In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
ds = pd.read_csv('googleplaystore.csv',sep=',')
ds.head()
```

Out[1]:

| | App | Category | Rating | Reviews | Size | Installs | Type | Price | Content Rating | Genres | Last Updated | Current Ver | Android Ver |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Photo Editor & Candy Camera & Grid & ScrapBook | ART_AND_DESIGN | 4.1 | 159 | 19M | 10,000+ | Free | 0 | Everyone | Art & Design | January 7, 2018 | 1.0.0 | 4.0.3 and up |
| **1** | Coloring book moana | ART_AND_DESIGN | 3.9 | 967 | 14M | 500,000+ | Free | 0 | Everyone | Art & Design;Pretend Play | January 15, 2018 | 2.0.0 | 4.0.3 and up |
| **2** | U Launcher Lite – FREE Live Cool Themes, Hide ... | ART_AND_DESIGN | 4.7 | 87510 | 8.7M | 5,000,000+ | Free | 0 | Everyone | Art & Design | August 1, 2018 | 1.2.4 | 4.0.3 and up |
| **3** | Sketch - Draw & Paint | ART_AND_DESIGN | 4.5 | 215644 | 25M | 50,000,000+ | Free | 0 | Teen | Art & Design | June 8, 2018 | Varies with device | 4.2 and up |
| **4** | Pixel Draw - Number Art Coloring Book | ART_AND_DESIGN | 4.3 | 967 | 2.8M | 100,000+ | Free | 0 | Everyone | Art & Design;Creativity | June 20, 2018 | 1.1 | 4.4 and up |

In [2]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10841 entries, 0 to 10840
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   App             10841 non-null  object
 1   Category        10841 non-null  object
 2   Rating          9367 non-null   float64
 3   Reviews         10841 non-null  object
 4   Size            10841 non-null  object
 5   Installs        10841 non-null  object
 6   Type            10840 non-null  object
 7   Price           10841 non-null  object
 8   Content Rating  10840 non-null  object
 9   Genres          10841 non-null  object
 10  Last Updated    10841 non-null  object
 11  Current Ver     10833 non-null  object
 12  Android Ver     10838 non-null  object
dtypes: float64(1), object(12)
memory usage: 1.1+ MB
```

In [3]:
```
ds.isna().sum()
```

Out[3]:
```
App                  0
Category             0
Rating            1474
Reviews              0
Size                 0
Installs             0
Type                 1
Price                0
Content Rating       1
Genres               0
Last Updated         0
Current Ver          8
Android Ver          3
dtype: int64
```

In [4]:
```python
from sklearn.impute import SimpleImputer
ratings = ds[['Rating']]
imp_mean = SimpleImputer(missing_values=np.nan, strategy='median')
imp_mean.fit(ratings)
ratings = imp_mean.transform(ratings)
print(np.unique(ratings))
ds['Rating'] = list(map(lambda x : 5 if x>5 else x[0], ratings))
ratings = ds['Rating']
print(np.unique(ratings))
```

```
[ 1.   1.2  1.4  1.5  1.6  1.7  1.8  1.9  2.   2.1  2.2  2.3  2.4  2.5
  2.6  2.7  2.8  2.9  3.   3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
  4.   4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.  19. ]
[1.  1.2 1.4 1.5 1.6 1.7 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
 3.  3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7
 4.8 4.9 5. ]
```

In [5]:
```python
ds = ds.dropna()
ds.describe().T
```

Out[5]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Rating** | 10829.0 | 4.20651 | 0.480467 | 1.0 | 4.1 | 4.3 | 4.5 | 5.0 |

In [6]:
```python
unique_installs = np.unique(ds['Installs'])
unique_installs
```

Out[6]:
```
array(['0+', '1+', '1,000+', '1,000,000+', '1,000,000,000+', '10+',
       '10,000+', '10,000,000+', '100+', '100,000+', '100,000,000+', '5+',
       '5,000+', '5,000,000+', '50+', '50,000+', '50,000,000+', '500+',
       '500,000+', '500,000,000+'], dtype=object)
```

In [7]:
```python
def installs_to_int(install):
     if install == "0":
        return 0
     else:
        return int(float(''.join(install[:-1].split(','))))
ds["Installs"] = list(map(installs_to_int,ds["Installs"]))
np.unique(ds['Installs'])
```

Out[7]:
```
array([         0,          1,          5,         10,         50,
              100,        500,       1000,       5000,      10000,
            50000,     100000,     500000,    1000000,    5000000,
         10000000,   50000000,  100000000,  500000000, 1000000000],
      dtype=int64)
```

In [8]:
```python
ut = np.unique(ds['Type'])
ut
```

Out[8]:
```
array(['Free', 'Paid'], dtype=object)
```

In [11]:
```python
ds['Reviews'] = ds['Reviews'].apply(int)
uav = np.unique(ds['Android Ver'])
print(uav)
```

```
['1.0 and up' '1.5 and up' '1.6 and up' '2.0 and up' '2.0.1 and up'
 '2.1 and up' '2.2 - 7.1.1' '2.2 and up' '2.3 and up' '2.3.3 and up'
 '3.0 and up' '3.1 and up' '3.2 and up' '4.0 and up' '4.0.3 - 7.1.1'
 '4.0.3 and up' '4.1 - 7.1.1' '4.1 and up' '4.2 and up' '4.3 and up'
 '4.4 and up' '4.4W and up' '5.0 - 6.0' '5.0 - 7.1.1' '5.0 - 8.0'
 '5.0 and up' '5.1 and up' '6.0 and up' '7.0 - 7.1.1' '7.0 and up'
 '7.1 and up' '8.0 and up' 'Varies with device']
```

In [12]:
```python
ds = ds.drop(ds[ds['Android Ver'] == 'Varies with device'].index)
uav = np.unique(ds['Android Ver'])
print(uav)
```

```
['1.0 and up' '1.5 and up' '1.6 and up' '2.0 and up' '2.0.1 and up'
 '2.1 and up' '2.2 - 7.1.1' '2.2 and up' '2.3 and up' '2.3.3 and up'
 '3.0 and up' '3.1 and up' '3.2 and up' '4.0 and up' '4.0.3 - 7.1.1'
 '4.0.3 and up' '4.1 - 7.1.1' '4.1 and up' '4.2 and up' '4.3 and up'
 '4.4 and up' '4.4W and up' '5.0 - 6.0' '5.0 - 7.1.1' '5.0 - 8.0'
 '5.0 and up' '5.1 and up' '6.0 and up' '7.0 - 7.1.1' '7.0 and up'
 '7.1 and up' '8.0 and up']
```

In [13]:
```python
ds['Type'] = list(map(lambda x: True if x == "Free" else
False,ds['Type']))
np.unique(ds['Type'])
ds.rename(columns={'Type': 'IsFree'}, inplace=True)
ds['IsFree']
```

Out[13]:
```
0        True
1        True
2        True
3        True
4        True
         ...
10834    True
10835    True
10836    True
10837    True
10838    True
Name: IsFree, Length: 9468, dtype: bool
```

In [14]:
```python
up = np.unique(ds['Price'])
up
```

Out[14]:
```
array(['$0.99', '$1.00', '$1.04', '$1.20', '$1.26', '$1.29', '$1.49',
       '$1.50', '$1.59', '$1.61', '$1.70', '$1.75', '$1.76', '$1.96',
       '$1.97', '$1.99', '$10.00', '$10.99', '$109.99', '$11.99',
       '$12.99', '$13.99', '$14.00', '$14.99', '$15.46', '$15.99',
       '$154.99', '$16.99', '$17.99', '$18.99', '$19.40', '$19.90',
       '$19.99', '$2.00', '$2.49', '$2.56', '$2.59', '$2.60', '$2.90',
       '$2.99', '$200.00', '$24.99', '$25.99', '$28.99', '$29.99',
       '$299.99', '$3.02', '$3.04', '$3.08', '$3.28', '$3.49', '$3.61',
       '$3.88', '$3.99', '$30.99', '$33.99', '$37.99', '$379.99',
       '$389.99', '$39.99', '$394.99', '$399.99', '$4.29', '$4.49',
       '$4.59', '$4.60', '$4.77', '$4.80', '$4.84', '$4.85', '$4.99',
       '$400.00', '$46.99', '$5.00', '$5.49', '$5.99', '$6.49', '$6.99',
       '$7.49', '$7.99', '$74.99', '$79.99', '$8.49', '$8.99', '$89.99',
       '$9.00', '$9.99', '0'], dtype=object)
```

In [15]:
```python
def pf(price):
    if(price != '0'):
        result = float(''.join(price[1:]))
        return result
    else:
        return 0.0
ds['Price'] = list(map(pf,ds['Price']))
up = np.unique(ds['Price'])
up
```

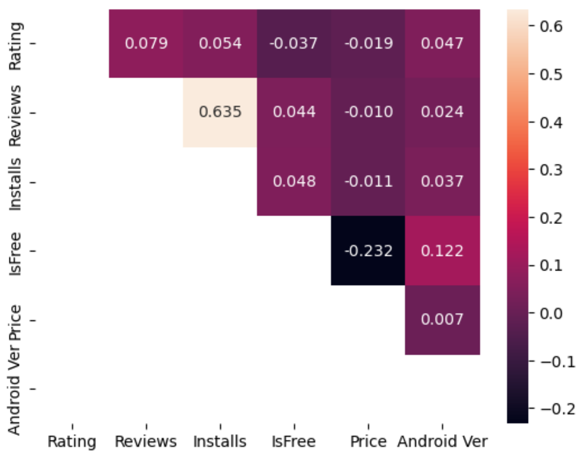
Out[15]:
```
array([  0.  ,   0.99,   1.  ,   1.04,   1.2 ,   1.26,   1.29,   1.49,
         1.5 ,   1.59,   1.61,   1.7 ,   1.75,   1.76,   1.96,   1.97,
         1.99,   2.  ,   2.49,   2.56,   2.59,   2.6 ,   2.9 ,   2.99,
         3.02,   3.04,   3.08,   3.28,   3.49,   3.61,   3.88,   3.99,
         4.29,   4.49,   4.59,   4.6 ,   4.77,   4.8 ,   4.84,   4.85,
         4.99,   5.  ,   5.49,   5.99,   6.49,   6.99,   7.49,   7.99,
         8.49,   8.99,   9.  ,   9.99,  10.  ,  10.99,  11.99,  12.99,
        13.99,  14.  ,  14.99,  15.46,  15.99,  16.99,  17.99,  18.99,
        19.4 ,  19.9 ,  19.99,  24.99,  25.99,  28.99,  29.99,  30.99,
        33.99,  37.99,  39.99,  46.99,  74.99,  79.99,  89.99, 109.99,
       154.99, 200.  , 299.99, 379.99, 389.99, 394.99, 399.99, 400.  ])
```

In [16]:
```python
def version(anver):
    if '-' in anver:
        return float(anver.split(' ')[-1][:3])
    else:
        return float(anver.split(' ')[0][:3])
ds['Android Ver'] = list(map(version,ds['Android Ver']))
uav = np.unique(ds['Android Ver'])
print(uav)
```

```
[1.  1.5 1.6 2.  2.1 2.2 2.3 3.  3.1 3.2 4.  4.1 4.2 4.3 4.4 5.  5.1 6.
 7.  7.1 8. ]
```

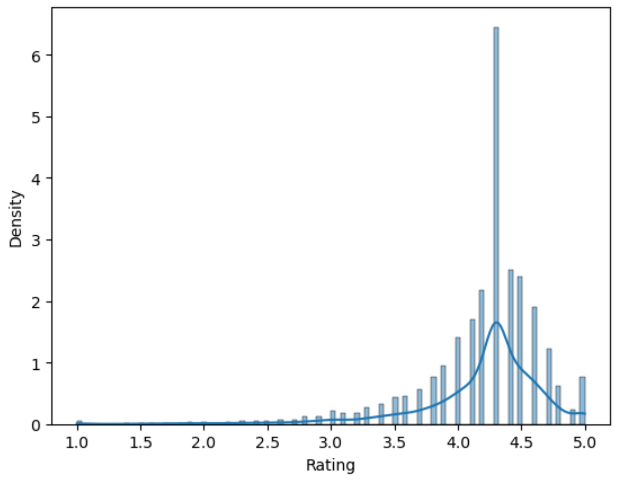
In [18]:
```python
mask = np.zeros_like(ds.corr(), dtype=bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(ds.corr(), mask=mask, annot=True,fmt='.3f')
```

```
Out[18]: <AxesSubplot:>
```



```
In [19]: sns.histplot(ds['Rating'], kde=True, stat='density')
Out[19]: <AxesSubplot:xlabel='Rating', ylabel='Density'>
```

```
In [19]: sns.histplot(ds['Rating'], kde=True, stat='density')
Out[19]: <AxesSubplot:xlabel='Rating', ylabel='Density'>
```



Как видно из таблицы корреляции, есть высокая корреляция между количеством отзывов и количеством скачиваний.