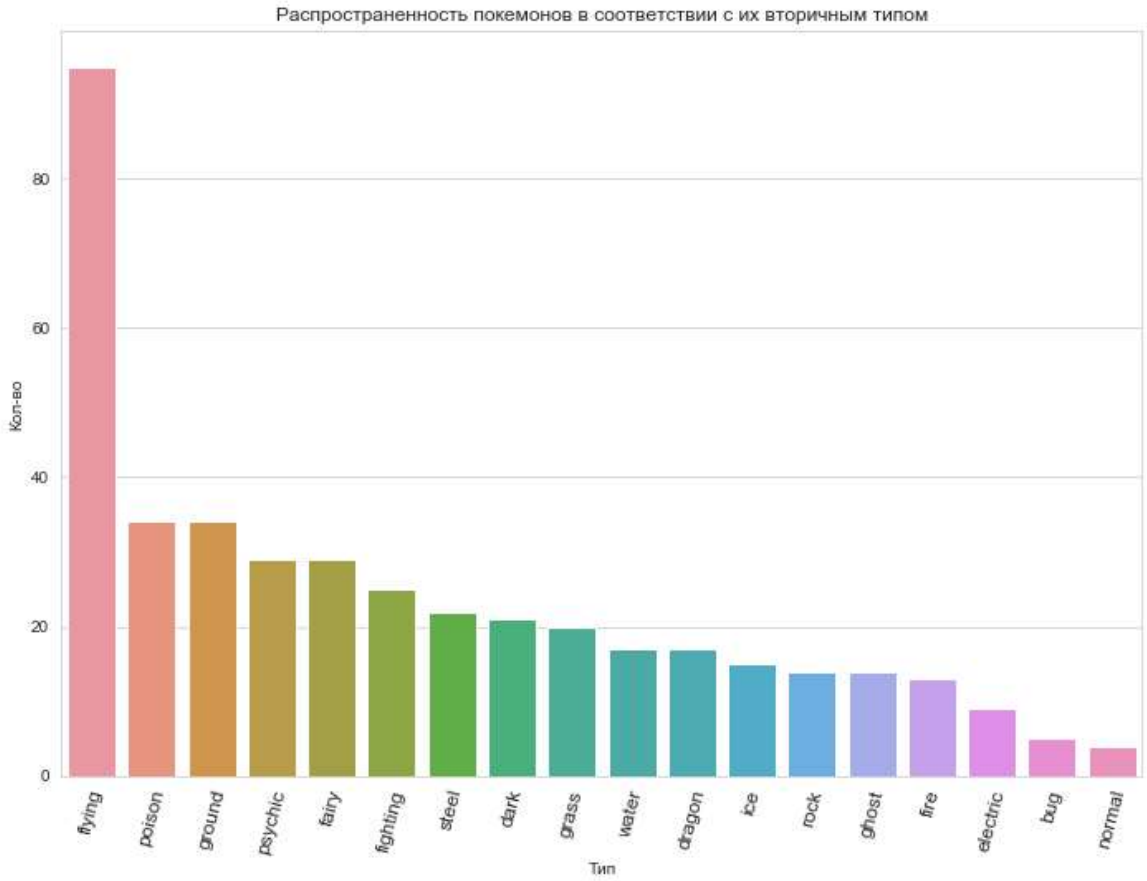
```
yy = pd.value_counts(df['type2'])

fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.set_style("whitegrid")

ax = sns.barplot(x=yy.index, y=yy, data=df)
ax.set_xticklabels(ax.get_xticklabels(), rotation = 75, fontsize = 12)
ax.set(xlabel='Тип', ylabel='Кол-во')
ax.set_title('Распространенность покемонов в соответствии с их вторичным типом')
```

Out[73]:

```
Text(0.5, 1.0, 'Распространенность покемонов в соответствии с их вторичным типом')
```



Корелляции

Визуализация всех возможных корелляций

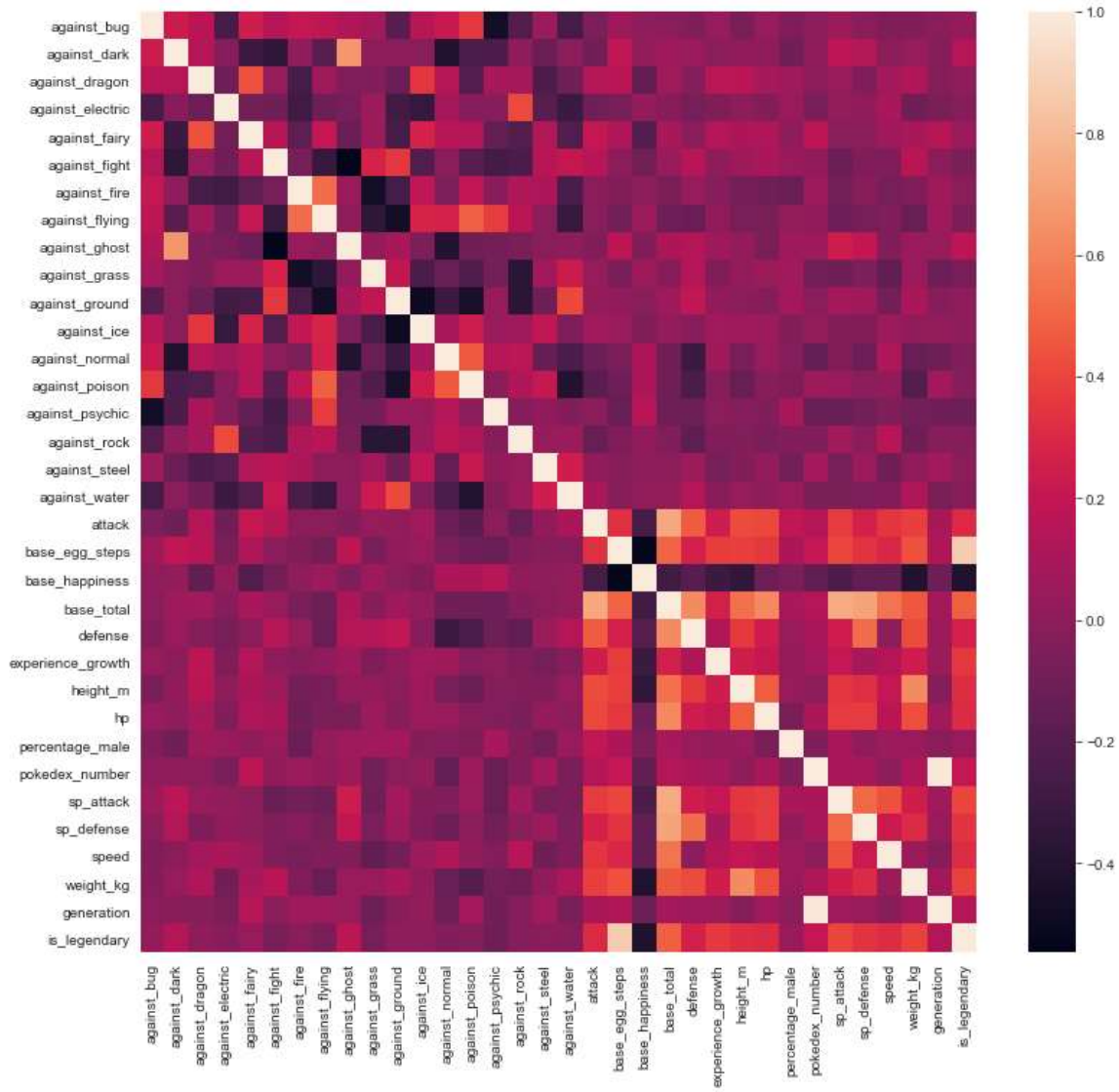
In [77]:

```
numeric_clmns = df_raw.dtypes[df_raw.dtypes != "object"].index

f, ax = plt.subplots(figsize=(13, 12))
corr = df_raw[numeric_clmns].corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Out[77]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1707fd7de20>
```



Определение свойств легендарного покемона

Узнаем, связан ли первичный/вторичный тип покемона с его вероятностью оказаться легендарным

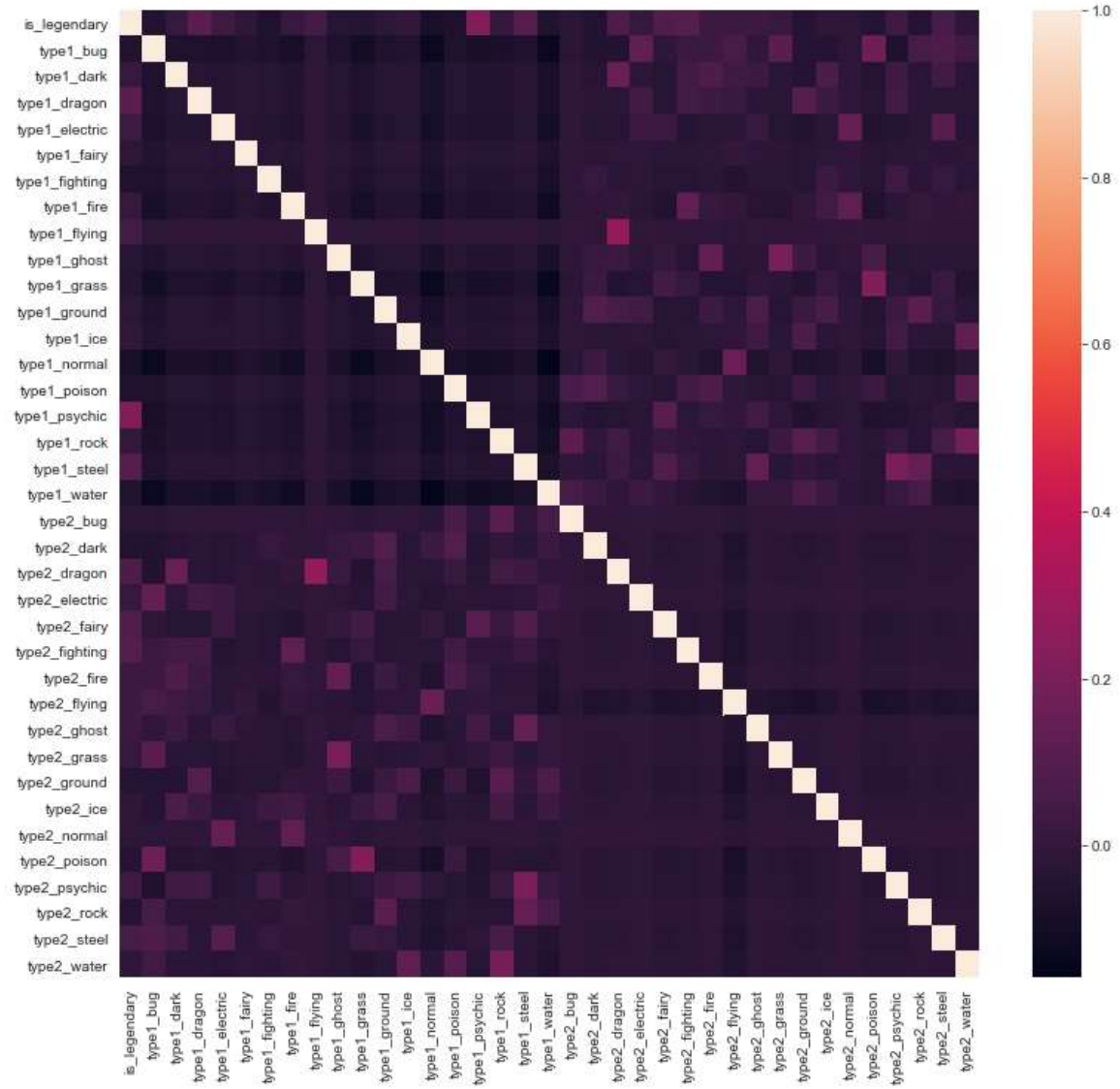
In [76]:

```
sd = df[['type1', 'type2', 'is_legendary']]
md = pd.get_dummies(sd)

corr = md.corr()
f, ax = plt.subplots(figsize=(13, 12))
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Out[76]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1707e6535b0>
```



Как мы можем видеть, тип не влияет на вероятность того, что покемон окажется легендарным

Давайте попробуем поработать с колонкой **capture_rate**

In [79]:

```
df.capture_rate.unique()
```

Out[79]:

```
array(['45', '255', '120', '127', '90', '190', '75', '235', '150', '25',
       '170', '50', '200', '100', '180', '60', '225', '30', '35', '3',
       '65', '70', '125', '205', '155', '145', '130', '140', '15', '220',
       '160', '80', '55', '30 (Meteorite)255 (Core)'], dtype=object)
```

Из выведенных данных видно, что каждый элемент столбца - это переменные типа string, но также есть элемент *'30 (Meteorite)255 (Core)'*. Давайте заменим его на 1000, чтобы в будущем не возникало проблем при обработке данных, а также заменим типы даннх с string на int.

In [81]:

```
df['capture_rate'].replace('30 (Meteorite)255 (Core)', '1000', inplace=True)
pd.to_numeric(df['capture_rate'])
df['capture_rate'] = df['capture_rate'].astype(int)
df.capture_rate.unique()
```

```
c:\users\daniil\appdata\local\programs\python\python38\lib\site-packages\p
andas\core\generic.py:6746: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._update_inplace(new_data)
<ipython-input-81-9f632e1429fa>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['capture_rate'] = df['capture_rate'].astype(int)
```

Out[81]:

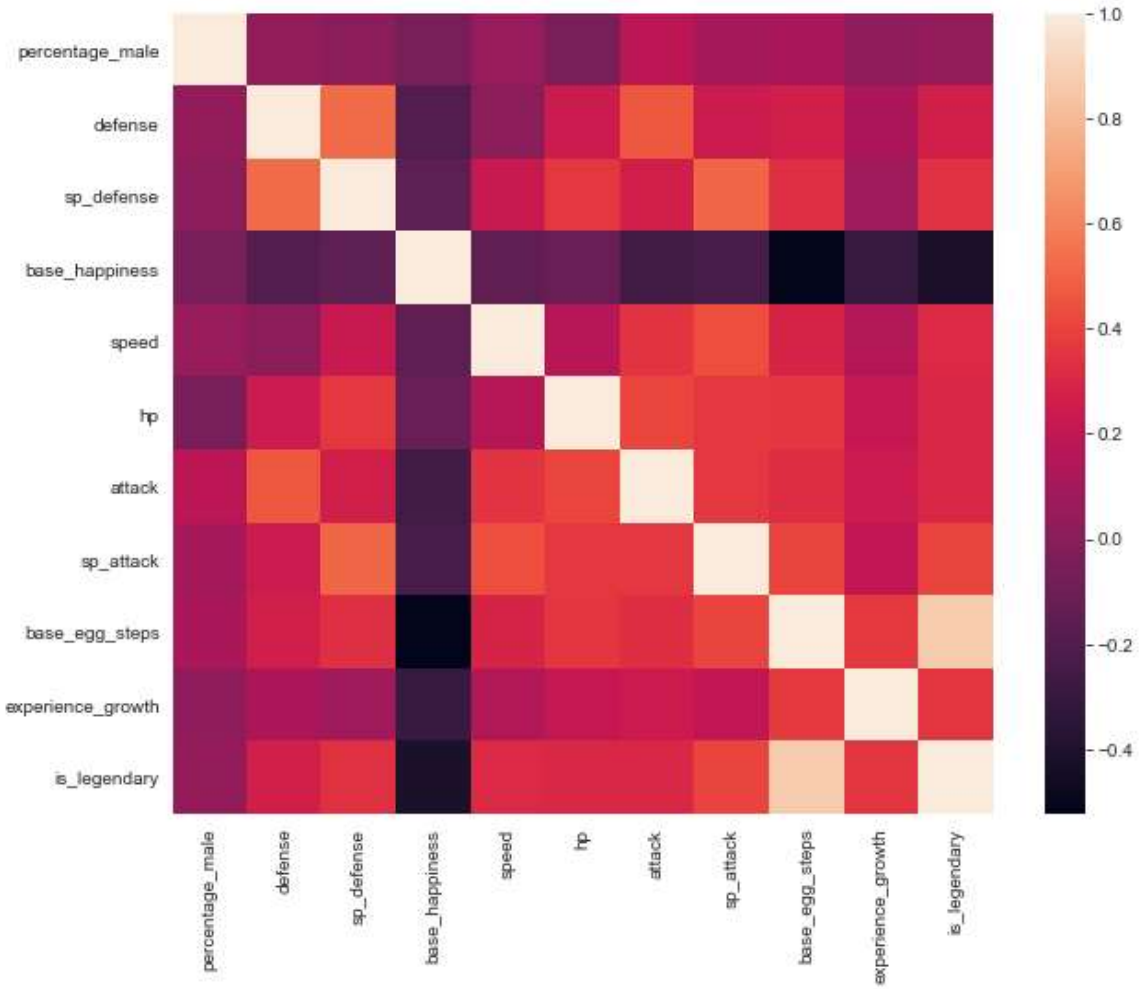
```
array([  45,  255,  120,  127,   90,  190,   75,  235,  150,   25,  170,
         50,  200,  100,  180,   60,  225,   30,   35,    3,   65,   70,
        125,  205,  155,  145,  130,  140,   15,  220,  160,   80,   55,
       1000])
```

In [84]:

```
corr = df[['percentage_male', 'capture_rate','defense','sp_defense','base_happiness',
'speed', 'hp','attack','sp_attack','base_egg_steps','experience_growth','is_legendary'
]].corr()
f, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1707e30bd90>
```



Исходя из полученной матрицы:
base_egg_steps имеет высокую корелляцию с вероятностью легендарности покемона
base_happiness, percentage_male и **capture_rate** имеют негативную корелляцию

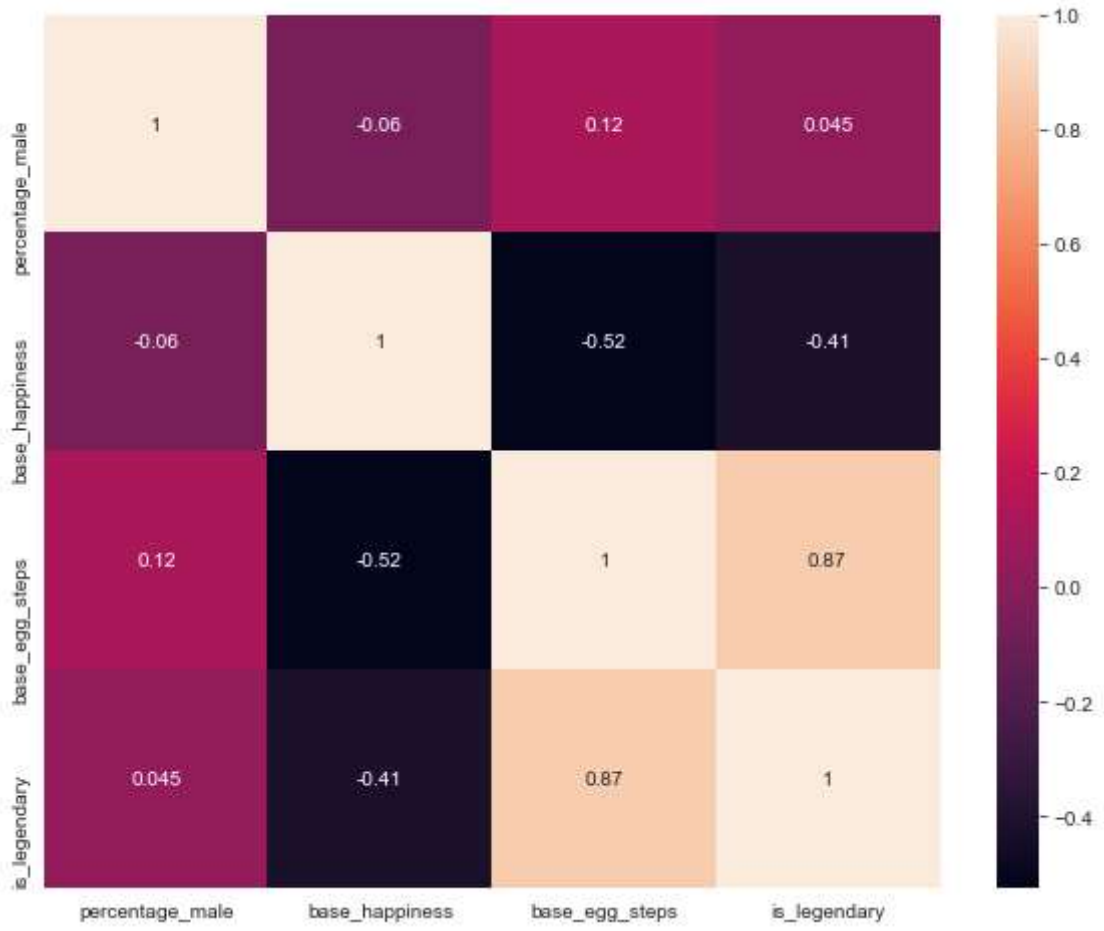
Взглянем на отобранные параметры поближе:

In [88]:

```
corr = df[['percentage_male', 'capture_rate','base_happiness','base_egg_steps','is_lege
ndary']].corr()
f, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values,
           annot=True)
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1707ab41eb0>
```

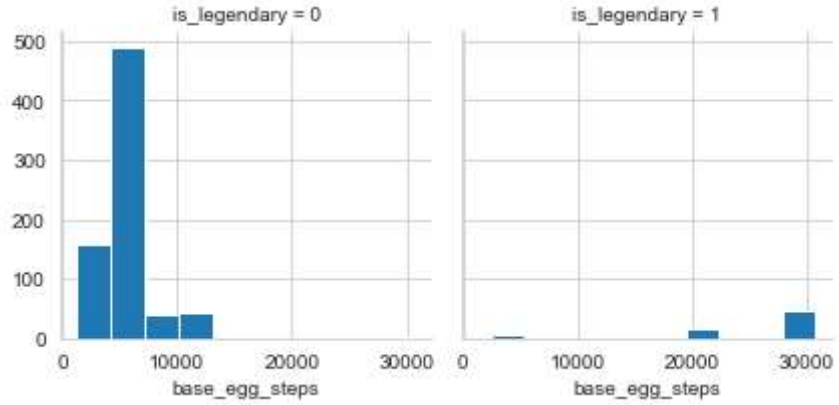


Похоже, что именно **base_egg_steps** является ключевым критерием для определения легендарности покемона.

Посмотрим на базовое количество шагов у всех легендарных покемонов:

In [93]:

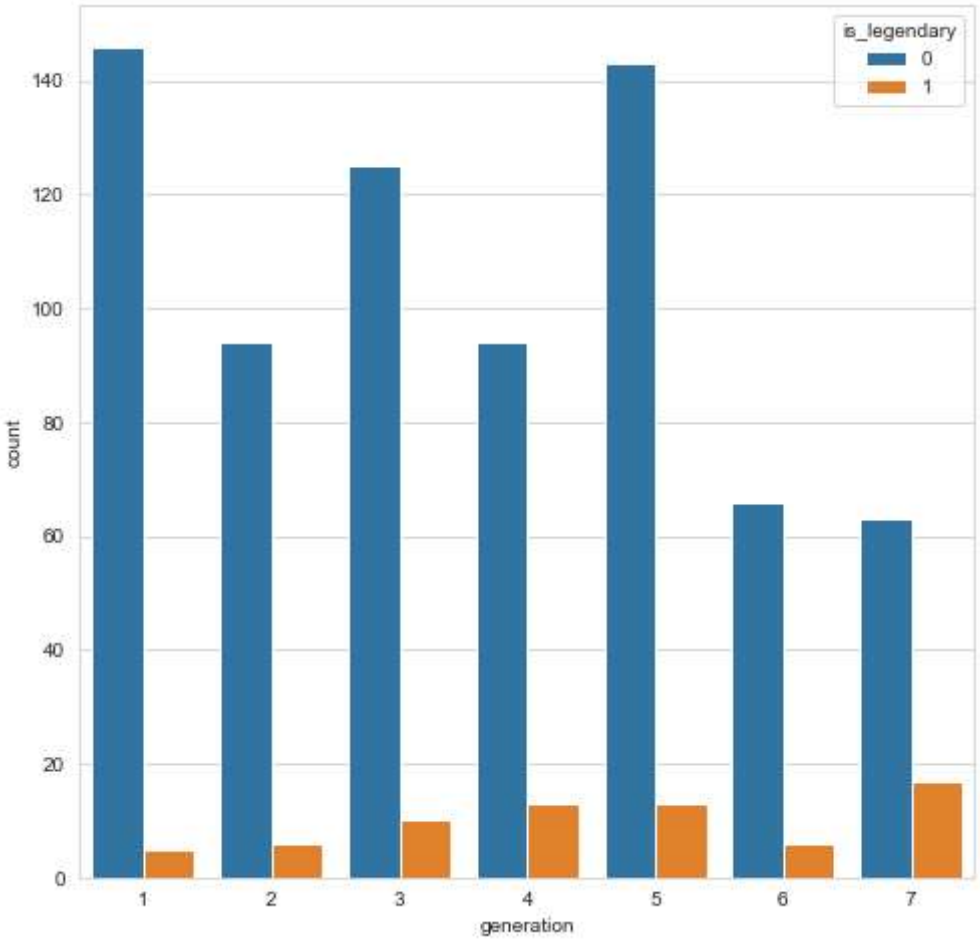
```
ax = sns.FacetGrid(df, col='is_legendary')
ax = ax.map(plt.hist, 'base_egg_steps')
```



Посмотрим на наличие связи между поколением и количеством легендарных покемонов

In [95]:

```
f, ax = plt.subplots(figsize=(8, 8))
ax = sns.countplot(x="generation", hue = 'is_legendary', data=df, )
```



Как видно из графика, количество легендарных покемонов всегда примерно одинаковое, в пределах 20 на поколение.
Однако, что интересно, в 7 поколении почти треть от всех покемонов - легендарные.

In [ ]:

```
```