top_10000-most-popular-movies-from-imdb

Model Developed by
Name: **Umair Ali**
Contact: **+923480233673**
Email: **uape00@gmail.com**
GitHub: **https://github.com/1umairali/models**

Developed a comprehensive data analysis project using a dataset of the top 10,000 most popular movies from IMDb. The project involved data cleaning, exploratory data analysis (EDA), and visualizations using Python libraries such as Pandas, Matplotlib, and Seaborn. Extracted insights into trends in movie ratings, genres, release years, and other key metrics. The analysis helped highlight correlations between popularity, ratings, and other attributes of successful films.

In [1]:
```
import numpy as np  # for numeric calculation
import pandas as pd  # for data analysis and manupulation
import matplotlib.pyplot as plt  # for data visualization
import seaborn as sns  # for data visualization
```

In [2]:
```
url = "https://raw.githubusercontent.com/1umairali/models/main/imdb_10000_popular_movies/top_10000_most_popular_movies_from_im
orignal_df = pd.read_csv(url)
orignal_df
```

Out[2]:

	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget
0	758323	The Pope's Exorcist	2023-04-05	['Horror', 'Mystery', 'Thriller']	English	7.4	619	5089.969	Father Gabriele Amorth, Chief Exorcist of the ...	18000000
1	640146	Ant-Man and the Wasp: Quantumania	2023-02-15	['Action', 'Adventure', 'Science Fiction']	English	6.6	2294	4665.438	Super-Hero partners Scott Lang and Hope van Dy...	200000000
2	502356	The Super Mario Bros. Movie	2023-04-05	['Animation', 'Adventure', 'Family', 'Fantasy'...	English	7.5	1861	3935.550	While working underground to fix a water main,...	100000000
3	868759	Ghosted	2023-04-18	['Action', 'Comedy', 'Romance']	English	7.2	652	2791.532	Salt-of-the-earth Cole falls head over heels f...	0
4	594767	Shazam! Fury of the Gods	2023-03-15	['Action', 'Comedy', 'Fantasy', 'Adventure']	English	6.8	1510	2702.593	Billy Batson and his foster siblings, who tran...	125000000
...	...	...	...	...	...	...	...	...	...	...
9995	374473	I, Daniel Blake	2016-10-21	['Drama']	English	7.7	1220	10.774	A middle aged carpenter, who requires state we...	0

	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget
9996	16774	Hellboy Animated: Sword of Storms	2006-10-28	['TV Movie', 'Fantasy', 'Animation', 'Action',...	English	6.3	99	12.739	A folklore professor becomes unwittingly posse...	0
9997	13564	Return to House on Haunted Hill	2007-10-03	['Horror', 'Thriller']	English	5.6	263	12.769	Eight years have passed since Sara Wolfe and E...	0
9998	482204	My Sister-in-law's Job	2017-08-31	['Drama', 'Romance']	Korean	5.0	5	10.425	An erotic film that depicts the dangerous rela...	0
9999	444539	The Bookshop	2017-11-10	['Drama']	English	6.5	382	12.525	Set in a small English town in 1959, a woman d...	5400000

10000 rows × 14 columns

In [3]:
```
orignal_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    10000 non-null  int64
 1   title                 10000 non-null  object
 2   release_date          9979 non-null   object
 3   genres                10000 non-null  object
 4   original_language     10000 non-null  object
 5   vote_average          10000 non-null  float64
 6   vote_count            10000 non-null  int64
 7   popularity            10000 non-null  float64
 8   overview              9923 non-null   object
 9   budget                10000 non-null  int64
 10  production_companies  10000 non-null  object
 11  revenue               10000 non-null  int64
 12  runtime               10000 non-null  int64
 13  tagline               7241 non-null   object
dtypes: float64(2), int64(5), object(7)
memory usage: 1.1+ MB
```

In [4]:
```
orignal_df.isnull().sum()
```

Out[4]:
```
id                       0
title                    0
release_date            21
genres                   0
original_language        0
vote_average             0
vote_count               0
popularity               0
overview                77
budget                   0
production_companies     0
revenue                  0
runtime                  0
tagline               2759
dtype: int64
```

In [5]:
```
#copy orignal dataframe
df = orignal_df.copy()
```

In [6]:
```
# drop tagling column, contains max NaN values.
df = df.drop(['tagline'], axis=1)

# convert "release_date" data type 'object' to 'datetime'
df['release_date'] = pd.to_datetime(df['release_date'])
df['genres'] = df['genres'].astype('string')

# take median() of 'release_date' column and fill NaN value
df['release_date'].fillna(df['release_date'].median(), inplace=True)

# drop NaN value rows from 'overview' columns
df.dropna(subset=['overview'], inplace=True)
```

May be I didn't fill NaN values well

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9923 entries, 0 to 9999
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    9923 non-null   int64
 1   title                 9923 non-null   object
 2   release_date          9923 non-null   datetime64[ns]
 3   genres                9923 non-null   string
 4   original_language     9923 non-null   object
 5   vote_average          9923 non-null   float64
 6   vote_count            9923 non-null   int64
 7   popularity            9923 non-null   float64
 8   overview              9923 non-null   object
 9   budget                9923 non-null   int64
 10  production_companies  9923 non-null   object
 11  revenue               9923 non-null   int64
 12  runtime               9923 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(5), object(4), string(1)
memory usage: 1.1+ MB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
id                      0
title                   0
release_date            0
genres                  0
original_language       0
vote_average            0
vote_count              0
popularity              0
overview                0
budget                  0
production_companies    0
revenue                 0
runtime                 0
dtype: int64
```

Upcomming Movies

In [9]:
```
upcomming_movies_df = df.copy()

upcomming_movies_df.sort_values(by=['release_date'], inplace=True, ascending=True)
upcomming_movies = upcomming_movies_df.loc[(df['release_date'] >= '2023-09-15')].reset_index()
upcomming_movies = upcomming_movies.drop(['vote_average','vote_count','budget','revenue','runtime'], axis=1)
upcomming_movies.head()
```

Out[9]:

	index	id	title	release_date	genres	original_language	popularity	overview	production_companies
0	2976	299054	The Expendables 4	2023-09-21	['Action', 'Adventure', 'Thriller', 'War']	English	22.788	The Expendables square up against an arms deal...	['Millennium Films', 'Campbell Grobman Films']
1	7558	893723	PAW Patrol: The Mighty Movie	2023-09-28	['Animation', 'Family', 'Comedy']	English	11.993	A magical meteor crash lands in Adventure City...	['Nickelodeon Movies', 'Spin Master', 'Paramou...
2	7467	539972	Kraven the Hunter	2023-10-04	['Action', 'Science Fiction']	English	11.255	Sergei Kravinoff is a big game hunter, who tak...	['Marvel Entertainment', 'Columbia Pictures', ...
3	5587	466420	Killers of the Flower Moon	2023-10-06	['Crime', 'Drama', 'Thriller']	English	14.445	When oil is discovered in 1920s Oklahoma under...	['Appian Way', 'Imperative Entertainment', 'Si...
4	8260	807172	The Exorcist: Believer	2023-10-11	['Horror', 'Thriller']	English	12.623	The father of a possessed child seeks out the ...	['Blumhouse Productions', 'Morgan Creek Produc...

Released Movies

In [10]:
```
released_movies_df = df.copy()

released_movies_df.sort_values(by=['release_date'], inplace=True, ascending=False)
```

```
released_movies = released_movies_df.loc[(df['release_date'] <= '2023-09-14')].reset_index()
released_movies.head()
```

Out[10]:

	index	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget	p
0	2125	820525	After Everything	2023-09-13	['Romance', 'Drama']	English	0.0	0	36.319	The sequel to 'After Ever Happy' (2022), which...	14000000	
1	854	968051	The Nun II	2023-09-05	['Horror', 'Mystery']	English	0.0	0	45.084	Four years after the events at the Abbey of St...	0	
2	1062	926393	The Equalizer 3	2023-08-30	['Action', 'Thriller', 'Crime']	English	0.0	0	48.772	Robert McCall finds himself at home in Souther...	0	
3	217	912908	Strays	2023-08-17	['Comedy', 'Adventure']	English	0.0	0	146.598	When Reggie, a naïve, relentlessly optimistic ...	0	
4	3476	565770	Blue Beetle	2023-08-16	['Action', 'Science Fiction']	English	0.0	0	22.946	Recent college grad Jaime Reyes returns home f...	120000000	[

In [11]:
```
# movies language
df.original_language.unique()
```

Out[11]:
```
array(['English', 'French', 'Dutch', 'Spanish', 'Korean', 'Japanese',
       'Finnish', 'Ukrainian', 'Norwegian', 'Estonian', 'cn', 'Polish',
       'Russian', 'German', 'Chinese', 'Italian', 'Basque', 'Thai',
       'Turkish', 'Swedish', 'Icelandic', 'Tagalog', 'Bengali', 'Arabic',
       'Tamil', 'Telugu', 'Romanian', 'Indonesian', 'Galician', 'Danish',
       'Macedonian', 'Portuguese', 'Vietnamese', 'Catalan', 'Hindi',
       'Persian', 'Hebrew', 'Serbian', 'Malayalam', 'Greek', 'Hungarian',
       'Czech', 'Norwegian Bokmal', 'xx', 'Kannada', 'Irish', 'Khmer',
       'sh', 'Dzongkha', 'Panjabi', 'Sundanese'], dtype=object)
```

Analyze movies according to original language

In [12]:
```
# Language and total movies
df.original_language.value_counts()
```

Out[12]:

original_language	
English	7229
Japanese	697
Korean	388
Spanish	336
French	298
Chinese	151
Italian	148
cn	134
German	83
Russian	69
Tagalog	46
Hindi	36
Norwegian	32
Danish	30
Portuguese	29
Polish	28
Thai	26
Dutch	24
Swedish	23
Indonesian	17
Turkish	17
Tamil	9
Telugu	9
Finnish	6
Arabic	4
Romanian	4
Ukrainian	4
Greek	4
Malayalam	3
Basque	3
Vietnamese	3
Hungarian	3
Persian	3
Khmer	2
Kannada	2
xx	2
Czech	2
Serbian	2
Hebrew	2

```
Catalan                2
Galician               2
Icelandic              2
Bengali                1
Macedonian             1
Norwegian Bokmal       1
Estonian               1
Irish                  1
sh                     1
Dzongkha               1
Panjabi                1
Sundanese              1
Name: count, dtype: int64
```

In [13]:

```
# language and total movies in dataframe
language_count = df.original_language.value_counts().to_frame().reset_index()
language_count.columns = ["original_language","total"]
language_count
```

Out[13]:

	original_language	total
0	English	7229
1	Japanese	697
2	Korean	388
3	Spanish	336
4	French	298
5	Chinese	151
6	Italian	148
7	cn	134
8	German	83
9	Russian	69
10	Tagalog	46
11	Hindi	36
12	Norwegian	32
13	Danish	30
14	Portuguese	29
15	Polish	28
16	Thai	26
17	Dutch	24
18	Swedish	23
19	Indonesian	17
20	Turkish	17
21	Tamil	9

	original_language	total
22	Telugu	9
23	Finnish	6
24	Arabic	4
25	Romanian	4
26	Ukrainian	4
27	Greek	4
28	Malayalam	3
29	Basque	3
30	Vietnamese	3
31	Hungarian	3
32	Persian	3
33	Khmer	2
34	Kannada	2
35	xx	2
36	Czech	2
37	Serbian	2
38	Hebrew	2
39	Catalan	2
40	Galician	2
41	Icelandic	2
42	Bengali	1
43	Macedonian	1

	original_language	total
44	Norwegian Bokmal	1
45	Estonian	1
46	Irish	1
47	sh	1
48	Dzongkha	1
49	Panjabi	1
50	Sundanese	1

There are a lot more english entries, as expected, so we will analyse the overall number of movies in two ways: comparing english movies with the rest and comparing the data without the english spoken movies.

In [14]:
```
#total english movies and all other languages movies
language_total = df["original_language"].value_counts()
total = language_total.sum()
total_english = language_total.loc["English"]
total_others = total - total_english
print(total_english, total_others)
```

```
7229 2694
```

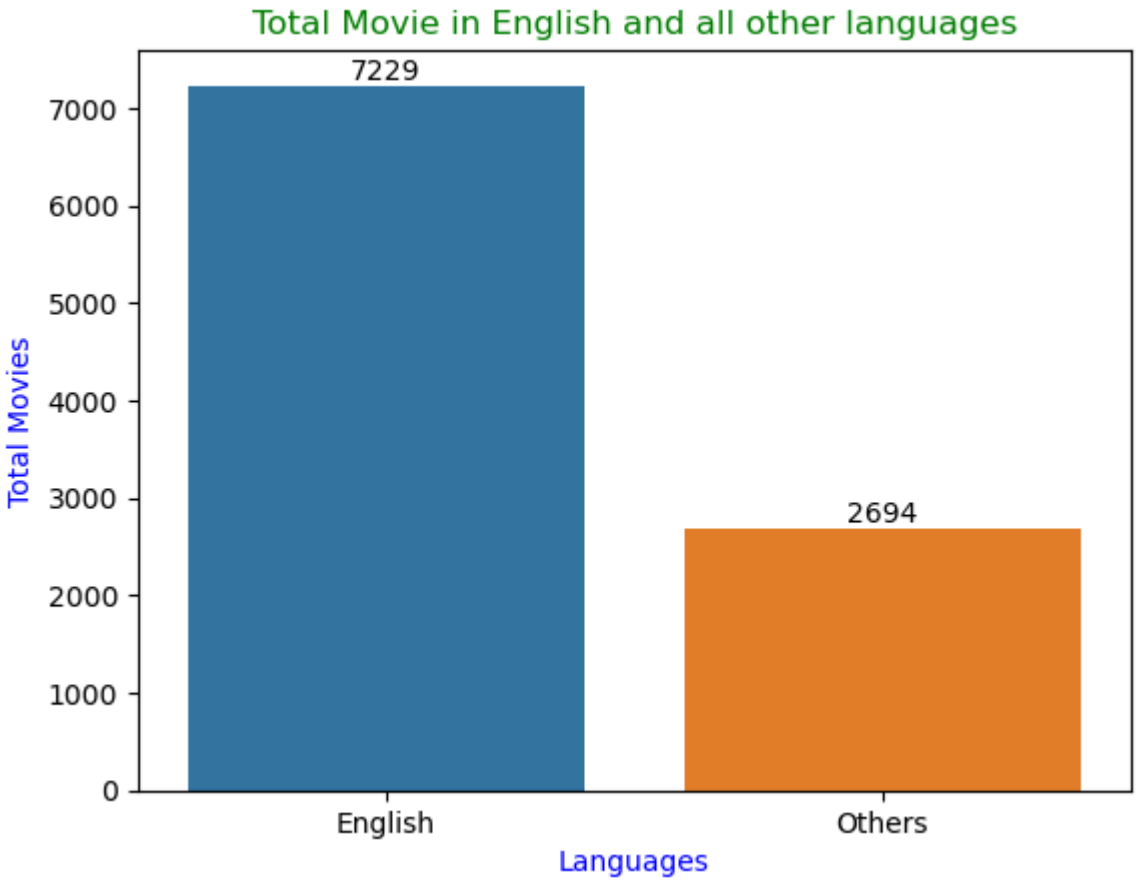
In [15]:
```
# English language and Other language in dataframe
eng_and_others = {'language' : ['English','Others'],
        'total' : [total_english, total_others]}
eng_and_others = pd.DataFrame(eng_and_others)
eng_and_others
```

Out[15]:

	language	total
0	English	7229
1	Others	2694

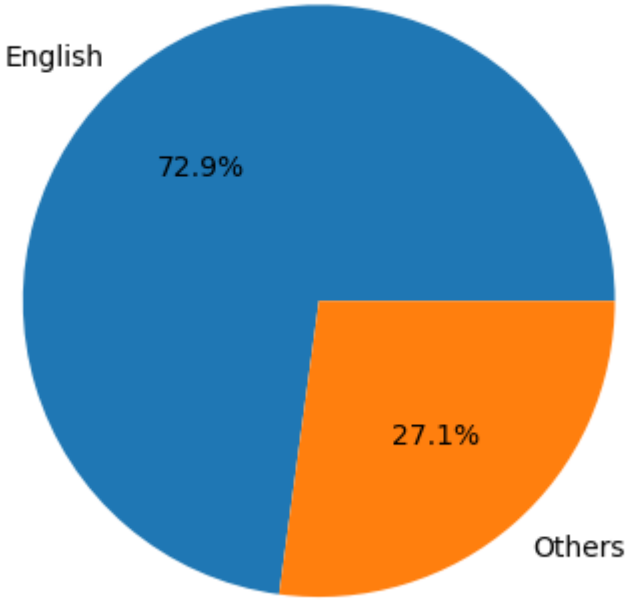
In [16]:

```
# English and other languages in barplot
ax = sns.barplot(data = eng_and_others, x = 'language', y='total')

# show values on bar
for i in ax.containers:
    ax.bar_label(i,)

# title and labels
plt.title('Total Movie in English and all other languages',fontsize = 12, color='green')
plt.xlabel('Languages',fontsize = 10,color='blue')
plt.ylabel('Total Movies',fontsize = 10,color='blue')

plt.show()
```



In [17]:
```
#Total Movies in English and all other languages with percentage
# pieplot
plt.pie(eng_and_others.total, labels = eng_and_others.language,  autopct='%1.1f%%')

# for title
plt.title('Total Movies in English and all other languages with percentage',fontsize = 10, color='green')
plt.show()
```

Total Movies in English and all other languages with percentage



As we can see by the visualizations, the number of english speaking movies is too high compared to the other, so it's interesting to analyse the languages apart from english.

All languages movies count

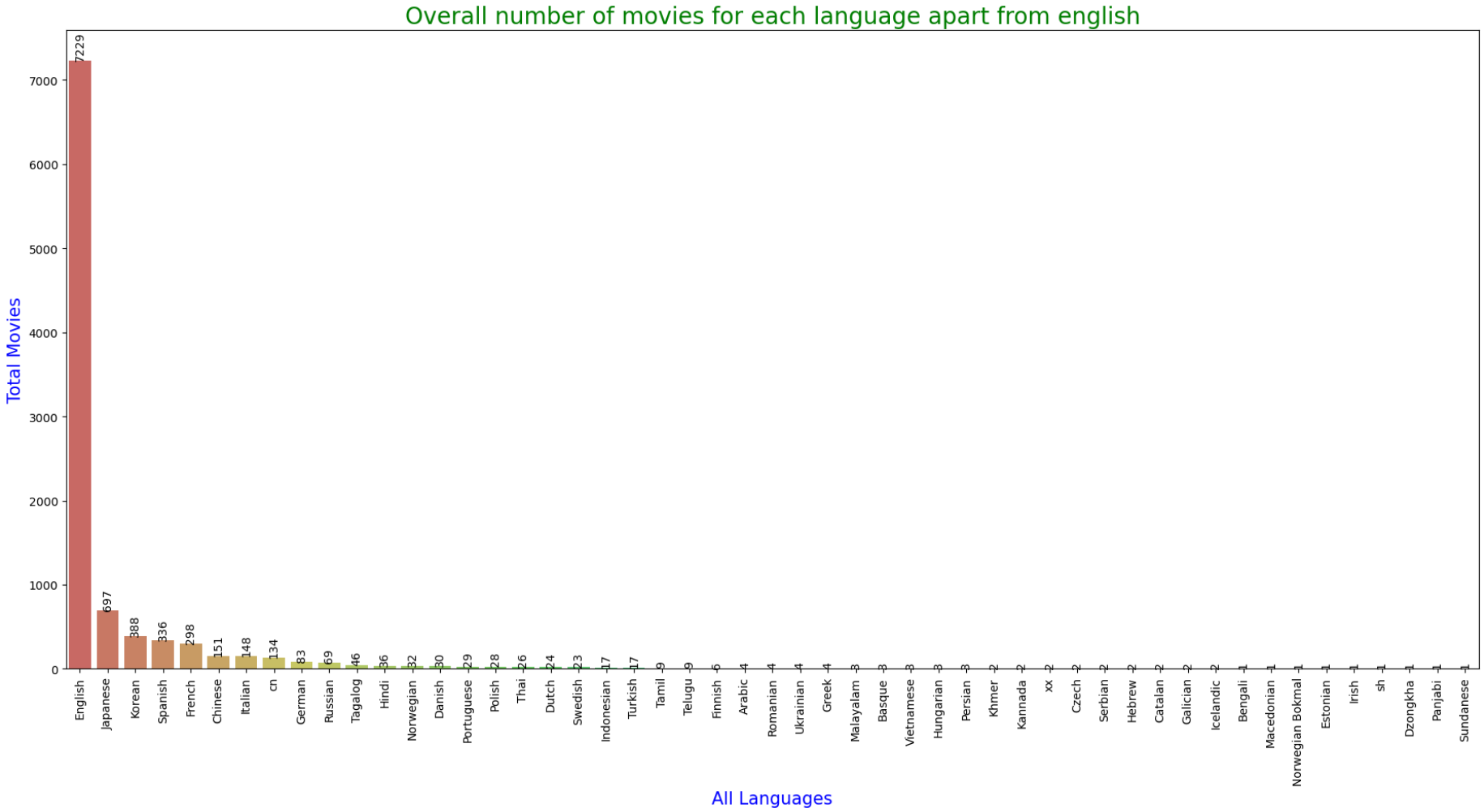
In [18]:
```
# all languages movie count
all_lang_movies_count = df.original_language.value_counts().to_frame().reset_index()
all_lang_movies_count.columns = ["original_language","total"]
all_lang_movies_count.head()
```

Out[18]:

	original_language	total
0	English	7229
1	Japanese	697
2	Korean	388
3	Spanish	336
4	French	298

In [19]:

```
plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'total', data = all_lang_movies_count, palette = 'hls')

# bar labels in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90)

# titel and lables
plt.title('Overall number of movies for each language apart from english',fontsize = 20, color='green')
plt.xlabel('All Languages',fontsize = 15,color='blue')
plt.ylabel('Total Movies',fontsize = 15,color='blue')
plt.xticks(rotation = 90)

plt.show()
```



All languages movies count except English

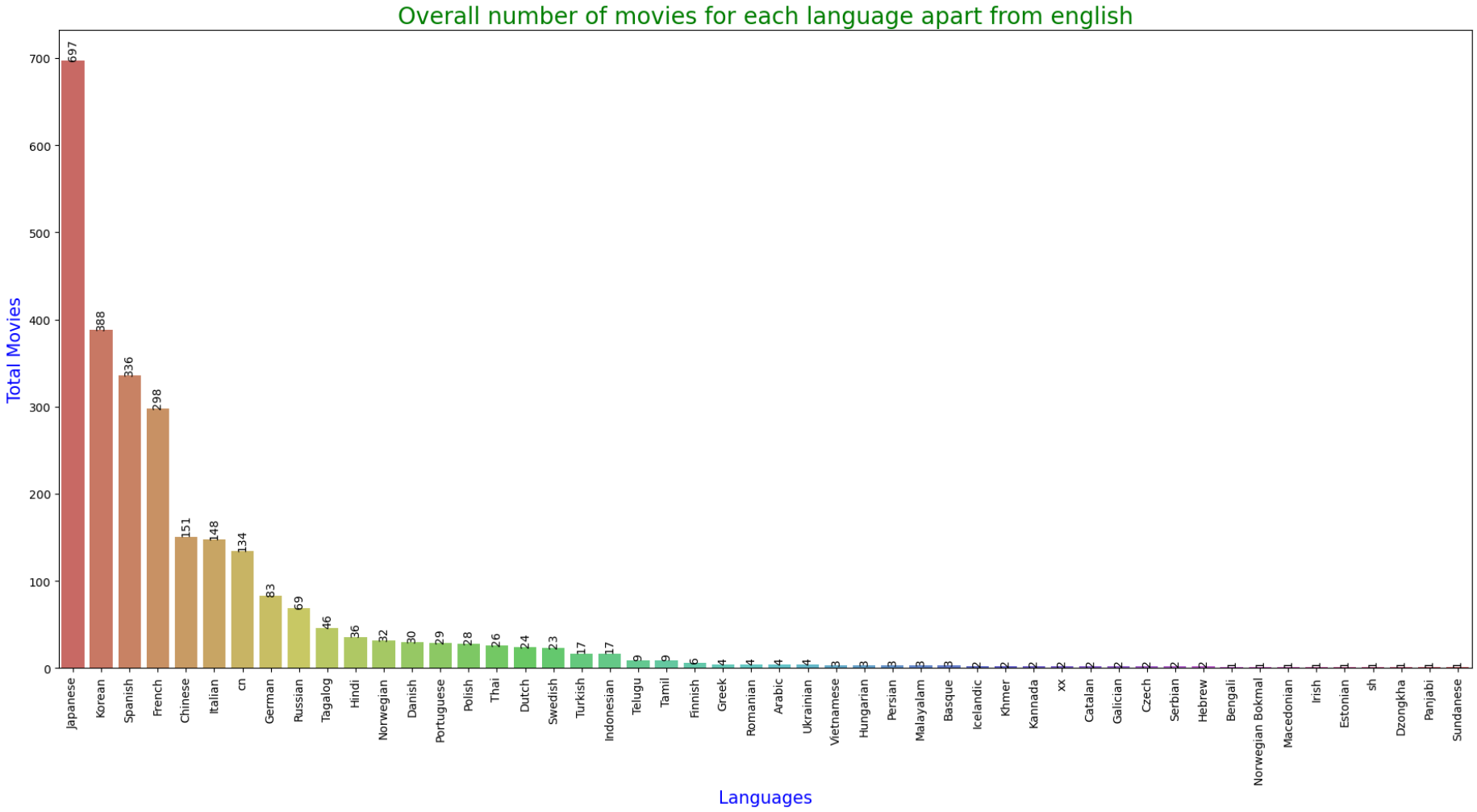
In [20]:
```
# all languages movie except english
non_english = df.query("original_language != 'English'").original_language.value_counts().to_frame().reset_index()
non_english.columns = ["original_language","total"]
non_english.head()
```

Out[20]:

	original_language	total
0	Japanese	697
1	Korean	388
2	Spanish	336
3	French	298
4	Chinese	151

In [21]:

```
plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'total', data = non_english, palette = 'hls')

# bar labels in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90)

# title and lables
plt.title('Overall number of movies for each language apart from english',fontsize = 20, color='green')
plt.xlabel('Languages',fontsize = 15,color='blue')
plt.ylabel('Total Movies',fontsize = 15,color='blue')
plt.xticks(rotation = 90)

plt.show()
```



The first five are: Japanese, Korean, Spanish, French, Chinese. Those are the languages with the most movies in this dataframe, apart from english.

Analysing the budget of the movies for each language

In [22]:
```
df.budget.describe()
```

Out[22]:
```
count    9.923000e+03
mean     1.963073e+07
std      3.891562e+07
min      0.000000e+00
25%      0.000000e+00
50%      5.650000e+05
75%      2.200000e+07
max      5.793304e+08
Name: budget, dtype: float64
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['id', 'title', 'release_date', 'genres', 'original_language',
       'vote_average', 'vote_count', 'popularity', 'overview', 'budget',
       'production_companies', 'revenue', 'runtime'],
      dtype='object')
```

In [24]:
```
# budget and all languages columns
budget_lan = df[['budget','original_language']]
budget_lan
```

Out[24]:

	budget	original_language
0	18000000	English
1	200000000	English
2	100000000	English
3	0	English
4	125000000	English
...	...	...
9995	0	English
9996	0	English
9997	0	English
9998	0	Korean
9999	5400000	English

9923 rows × 2 columns

index 3 and other's budget is not NaN. during entry values given as 0.

In [25]:

```
# drop 0 budget rows
budget_lan_drop = budget_lan.query('budget == 0').index
budget_lan.drop(budget_lan_drop,axis=0,inplace = True)
budget_lan
```

```
C:\Users\Umair Ali\AppData\Local\Temp\ipykernel_8724\4214584850.py:3: 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-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  budget_lan.drop(budget_lan_drop,axis=0,inplace = True)
```

Out[25]:

	budget	original_language
0	18000000	English
1	200000000	English
2	100000000	English
4	125000000	English
5	460000000	English
...	...	...
9979	786801	Italian
9980	12000000	English
9982	700000	English
9989	7000000	English
9999	5400000	English

5134 rows × 2 columns

In [26]:

```
# sum of budget by Language
total_budget_lan = budget_lan.groupby("original_language").sum().reset_index()
total_budget_lan_sorted = total_budget_lan.sort_values("budget",ascending=False).reset_index()
total_budget_lan_sorted
```

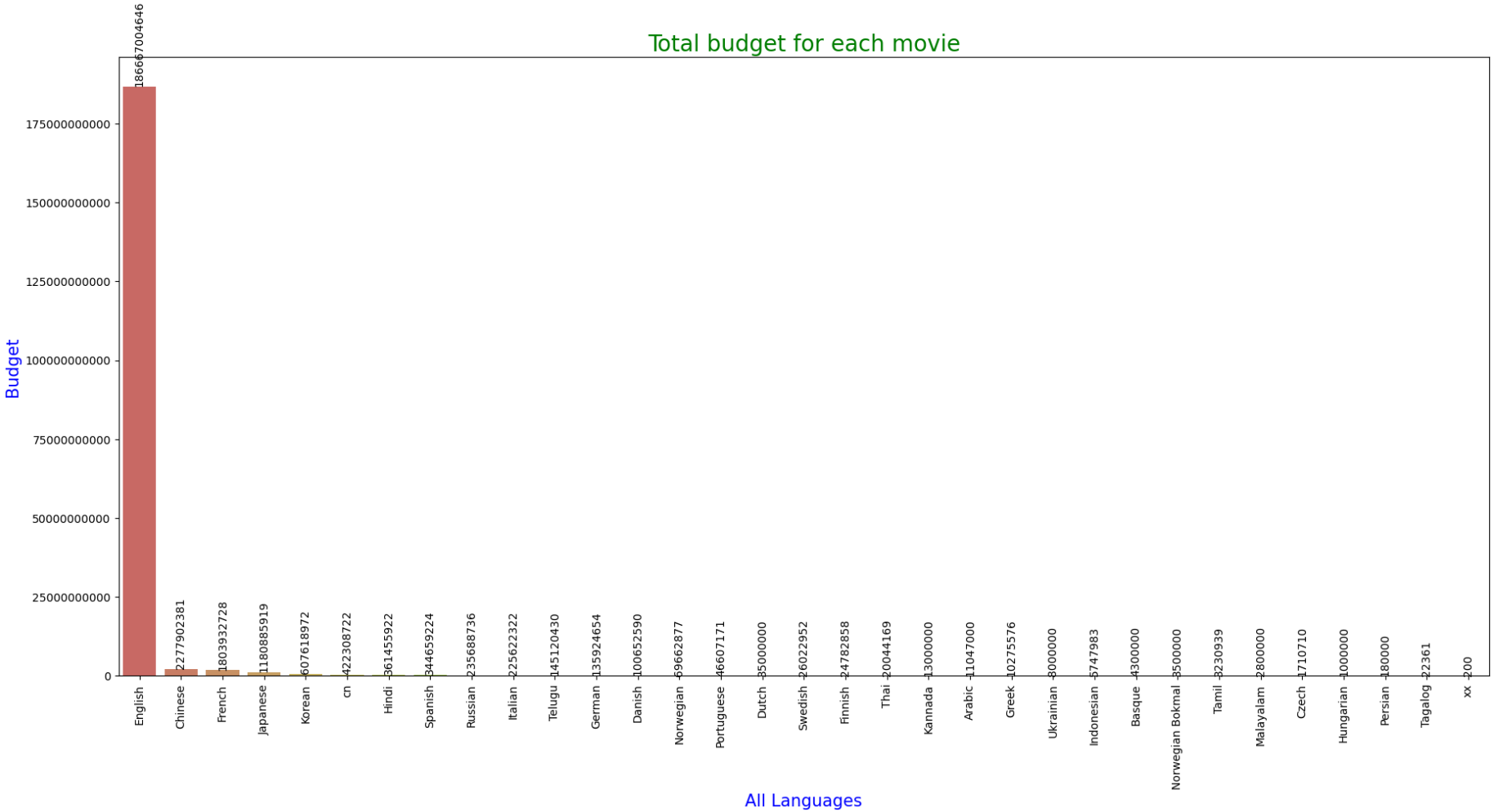
Out[26]:

	index	original_language	budget
0	6	English	186667004646
1	2	Chinese	2277902381
2	8	French	1803932728
3	15	Japanese	1180885919
4	17	Korean	607618972
5	31	cn	422308722
6	11	Hindi	361455922
7	24	Spanish	344659224
8	23	Russian	235688736
9	14	Italian	225622322
10	28	Telugu	145120430
11	9	German	135924654
12	4	Danish	100652590
13	19	Norwegian	69662877
14	22	Portuguese	46607171
15	5	Dutch	35000000
16	25	Swedish	26022952
17	7	Finnish	24782858
18	29	Thai	20044169
19	16	Kannada	13000000
20	0	Arabic	11047000
21	10	Greek	10275576

	index	original_language	budget
22	30	Ukrainian	8000000
23	13	Indonesian	5747983
24	1	Basque	4300000
25	20	Norwegian Bokmal	3500000
26	27	Tamil	3230939
27	18	Malayalam	2800000
28	3	Czech	1710710
29	12	Hungarian	1000000
30	21	Persian	180000
31	26	Tagalog	22361
32	32	xx	200

In [27]:

```
plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'budget', data = total_budget_lan_sorted, palette = 'hls')

# y-ticks values in complete figures
plt.ticklabel_format(style='plain', axis='y')

# bar labels/values in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90)


plt.title('Total budget for each movie',fontsize = 20, color='green')
plt.xlabel('All Languages',fontsize = 15,color='blue')
plt.ylabel('Budget',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



As expected, the total budget of the english speaking movies is too high, because they have too much entries. We will also analyse the total budget from the movies excluding the english ones as well.

In [28]:
```
# total budget except english language
total_budget_non_english = budget_lan.query("original_language != 'English'")
total_budget_non_english.head()
```

Out[28]:

	budget	original_language
25	12300000	Spanish
48	6200000	Finnish
67	1000000	Ukrainian
86	15800000	Japanese
124	20000000	German

In [29]:

```
# total budget except english language in ascending order
total_budget_non_english_sorted = total_budget_non_english.groupby("original_language").sum().reset_index()
total_budget_non_english_sorted = total_budget_non_english_sorted.sort_values("budget",ascending=False).reset_index()
total_budget_non_english_sorted.head()
```

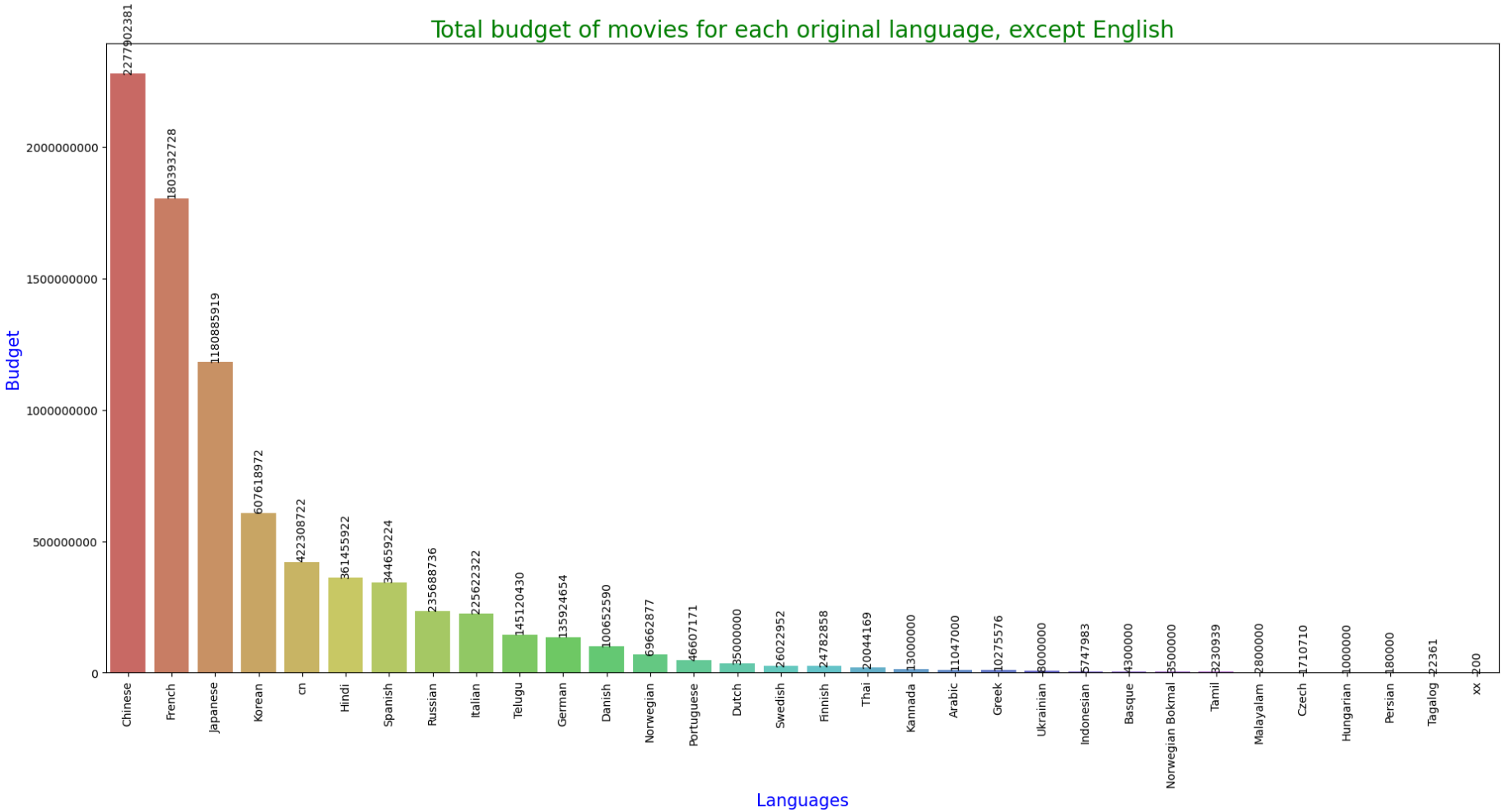
Out[29]:

	index	original_language	budget
0	2	Chinese	2277902381
1	7	French	1803932728
2	14	Japanese	1180885919
3	16	Korean	607618972
4	30	cn	422308722

In [30]:

```
#plot of total budget except english

plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'budget', data = total_budget_non_english_sorted, palette = 'hls')

# y_ticks values/labels in complete figures
plt.ticklabel_format(style='plain', axis='y')

# bar values/labels in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90)
```

```python
#title and labels
plt.title('Total budget of movies for each original language, except English',fontsize = 20, color='green')
plt.xlabel('Languages',fontsize = 15,color='blue')
plt.ylabel('Budget',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



The first five are: Chinese, French, Japanese, Korean, and cn. Those are the languages with the most budget movies in this database, apart from english.

In [31]:
```
# mean/average budget of all languages

budget_lan_mean = budget_lan.groupby("original_language").mean().reset_index()
budget_lan_mean_sort = budget_lan_mean.sort_values("budget",ascending=False).reset_index()
budget_lan_mean_sort
```

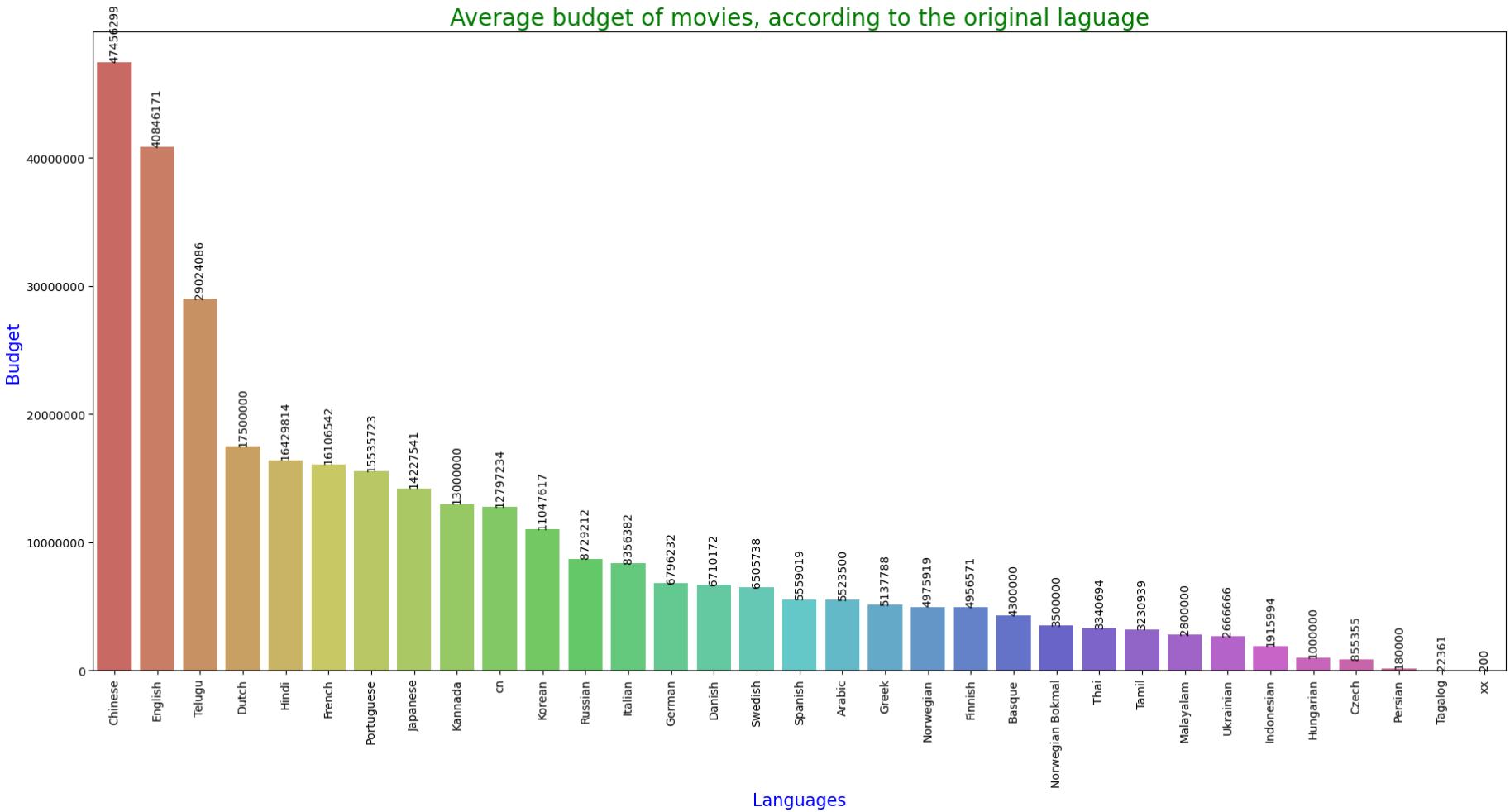
Out[31]:

	index	original_language	budget
0	2	Chinese	4.745630e+07
1	6	English	4.084617e+07
2	28	Telugu	2.902409e+07
3	5	Dutch	1.750000e+07
4	11	Hindi	1.642981e+07
5	8	French	1.610654e+07
6	22	Portuguese	1.553572e+07
7	15	Japanese	1.422754e+07
8	16	Kannada	1.300000e+07
9	31	cn	1.279723e+07
10	17	Korean	1.104762e+07
11	23	Russian	8.729212e+06
12	14	Italian	8.356382e+06
13	9	German	6.796233e+06
14	4	Danish	6.710173e+06
15	25	Swedish	6.505738e+06
16	24	Spanish	5.559020e+06
17	0	Arabic	5.523500e+06
18	10	Greek	5.137788e+06
19	19	Norwegian	4.975920e+06
20	7	Finnish	4.956572e+06
21	1	Basque	4.300000e+06

	index	original_language	budget
22	20	Norwegian Bokmal	3.500000e+06
23	29	Thai	3.340695e+06
24	27	Tamil	3.230939e+06
25	18	Malayalam	2.800000e+06
26	30	Ukrainian	2.666667e+06
27	13	Indonesian	1.915994e+06
28	12	Hungarian	1.000000e+06
29	3	Czech	8.553550e+05
30	21	Persian	1.800000e+05
31	26	Tagalog	2.236100e+04
32	32	xx	2.000000e+02

In [32]:

```
# plot # mean/average budget of all languages

plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'budget', data = budget_lan_mean_sort, palette = 'hls')

# y_ticks values/budget in complete figures
plt.ticklabel_format(style='plain', axis='y')
# bar labels in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90)

# title and labels
plt.title('Average budget of movies, according to the original laguage',fontsize = 20, color='green')
plt.xlabel('Languages',fontsize = 15,color='blue')
plt.ylabel('Budget',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



If we compare the average budget for each language, there is no need to separate languages, because no one have a really outstanding average. The first five are: Chinese, English, Telugu, Dutuch, and Hindi.

Analysing the average voting grade for each original language

In [33]:
```
df.vote_average.describe()
```

Out[33]:
```
count    9923.000000
mean        6.348201
std         1.371006
min         0.000000
25%         5.900000
50%         6.500000
75%         7.100000
max        10.000000
Name: vote_average, dtype: float64
```

We can see that more than 50% of the movies have a higher average voting than the mean, but just 25% of the movies have a high grade that is 5.9

In [34]:
```
# get vote and language column
vote_lan = df[['vote_average','original_language']]
vote_lan.head()
```

Out[34]:

	vote_average	original_language
0	7.4	English
1	6.6	English
2	7.5	English
3	7.2	English
4	6.8	English

In [35]:
```
# average/mean of vote by language, and sort by ascending order
vote_lan_mean = vote_lan.groupby("original_language").mean().reset_index()
vote_lan_mean_sorted = vote_lan_mean.sort_values("vote_average",ascending=False).reset_index()
vote_lan_mean_sorted
```

Out[35]:

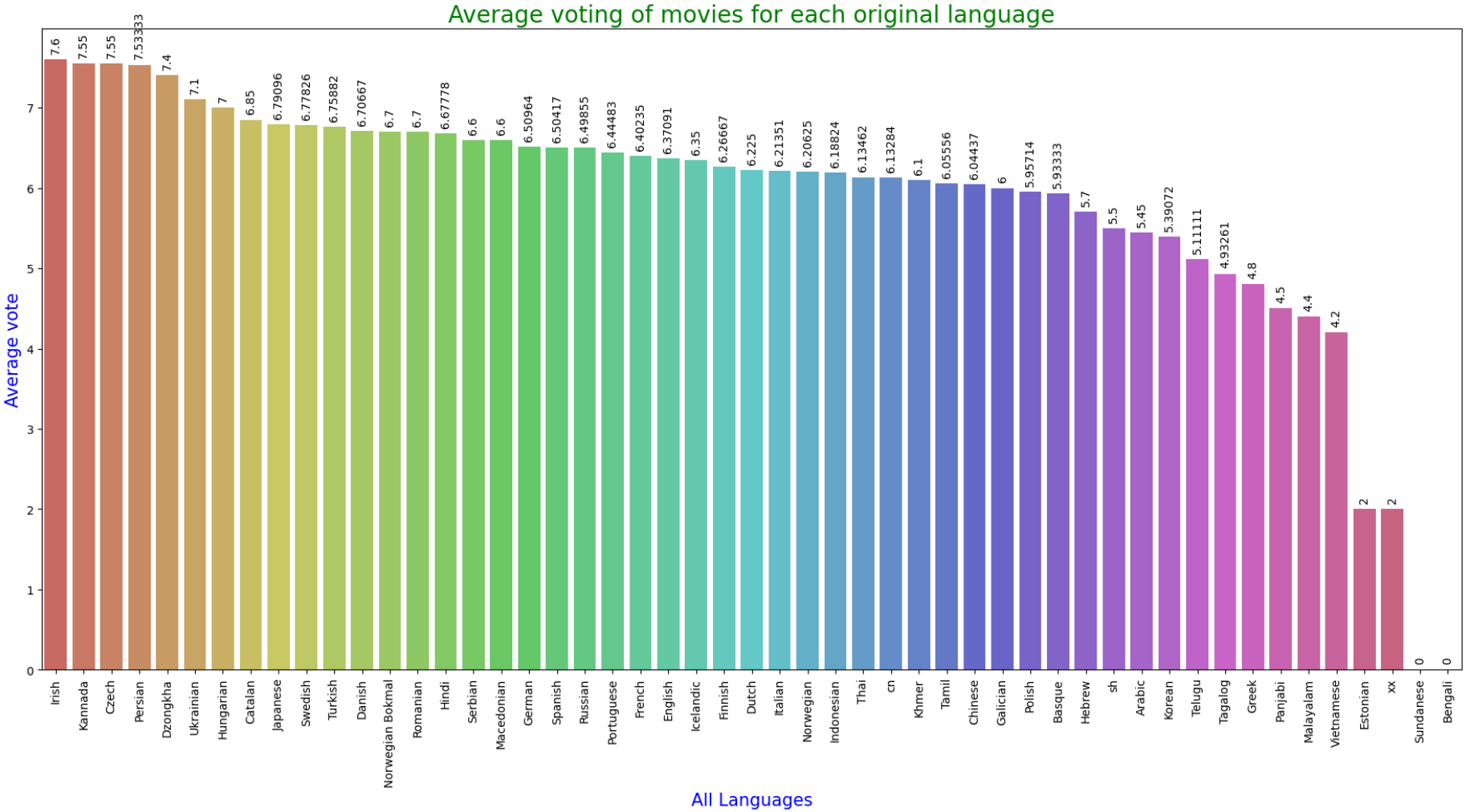
	index	original_language	vote_average
0	21	Irish	7.600000
1	24	Kannada	7.550000
2	5	Czech	7.550000
3	32	Persian	7.533333
4	8	Dzongkha	7.400000
5	46	Ukrainian	7.100000
6	18	Hungarian	7.000000
7	3	Catalan	6.850000
8	23	Japanese	6.790961
9	40	Swedish	6.778261
10	45	Turkish	6.758824
11	6	Danish	6.706667
12	30	Norwegian Bokmal	6.700000
13	35	Romanian	6.700000
14	17	Hindi	6.677778
15	37	Serbian	6.600000
16	27	Macedonian	6.600000
17	14	German	6.509639
18	38	Spanish	6.504167
19	36	Russian	6.498551
20	34	Portuguese	6.444828
21	12	French	6.402349

	index	original_language	vote_average
22	9	English	6.370909
23	19	Icelandic	6.350000
24	11	Finnish	6.266667
25	7	Dutch	6.225000
26	22	Italian	6.213514
27	29	Norwegian	6.206250
28	20	Indonesian	6.188235
29	44	Thai	6.134615
30	48	cn	6.132836
31	25	Khmer	6.100000
32	42	Tamil	6.055556
33	4	Chinese	6.044371
34	13	Galician	6.000000
35	33	Polish	5.957143
36	1	Basque	5.933333
37	16	Hebrew	5.700000
38	49	sh	5.500000
39	0	Arabic	5.450000
40	26	Korean	5.390722
41	43	Telugu	5.111111
42	41	Tagalog	4.932609
43	15	Greek	4.800000

	index	original_language	vote_average
44	31	Panjabi	4.500000
45	28	Malayalam	4.400000
46	47	Vietnamese	4.200000
47	10	Estonian	2.000000
48	50	xx	2.000000
49	39	Sundanese	0.000000
50	2	Bengali	0.000000

In [36]:

```
#vote average plot
plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'vote_average', data = vote_lan_mean_sorted, palette = 'hls')

# show values on bar
for i in ax.containers:
    ax.bar_label(i,rotation = 90, padding=5)

# title and labels
plt.title('Average voting of movies for each original language',fontsize = 20, color='green')
plt.xlabel('All Languages',fontsize = 15,color='blue')
plt.ylabel('Average vote',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



They must really love that movie in Irish, because it is the leader in here as well. The other four higher are indonesian, hebrew, persian and arabic, again, mostly languages with few total entries in the data.

Popularity

In [37]:
```
df.popularity.describe()
```

Out[37]:
```
count    9923.000000
mean       31.550217
std       111.857228
min         7.219000
25%        13.541500
50%        17.626000
75%        27.175500
max      5089.969000
Name: popularity, dtype: float64
```

Popularity is a strange number, but seems like the higher the number, the most popular the movie is. The max value of popularity is much more higher than the average, but there are around 25% of the movies that have a higher popularity.

In [38]:
```
#get popularity and orignal_language column
pop_lan = df[['popularity','original_language']]
pop_lan.head()
```

Out[38]:

	popularity	original_language
0	5089.969	English
1	4665.438	English
2	3935.550	English
3	2791.532	English
4	2702.593	English

In [39]:
```
# average/mean of popularity by language, and sort by ascending order
pop_lan_mean = pop_lan.groupby("original_language").mean().reset_index()
pop_lan_mean_sorted = pop_lan_mean.sort_values("popularity",ascending=False).reset_index()
pop_lan_mean_sorted
```

Out[39]:

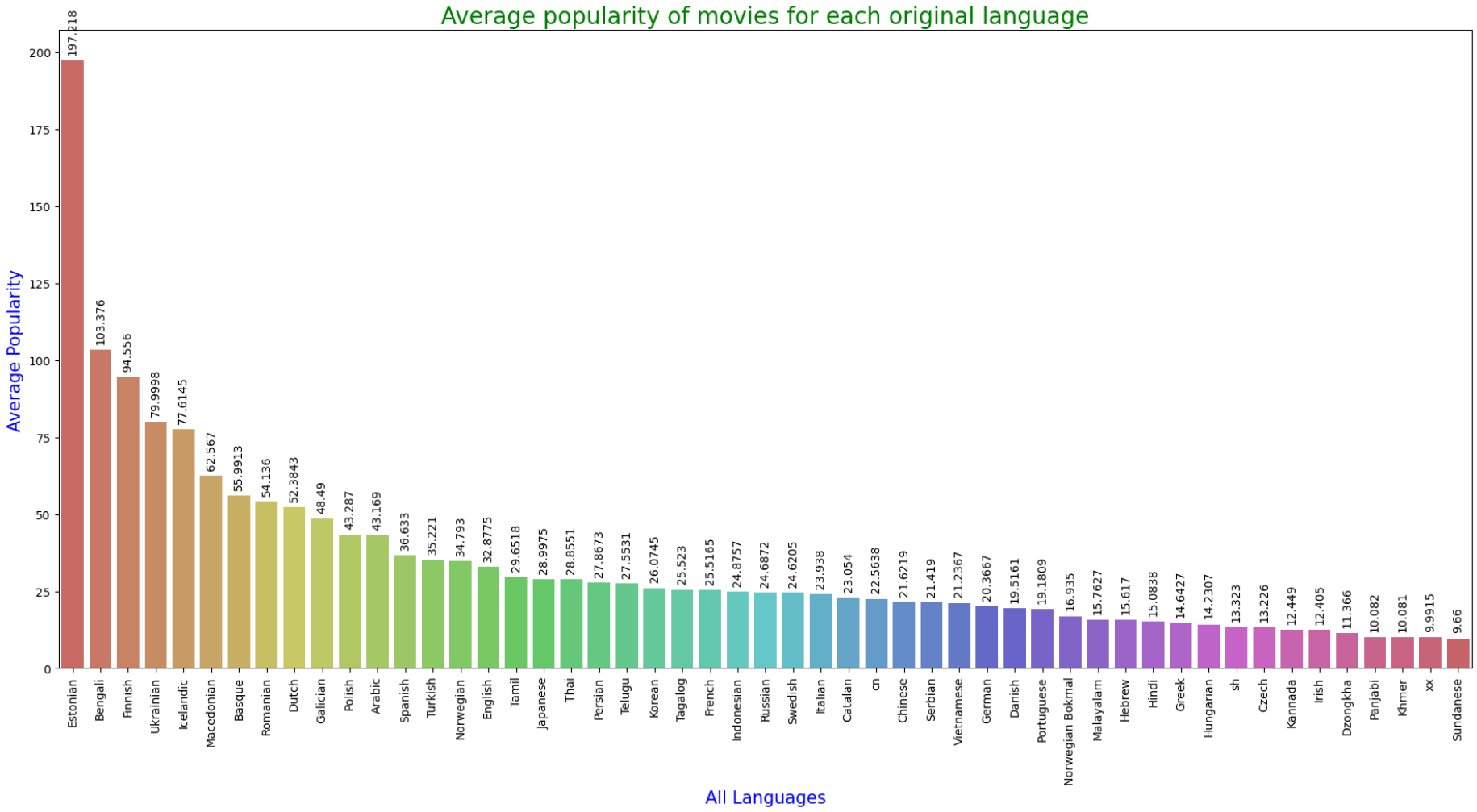
	index	original_language	popularity
0	10	Estonian	197.218000
1	2	Bengali	103.376000
2	11	Finnish	94.556000
3	46	Ukrainian	79.999750
4	19	Icelandic	77.614500
5	27	Macedonian	62.567000
6	1	Basque	55.991333
7	35	Romanian	54.136000
8	7	Dutch	52.384250
9	13	Galician	48.490000
10	33	Polish	43.287036
11	0	Arabic	43.169000
12	38	Spanish	36.633024
13	45	Turkish	35.221000
14	29	Norwegian	34.792969
15	9	English	32.877477
16	42	Tamil	29.651778
17	23	Japanese	28.997465
18	44	Thai	28.855077
19	32	Persian	27.867333
20	43	Telugu	27.553111
21	26	Korean	26.074518

	index	original_language	popularity
22	41	Tagalog	25.523043
23	12	French	25.516510
24	20	Indonesian	24.875706
25	36	Russian	24.687174
26	40	Swedish	24.620478
27	22	Italian	23.938014
28	3	Catalan	23.054000
29	48	cn	22.563799
30	4	Chinese	21.621868
31	37	Serbian	21.419000
32	47	Vietnamese	21.236667
33	14	German	20.366663
34	6	Danish	19.516133
35	34	Portuguese	19.180862
36	30	Norwegian Bokmal	16.935000
37	28	Malayalam	15.762667
38	16	Hebrew	15.617000
39	17	Hindi	15.083750
40	15	Greek	14.642750
41	18	Hungarian	14.230667
42	49	sh	13.323000
43	5	Czech	13.226000

	index	original_language	popularity
44	24	Kannada	12.449000
45	21	Irish	12.405000
46	8	Dzongkha	11.366000
47	31	Panjabi	10.082000
48	25	Khmer	10.081000
49	50	xx	9.991500
50	39	Sundanese	9.660000

In [40]:

```
# mean/average popularity of orignal languages
plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'popularity', data = pop_lan_mean_sorted, palette = 'hls')

# values on bar
for i in ax.containers:
    ax.bar_label(i,rotation = 90, padding=5)

# title and labels
plt.title('Average popularity of movies for each original language',fontsize = 20, color='green')
plt.xlabel('All Languages',fontsize = 15,color='blue')
plt.ylabel('Average Popularity',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



Here we have a top 5 consisting of: Estonian, Bengali, Finnish, Ukrainian, and Icelandic. I expected to see the average of the english movies to drop because of the high number of entries, but if they got a good average, must be because they have lots of popular movies in english.

Revenue

In [41]:
```
df.revenue.describe()
```

Out[41]:
```
count    9.923000e+03
mean     6.023300e+07
std      1.548025e+08
min      0.000000e+00
25%      0.000000e+00
50%      1.165882e+06
75%      5.000625e+07
max      2.923706e+09
Name: revenue, dtype: float64
```

The average of revenue is much more higher than the median. That is also expected, mostly because the data has too much movies in english, and they have a tendency to have high profit movies in this language.

In [42]:
```
# Get revenues and orignal_language
rev_lan = df[['revenue','original_language']]
rev_lan.head()
```

Out[42]:

	revenue	original_language
0	65675816	English
1	464566092	English
2	1121048165	English
3	0	English
4	133437105	English

In [43]:
```
# drop 0 revenue rows
rev_lan_drop = rev_lan.query('revenue == 0').index
rev_lan.drop(rev_lan_drop,axis=0,inplace = True)
rev_lan
```

```
C:\Users\Umair Ali\AppData\Local\Temp\ipykernel_8724\3560680913.py:3: 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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  rev_lan.drop(rev_lan_drop,axis=0,inplace = True)
```

Out[43]:

	revenue	original_language
0	65675816	English
1	464566092	English
2	1121048165	English
4	133437105	English
5	2319331580	English
...	...	...
9986	32511047	English
9989	21600000	English
9990	29066681	English
9995	15793051	English
9999	12055868	English

5423 rows × 2 columns

In [44]:

```
# average/mean of revenue by language, and sort by ascending order
rev_lan_mean = rev_lan.groupby("original_language").mean().reset_index()
rev_lan_mean_sorted = rev_lan_mean.sort_values("revenue",ascending=False).reset_index()
rev_lan_mean_sorted
```

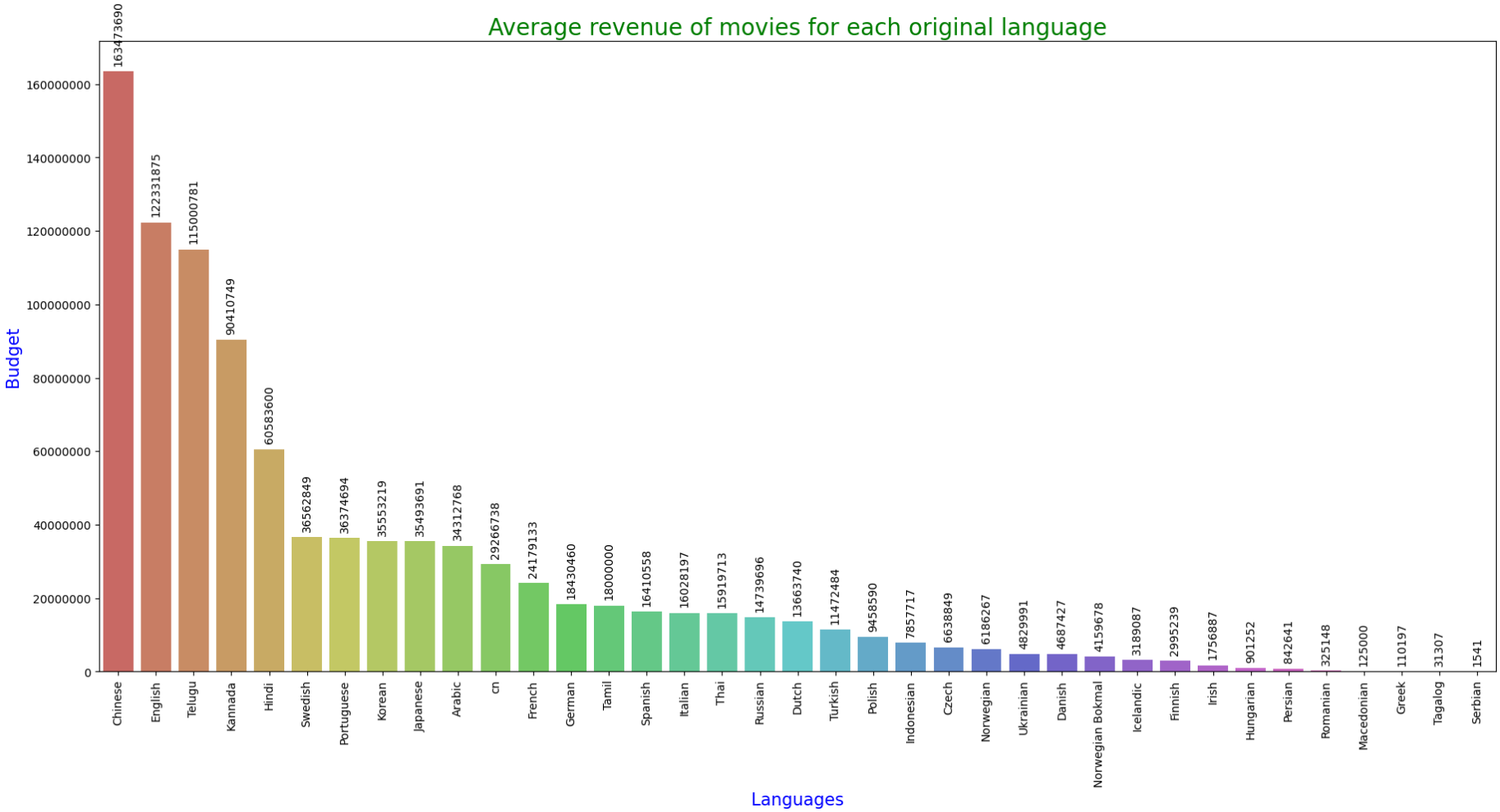
Out[44]:

	index	original_language	revenue
0	1	Chinese	1.634737e+08
1	5	English	1.223319e+08
2	32	Telugu	1.150008e+08
3	17	Kannada	9.041075e+07
4	10	Hindi	6.058360e+07
5	29	Swedish	3.656285e+07
6	24	Portuguese	3.637469e+07
7	18	Korean	3.555322e+07
8	16	Japanese	3.549369e+07
9	0	Arabic	3.431277e+07
10	36	cn	2.926674e+07
11	7	French	2.417913e+07
12	8	German	1.843046e+07
13	31	Tamil	1.800000e+07
14	28	Spanish	1.641056e+07
15	15	Italian	1.602820e+07
16	33	Thai	1.591971e+07
17	26	Russian	1.473970e+07
18	4	Dutch	1.366374e+07
19	34	Turkish	1.147248e+07
20	23	Polish	9.458590e+06
21	13	Indonesian	7.857718e+06

	index	original_language	revenue
22	2	Czech	6.638849e+06
23	20	Norwegian	6.186268e+06
24	35	Ukrainian	4.829992e+06
25	3	Danish	4.687428e+06
26	21	Norwegian Bokmal	4.159678e+06
27	12	Icelandic	3.189087e+06
28	6	Finnish	2.995239e+06
29	14	Irish	1.756887e+06
30	11	Hungarian	9.012520e+05
31	22	Persian	8.426417e+05
32	25	Romanian	3.251480e+05
33	19	Macedonian	1.250000e+05
34	9	Greek	1.101970e+05
35	30	Tagalog	3.130700e+04
36	27	Serbian	1.541000e+03

In [45]:
```
# average/mean revenue of all languages

plt.figure(figsize=(22,10))
ax = sns.barplot(x = 'original_language', y = 'revenue', data = rev_lan_mean_sorted, palette = 'hls')

# y_ticks values in complete figure
plt.ticklabel_format(style='plain', axis='y')

# values on bar in complete figures
ax.bar_label(ax.containers[0], fmt = '%d',rotation = 90,padding=5)
```

```python
# title and labels
plt.title('Average revenue of movies for each original language',fontsize = 20, color='green')
plt.xlabel('Languages',fontsize = 15,color='blue')
plt.ylabel('Budget',fontsize = 15,color='blue')
plt.xticks(rotation = 90)
plt.show()
```



Finding other movies from unkown languages in the last barplot

In [46]:
```
xx = df.query("original_language == 'xx'")
xx
```

Out[46]:

	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget	production_con
6340	617932	Barbie	1977-01-01	[]	xx	2.0	1	9.890	Barbie comes home from shopping. She takes her...	0	
7837	922902	Vertigo	2016-08-10	['Drama']	xx	2.0	1	10.093	One Person on the street does not vertigo when...	200	

In [47]:
```
xx_info = xx.production_companies.reset_index()
xx_info
```

Out[47]:

	index	production_companies
0	6340	[]
1	7837	[]

i was confused that which country has 'xx' language. there is only two movies and not any complete detail.

In [48]:
```
cn = df.query("original_language == 'cn'")
cn
```

Out[48]:

	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget	prod
89	32516	The Forbidden Legend: Sex & Chopsticks 2	2009-03-04	['Comedy', 'Drama', 'Romance']	cn	6.2	74	255.110	Rich and powerful Simon Qing has been schooled...	0	Stud
138	67308	3-D Sex and Zen: Extreme Ecstasy	2011-04-14	['Drama', 'Adventure', 'Romance', 'Action', 'H...	cn	4.8	86	204.230	Loosely based on a 17th century erotic Chinese...	3500000	['On Lim
431	60898	Erotic Ghost Story	1990-05-19	['Fantasy', 'Drama', 'Horror']	cn	6.0	28	85.259	Three vixens have meditated for 1,000 years to...	0	[
580	146312	The Spiritual Love	1992-11-26	['Horror', 'Fantasy']	cn	5.5	1	87.475	Pleasing the libidos leads to intimate encount...	0	['Lu C
804	410119	Yu Pui Tsuen II	1987-10-15	['Drama', 'Fantasy', 'Romance']	cn	6.2	3	56.151	Before Michael Mak's Sex And Zen became a cult...	0	
...	...	...	...	...	...	...	...	...	...	...	

	id	title	release_date	genres	original_language	vote_average	vote_count	popularity	overview	budget	prod
9595	18665	High Risk	1995-07-12	['Action', 'Comedy']	cn	6.7	107	8.314	After failing to save his wife from 'The Docto...	0	
9688	32629	A Chinese Torture Chamber Story	1994-05-19	['Horror', 'Drama', 'Adventure', 'Comedy']	cn	5.5	31	10.731	A corrupt magistrate subjects a innocent young...	0	
9736	531380	Golden Job	2018-09-20	['Action', 'Adventure', 'Crime']	cn	6.5	80	10.455	A group of former mercenaries reunite to plan ...	0	F
9795	172752	Cash on Delivery	1992-04-30	['Drama', 'Comedy']	cn	3.5	5	9.688	tells the story of a rookie gigolo (Michael Ch...	0	Pr
9853	105001	Iceman	2014-04-17	['Adventure', 'Action', 'Science Fiction', 'Co...	cn	5.5	172	8.537	In the Ming Dynasty, there lives four orphans,...	25477000	

134 rows × 13 columns

Seems like 'cn' is another abreviation for a language from China, Hong Kong or for movies from both

Final thoughts

I thought that movies in english would dominate most of the analysis, because they probably are mostly from USA, but I had to consider as well that India and China are huge markets, and the one and only movie in Telugu proves that point, because it is from India and dominated lots of visualizations, specially those with based on averages.