In [2]:
```
#import sys
!{sys.executable} -m pip install PyAthena
```

```
Requirement already satisfied: PyAthena in /Users/thamilventhananthonysmariathas/anaconda3/lib/pytho
n3.6/site-packages (2.3.0)
Requirement already satisfied: botocore>=1.5.52 in /Users/thamilventhananthonysmariathas/anaconda3/l
ib/python3.6/site-packages (from PyAthena) (1.21.56)
Requirement already satisfied: boto3>=1.4.4 in /Users/thamilventhananthonysmariathas/anaconda3/lib/p
ython3.6/site-packages (from PyAthena) (1.10.37)
Requirement already satisfied: tenacity>=4.1.0 in /Users/thamilventhananthonysmariathas/anaconda3/li
b/python3.6/site-packages (from PyAthena) (8.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /Users/thamilventhananthonysmariathas/
anaconda3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (2.8.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /Users/thamilventhananthonysmariathas/anaco
nda3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (0.9.4)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /Users/thamilventhananthonysmariathas/anacon
da3/lib/python3.6/site-packages (from botocore>=1.5.52->PyAthena) (1.25.6)
Requirement already satisfied: s3transfer<0.3.0,>=0.2.0 in /Users/thamilventhananthonysmariathas/ana
conda3/lib/python3.6/site-packages (from boto3>=1.4.4->PyAthena) (0.2.1)
Requirement already satisfied: six>=1.5 in /Users/thamilventhananthonysmariathas/anaconda3/lib/pytho
n3.6/site-packages (from python-dateutil<3.0.0,>=2.1->botocore>=1.5.52->PyAthena) (1.12.0)
```

In [2]:
```
!aws configure get region
```

```
us-west-2
```

In [3]:
```
from pyathena import connect

import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline
%config InlineBackend.figure_format='retina'
```

In [4]:
```
# Set Athena database & table
database_name = "amazonreviewsdb"
table_name = "amazon_reviews_parquet"
```

In [5]:
```
# Set S3 staging directory -- this is a temporary directory used for Athena queries
s3_staging_dir = "s3://athena-query-2060/queryresults/"
```

In [10]:
```
conn = connect(region_name='us-west-2', s3_staging_dir=s3_staging_dir)
```

Set Seaborn Parameters

In [11]:
```
sns.set_style = "seaborn-whitegrid"

sns.set(
    rc={
        "font.style": "normal",
        "axes.facecolor": "white",
        "grid.color": ".8",
        "grid.linestyle": "-",
        "figure.facecolor": "white",
        "figure.titlesize": 20,
        "text.color": "black",
        "xtick.color": "black",
        "ytick.color": "black",
        "axes.labelcolor": "black",
        "axes.grid": True,
        "axes.labelsize": 10,
        "xtick.labelsize": 10,
        "font.size": 10,
        "ytick.labelsize": 10,
    }
)
```

Helper Code to Display Values on Bars

In [12]:
```
def show_values_barplot(axs, space):
    def _show_on_plot(ax):
        for p in ax.patches:
            _x = p.get_x() + p.get_width() + float(space)
            _y = p.get_y() + p.get_height()
            value = round(float(p.get_width()), 2)
            ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _show_on_plot(ax)
    else:
        _show_on_plot(axs)
```

1. Which Product Categories are Highest Rated by Average Rating?

In [13]:
```
# SQL statement
statement = """
SELECT product_category, AVG(star_rating) AS avg_star_rating
FROM {}.{}
GROUP BY product_category
ORDER BY avg_star_rating DESC
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_category, AVG(star_rating) AS avg_star_rating
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY product_category
ORDER BY avg_star_rating DESC
```

In [14]:
```
df = pd.read_sql(statement, conn)
df.head(5)
```

Out[14]:

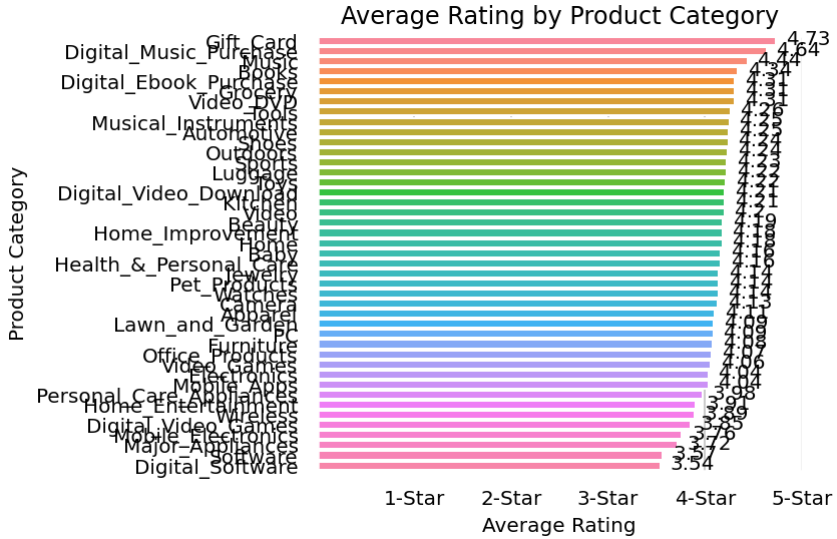
	product_category	avg_star_rating
0	Gift_Card	4.731363
1	Digital_Music_Purchase	4.636946
2	Music	4.440541
3	Books	4.340540
4	Digital_Ebook_Purchase	4.312491

In [15]:
```
# Store number of categories
num_categories = df.shape[0]
print(num_categories)

# Store average star ratings
average_star_ratings = df
```

```
43
```

Visualization for a Subset of Product Categories

In [18]:
```
# Create plot
barplot = sns.barplot(y="product_category", x="avg_star_rating", data=df, saturation=1)

if num_categories < 10:
    sns.set(rc={"figure.figsize": (50.0, 20.0)})

# Set title and x-axis ticks
plt.title("Average Rating by Product Category")
plt.xticks([1, 2, 3, 4, 5], ["1-Star", "2-Star", "3-Star", "4-Star", "5-Star"])

# Helper code to show actual values afters bars
show_values_barplot(barplot, 0.1)

plt.xlabel("Average Rating")
plt.ylabel("Product Category")

# Export plot if needed
plt.tight_layout()
# plt.savefig('avg_ratings_per_category.png', dpi=300)

# Show graphic
plt.show(barplot)
```



2. Which Product Categories Have the Most Reviews?

In [19]:
```
# SQL statement
statement = """
SELECT product_category, COUNT(star_rating) AS count_star_rating
FROM {}.{}
GROUP BY product_category
ORDER BY count_star_rating DESC
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_category, COUNT(star_rating) AS count_star_rating
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY product_category
ORDER BY count_star_rating DESC
```

In [20]:
```
df = pd.read_sql(statement, conn)
df.head()
```

Out[20]:

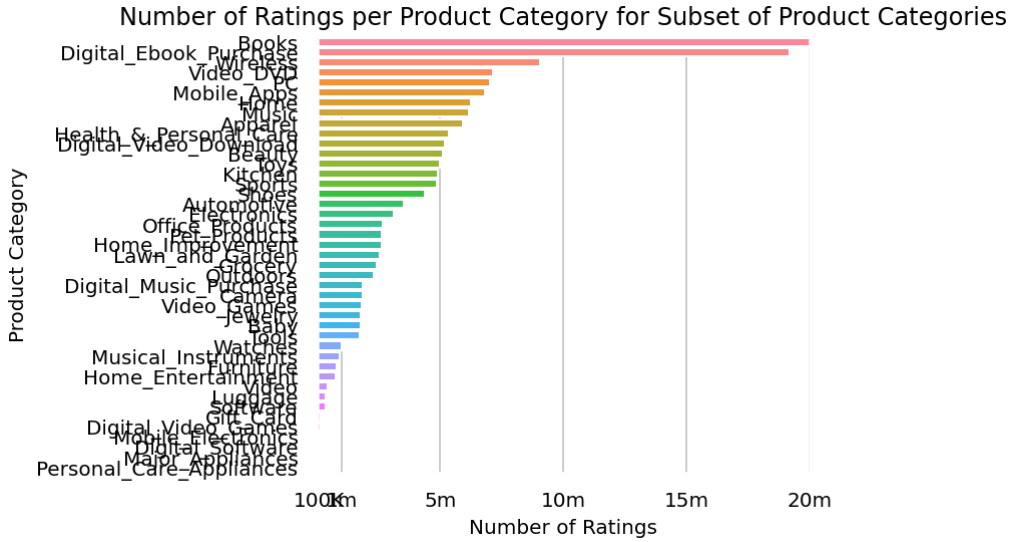
	product_category	count_star_rating
0	Books	20726160
1	Digital_Ebook_Purchase	19180765
2	Wireless	9038249
3	Video_DVD	7135819
4	PC	7004337

In [21]:
```
# Store counts
count_ratings = df["count_star_rating"]

# Store max ratings
max_ratings = df["count_star_rating"].max()
print(max_ratings)
```

```
20726160
```

Visualization for a Subset of Product Categories

In [22]:
```
# Create Seaborn barplot
barplot = sns.barplot(y="product_category", x="count_star_rating", data=df, saturation=1)

if num_categories < 10:
    sns.set(rc={"figure.figsize": (10.0, 5.0)})

# Set title
plt.title("Number of Ratings per Product Category for Subset of Product Categories")

# Set x-axis ticks to match scale
if max_ratings > 200000:
    plt.xticks([100000, 1000000, 5000000, 10000000, 15000000, 20000000], ["100K", "1m", "5m", "10m",
"15m", "20m"])
    plt.xlim(0, 20000000)
elif max_ratings <= 200000:
    plt.xticks([50000, 100000, 150000, 200000], ["50K", "100K", "150K", "200K"])
    plt.xlim(0, 200000)

plt.xlabel("Number of Ratings")
plt.ylabel("Product Category")

plt.tight_layout()

# Export plot if needed
# plt.savefig('ratings_per_category.png', dpi=300)

# Show the barplot
plt.show(barplot)
```



3. When did each product category become available in the Amazon catalog based on the date of the first review?

In [23]:
```
# SQL statement
statement = """
SELECT product_category, MIN(review_date) AS first_review_date
FROM {}.{}
GROUP BY product_category
ORDER BY first_review_date
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_category, MIN(review_date) AS first_review_date
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY product_category
ORDER BY first_review_date
```

In [24]:
```
df = pd.read_sql(statement, conn)
df.head()
```

Out[24]:

	product_category	first_review_date
0	Music	1322
1	Books	9305
2	Video	9445
3	Video_DVD	9685
4	Toys	9866

In [25]:
```
# Convert date strings (e.g. 2014-10-18) to datetime
import datetime as datetime

dates = pd.to_datetime(df["first_review_date"])
```

In [26]:
```
# See: https://stackoverflow.com/questions/60761410/how-to-graph-events-on-a-timeline


def modify_dataframe(df):
    """ Modify dataframe to include new columns """
    df["year"] = pd.to_datetime(df["first_review_date"], format="%Y-%m-%d").dt.year
    return df


def get_x_y(df):
    """ Get X and Y coordinates; return tuple """
    series = df["year"].value_counts().sort_index()
    # new_series = series.reindex(range(1,21)).fillna(0).astype(int)
    return series.index, series.values
```

In [27]:
```
new_df = modify_dataframe(df)
print(new_df)

X, Y = get_x_y(new_df)
```

	product_category	first_review_date	year
0	Music	1322	1970
1	Books	9305	1970
2	Video	9445	1970
3	Video_DVD	9685	1970
4	Toys	9866	1970
5	Sports	10143	1970
6	Video_Games	10171	1970
7	Home	10375	1970
8	Office_Products	10422	1970
9	Pet_Products	10461	1970
10	Software	10490	1970
11	Home_Entertainment	10514	1970
12	Camera	10550	1970
13	Wireless	10564	1970
14	Health_&_Personal_Care	10628	1970
15	Outdoors	10674	1970
16	Electronics	10751	1970
17	PC	10773	1970
18	Baby	10785	1970
19	Digital_Ebook_Purchase	10831	1970
20	Grocery	10839	1970
21	Automotive	10888	1970
22	Shoes	10903	1970
23	Home_Improvement	10903	1970
24	Tools	10904	1970
25	Lawn_and_Garden	10922	1970
26	Musical_Instruments	10938	1970
27	Kitchen	10976	1970
28	Furniture	11033	1970
29	Digital_Music_Purchase	11136	1970
30	Major_Appliances	11195	1970
31	Apparel	11206	1970
32	Digital_Video_Download	11234	1970
33	Personal_Care_Appliances	11259	1970
34	Beauty	11261	1970
35	Watches	11417	1970
36	Jewelry	11636	1970
37	Mobile_Electronics	11678	1970
38	Luggage	11996	1970
39	Gift_Card	12705	1970
40	Digital_Video_Games	13368	1970

```
41          Digital_Software             13904  1970
42               Mobile_Apps             14917  1970
```

4. What is the breakdown of ratings (1-5) per product category?

In [29]:
```
# SQL statement
statement = """
SELECT product_category,
         star_rating,
         COUNT(*) AS count_reviews
FROM {}.{}
GROUP BY  product_category, star_rating
ORDER BY  product_category ASC, star_rating DESC, count_reviews
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_category,
         star_rating,
         COUNT(*) AS count_reviews
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY  product_category, star_rating
ORDER BY  product_category ASC, star_rating DESC, count_reviews
```

In [30]:
```
df = pd.read_sql(statement, conn)
df
```

Out[30]:

	product_category	star_rating	count_reviews
0	Apparel	5	3320651
1	Apparel	4	1147254
2	Apparel	3	623483
3	Apparel	2	369608
4	Apparel	1	445464
...	...	...	...
210	Wireless	5	4845345
211	Wireless	4	1507668
212	Wireless	3	818151
213	Wireless	2	600436
214	Wireless	1	1266649

215 rows × 3 columns

Prepare for Stacked Percentage Horizontal Bar Plot Showing Proportion of Star Ratings per Product Category

In [31]:
```
# Create grouped DataFrames by category and by star rating
grouped_category = df.groupby("product_category")
grouped_star = df.groupby("star_rating")

# Create sum of ratings per star rating
df_sum = df.groupby(["star_rating"]).sum()

# Calculate total number of star ratings
total = df_sum["count_reviews"].sum()
print(total)
```

```
160796570
```

In [32]:
```
# Create dictionary of product categories and array of star rating distribution per category
distribution = {}
count_reviews_per_star = []
i = 0

for category, ratings in grouped_category:
    count_reviews_per_star = []
    for star in ratings["star_rating"]:
        count_reviews_per_star.append(ratings.at[i, "count_reviews"])
        i = i + 1
    distribution[category] = count_reviews_per_star

# Check if distribution has been created succesfully
print(distribution)
```

```
{'Apparel': [3320651, 1147254, 623483, 369608, 445464], 'Automotive': [2301688, 526898, 240023, 1478
43, 300024], 'Baby': [1078545, 289129, 150753, 101427, 145039], 'Beauty': [3254946, 741443, 398405,
264029, 456898], 'Books': [13662131, 3546319, 1543611, 861867, 1112232], 'Camera': [1085596, 344163,
144596, 92396, 172009], 'Digital_Ebook_Purchase': [11612150, 4311745, 1701301, 749678, 805891], 'Dig
ital_Music_Purchase': [1467877, 231676, 70507, 28743, 53381], 'Digital_Software': [46410, 16693, 830
8, 6890, 23783], 'Digital_Video_Download': [3122980, 984221, 449260, 251930, 365352], 'Digital_Video
_Games': [80677, 20406, 11629, 7749, 24970], 'Electronics': [1796672, 542181, 240859, 180668, 36055
8], 'Furniture': [447763, 153678, 73574, 43853, 73346], 'Gift_Card': [129709, 9859, 3156, 1569, 479
3], 'Grocery': [1662278, 293389, 161497, 105265, 180049], 'Health_&_Personal_Care': [3359016, 78202
5, 400572, 278066, 513204], 'Home': [3897623, 960326, 500884, 324345, 545389], 'Home_Entertainment':
[395254, 138172, 60898, 46242, 103134], 'Home_Improvement': [1660254, 419488, 189490, 124792, 24663
0], 'Jewelry': [1081479, 270579, 159735, 100852, 155108], 'Kitchen': [3129752, 732976, 350188, 24236
1, 427554], 'Lawn_and_Garden': [1546990, 401137, 194777, 135055, 281302], 'Luggage': [216857, 61488,
27872, 17887, 25028], 'Major_Appliances': [49704, 15158, 6718, 5475, 19846], 'Mobile_Apps': [374738
3, 1375672, 614333, 314750, 755028], 'Mobile_Electronics': [52373, 18112, 9754, 7320, 17600], 'Musi
c': [4296015, 1018968, 405541, 202861, 254396], 'Musical_Instruments': [582772, 161623, 68231, 4090
6, 67197], 'Office_Products': [1586632, 419489, 194113, 138717, 307540], 'Outdoors': [1436837, 41810
3, 179395, 109599, 161662], 'PC': [4166576, 1185891, 520947, 366603, 764320], 'Personal_Care_Applian
ces': [49310, 13782, 7107, 5381, 11106], 'Pet_Products': [1645587, 381294, 216639, 151284, 248866],
'Shoes': [2647584, 850452, 405263, 243424, 232752], 'Software': [154057, 58696, 30693, 24672, 7401
7], 'Sports': [3046077, 838841, 381388, 229890, 363858], 'Tools': [1118325, 300861, 126637, 75055, 1
27732], 'Toys': [3160447, 788495, 395143, 233754, 403762], 'Video': [263744, 81794, 39030, 21697, 31
144], 'Video_DVD': [4702079, 1138174, 540591, 296845, 458130], 'Video_Games': [1040615, 321861, 1554
96, 95999, 194515], 'Watches': [582164, 176290, 80381, 52864, 86343], 'Wireless': [4845345, 1507668,
818151, 600436, 1266649]}
```

In [33]:
```
# Check if distribution keys are set correctly to product categories
print(distribution.keys())
```

```
dict_keys(['Apparel', 'Automotive', 'Baby', 'Beauty', 'Books', 'Camera', 'Digital_Ebook_Purchase',
'Digital_Music_Purchase', 'Digital_Software', 'Digital_Video_Download', 'Digital_Video_Games', 'Elec
tronics', 'Furniture', 'Gift_Card', 'Grocery', 'Health_&_Personal_Care', 'Home', 'Home_Entertainmen
t', 'Home_Improvement', 'Jewelry', 'Kitchen', 'Lawn_and_Garden', 'Luggage', 'Major_Appliances', 'Mob
ile_Apps', 'Mobile_Electronics', 'Music', 'Musical_Instruments', 'Office_Products', 'Outdoors', 'P
C', 'Personal_Care_Appliances', 'Pet_Products', 'Shoes', 'Software', 'Sports', 'Tools', 'Toys', 'Vid
eo', 'Video_DVD', 'Video_Games', 'Watches', 'Wireless'])
```

In [34]:
```
# Check if star rating distributions are set correctly
print(distribution.items())
```

```
dict_items([('Apparel', [3320651, 1147254, 623483, 369608, 445464]), ('Automotive', [2301688, 52689
8, 240023, 147843, 300024]), ('Baby', [1078545, 289129, 150753, 101427, 145039]), ('Beauty', [325494
6, 741443, 398405, 264029, 456898]), ('Books', [13662131, 3546319, 1543611, 861867, 1112232]), ('Cam
era', [1085596, 344163, 144596, 92396, 172009]), ('Digital_Ebook_Purchase', [11612150, 4311745, 1701
301, 749678, 805891]), ('Digital_Music_Purchase', [1467877, 231676, 70507, 28743, 53381]), ('Digital
_Software', [46410, 16693, 8308, 6890, 23783]), ('Digital_Video_Download', [3122980, 984221, 449260,
251930, 365352]), ('Digital_Video_Games', [80677, 20406, 11629, 7749, 24970]), ('Electronics', [1796
672, 542181, 240859, 180668, 360558]), ('Furniture', [447763, 153678, 73574, 43853, 73346]), ('Gift_
Card', [129709, 9859, 3156, 1569, 4793]), ('Grocery', [1662278, 293389, 161497, 105265, 180049]),
('Health_&_Personal_Care', [3359016, 782025, 400572, 278066, 513204]), ('Home', [3897623, 960326, 50
0884, 324345, 545389]), ('Home_Entertainment', [395254, 138172, 60898, 46242, 103134]), ('Home_Impro
vement', [1660254, 419488, 189490, 124792, 246630]), ('Jewelry', [1081479, 270579, 159735, 100852, 1
55108]), ('Kitchen', [3129752, 732976, 350188, 242361, 427554]), ('Lawn_and_Garden', [1546990, 40113
7, 194777, 135055, 281302]), ('Luggage', [216857, 61488, 27872, 17887, 25028]), ('Major_Appliances',
[49704, 15158, 6718, 5475, 19846]), ('Mobile_Apps', [3747383, 1375672, 614333, 314750, 755028]), ('M
obile_Electronics', [52373, 18112, 9754, 7320, 17600]), ('Music', [4296015, 1018968, 405541, 202861,
254396]), ('Musical_Instruments', [582772, 161623, 68231, 40906, 67197]), ('Office_Products', [15866
32, 419489, 194113, 138717, 307540]), ('Outdoors', [1436837, 418103, 179395, 109599, 161662]), ('P
C', [4166576, 1185891, 520947, 366603, 764320]), ('Personal_Care_Appliances', [49310, 13782, 7107, 5
381, 11106]), ('Pet_Products', [1645587, 381294, 216639, 151284, 248866]), ('Shoes', [2647584, 85045
2, 405263, 243424, 232752]), ('Software', [154057, 58696, 30693, 24672, 74017]), ('Sports', [304607
7, 838841, 381388, 229890, 363858]), ('Tools', [1118325, 300861, 126637, 75055, 127732]), ('Toys',
[3160447, 788495, 395143, 233754, 403762]), ('Video', [263744, 81794, 39030, 21697, 31144]), ('Video
_DVD', [4702079, 1138174, 540591, 296845, 458130]), ('Video_Games', [1040615, 321861, 155496, 95999,
194515]), ('Watches', [582164, 176290, 80381, 52864, 86343]), ('Wireless', [4845345, 1507668, 81815
1, 600436, 1266649])])
```

In [35]:
```
# Sort distribution by average rating per category
sorted_distribution = {}

average_star_ratings.iloc[:, 0]
for index, value in average_star_ratings.iloc[:, 0].items():
    sorted_distribution[value] = distribution[value]
```

In [36]:
```
df_sorted_distribution_pct = pd.DataFrame(sorted_distribution).transpose().apply(
    lambda num_ratings: num_ratings/sum(num_ratings)*100, axis=1
)
df_sorted_distribution_pct.columns=['5', '4', '3', '2', '1']
df_sorted_distribution_pct
```

Out[36]:

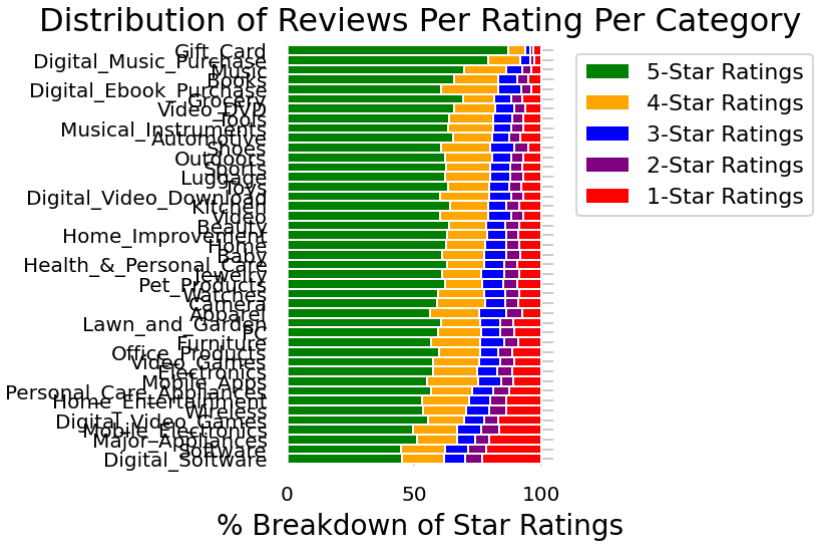
	5	4	3	2	1
Gift_Card	87.002804	6.612962	2.116899	1.052413	3.214923
Digital_Music_Purchase	79.251144	12.508261	3.806695	1.551844	2.882057
Music	69.539775	16.494078	6.564509	3.283720	4.117919
Books	65.917329	17.110352	7.447646	4.158354	5.366320
Digital_Ebook_Purchase	60.540599	22.479526	8.869829	3.908489	4.201558
Grocery	69.190145	12.211933	6.722101	4.381518	7.494304
Video_DVD	65.894034	15.950152	7.575739	4.159929	6.420146
Tools	63.955084	17.205723	7.242152	4.292266	7.304774
Musical_Instruments	63.294628	17.553808	7.410541	4.442784	7.298239
Automotive	65.454392	14.983694	6.825669	4.204294	8.531951
Shoes	60.454370	19.419040	9.253689	5.558292	5.314610
Outdoors	62.319548	18.134270	7.780851	4.753608	7.011723
Sports	62.675785	17.259911	7.847403	4.730194	7.486707
Luggage	62.113184	17.611677	7.983227	5.123277	7.168635
Toys	63.442395	15.828144	7.932048	4.692347	8.105065
Digital_Video_Download	60.362101	19.023384	8.683462	4.869395	7.061657
Kitchen	64.097078	15.011292	7.171823	4.963534	8.756273
Video	60.296885	18.699661	8.922999	4.960346	7.120110
Beauty	63.626339	14.493421	7.787856	5.161130	8.931253
Home_Improvement	62.872834	15.885762	7.175874	4.725799	9.339732
Home	62.576561	15.418089	8.041721	5.207378	8.756252
Baby	61.111070	16.382240	8.541764	5.746921	8.218005
Health_&_Personal_Care	62.986868	14.664207	7.511359	5.214178	9.623388
Jewelry	61.178174	15.306380	9.036047	5.705096	8.774303
Pet_Products	62.246309	14.422905	8.194631	5.722499	9.413656

	5	4	3	2	1
Watches	59.523415	18.024788	8.218563	5.405085	8.828148
Camera	59.039570	18.717125	7.863778	5.024908	9.354619
Apparel	56.220663	19.423716	10.555951	6.257691	7.541979
Lawn_and_Garden	60.446746	15.673939	7.610674	5.277109	10.991532
PC	59.485659	16.930810	7.437492	5.233943	10.912096
Furniture	56.520460	19.398546	9.287137	5.535499	9.258357
Office_Products	59.952292	15.850762	7.334731	5.241544	11.620671
Video_Games	57.540672	17.797262	8.598131	5.308252	10.755682
Electronics	57.568334	17.372373	7.717520	5.788901	11.552873
Mobile_Apps	55.050560	20.209174	9.024798	4.623804	11.091664
Personal_Care_Appliances	56.883464	15.898761	8.198556	6.207461	12.811757
Home_Entertainment	53.146968	18.578997	8.188517	6.217830	13.867689
Wireless	53.609333	16.680974	9.052096	6.643278	14.014318
Digital_Video_Games	55.474417	14.031396	7.996232	5.328300	17.169654
Mobile_Electronics	49.803631	17.223443	9.275478	6.960888	16.736561
Major_Appliances	51.293588	15.642769	6.932849	5.650096	20.480697
Software	45.028132	17.155801	8.971020	7.211189	21.633858
Digital_Software	45.462560	16.352220	8.138396	6.749344	23.297481

Visualization for a Subset of Product Categories

In [40]:
```
categories = df_sorted_distribution_pct.index

# Plot bars
if len(categories) > 10:
    plt.figure(figsize=(10,10))
else:
    plt.figure(figsize=(10,5))

df_sorted_distribution_pct.plot(kind="barh",
                                stacked=True,
                                edgecolor='white',
                                width=1.0,
                                color=['green',
                                       'orange',
                                       'blue',
                                       'purple',
                                       'red'])

plt.title("Distribution of Reviews Per Rating Per Category",
          fontsize='16')

plt.legend(bbox_to_anchor=(1.04,1),
           loc="upper left",
           labels=['5-Star Ratings',
                   '4-Star Ratings',
                   '3-Star Ratings',
                   '2-Star Ratings',
                   '1-Star Ratings'])

plt.xlabel("% Breakdown of Star Ratings", fontsize='14')
plt.gca().invert_yaxis()
plt.tight_layout()

plt.show();
```

```
<Figure size 720x720 with 0 Axes>
```



5. How Many Reviews per Star Rating? (5, 4, 3, 2, 1)

In [44]:
```
# SQL statement
statement = """
SELECT star_rating,
         COUNT(*) AS count_reviews
FROM {}.{}
GROUP BY  star_rating
ORDER BY  star_rating DESC, count_reviews
""".format(
    database_name, table_name
)

print(statement)
```

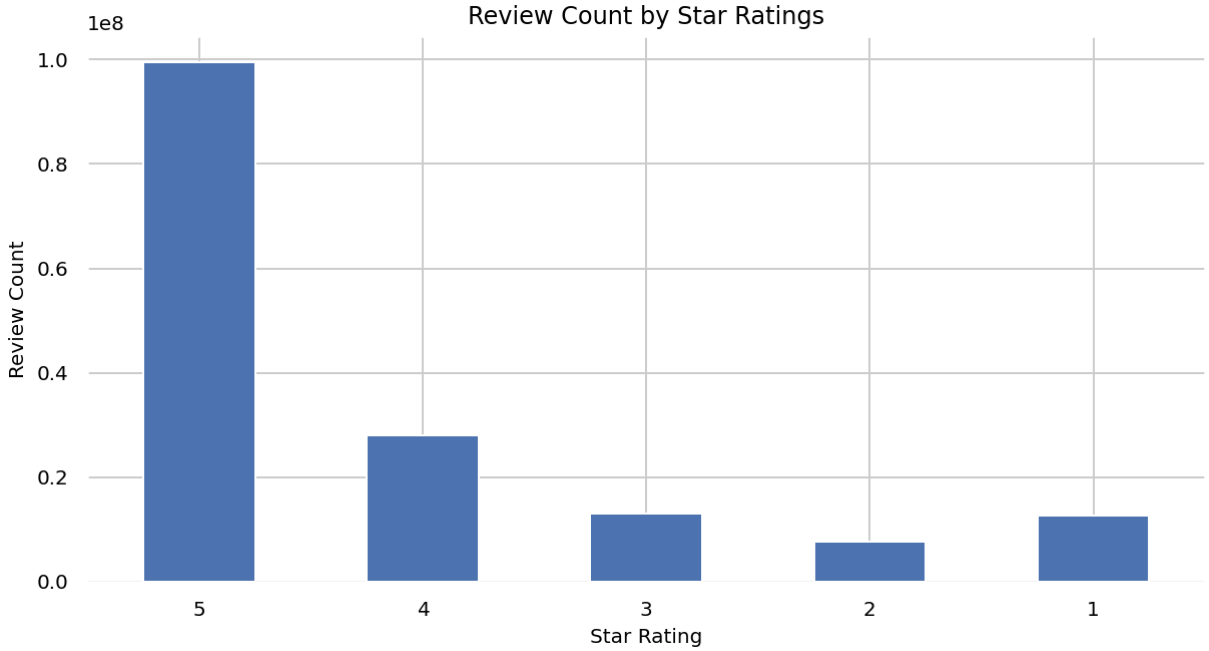
```
SELECT star_rating,
         COUNT(*) AS count_reviews
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY  star_rating
ORDER BY  star_rating DESC, count_reviews
```

In [45]:
```
df = pd.read_sql(statement, conn)
df
```

Out[45]:

	star_rating	count_reviews
0	5	99530924
1	4	27996469
2	3	12900929
3	2	7700647
4	1	12667601

In [47]:
```
chart = df.plot.bar(
    x="star_rating", y="count_reviews", rot="0", figsize=(10, 5), title="Review Count by Star Rating
s", legend=False
)

plt.xlabel("Star Rating")
plt.ylabel("Review Count")

plt.show(chart)
```



6. How Did Star Ratings Change Over Time?

Is there a drop-off point for certain product categories throughout the year?

In [48]:
```
# SQL statement
statement = """
SELECT year, ROUND(AVG(star_rating),4) AS avg_rating
FROM {}.{}
GROUP BY year
ORDER BY year
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT year, ROUND(AVG(star_rating),4) AS avg_rating
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY year
ORDER BY year
```

In [49]:
```
df = pd.read_sql(statement, conn)
df
```

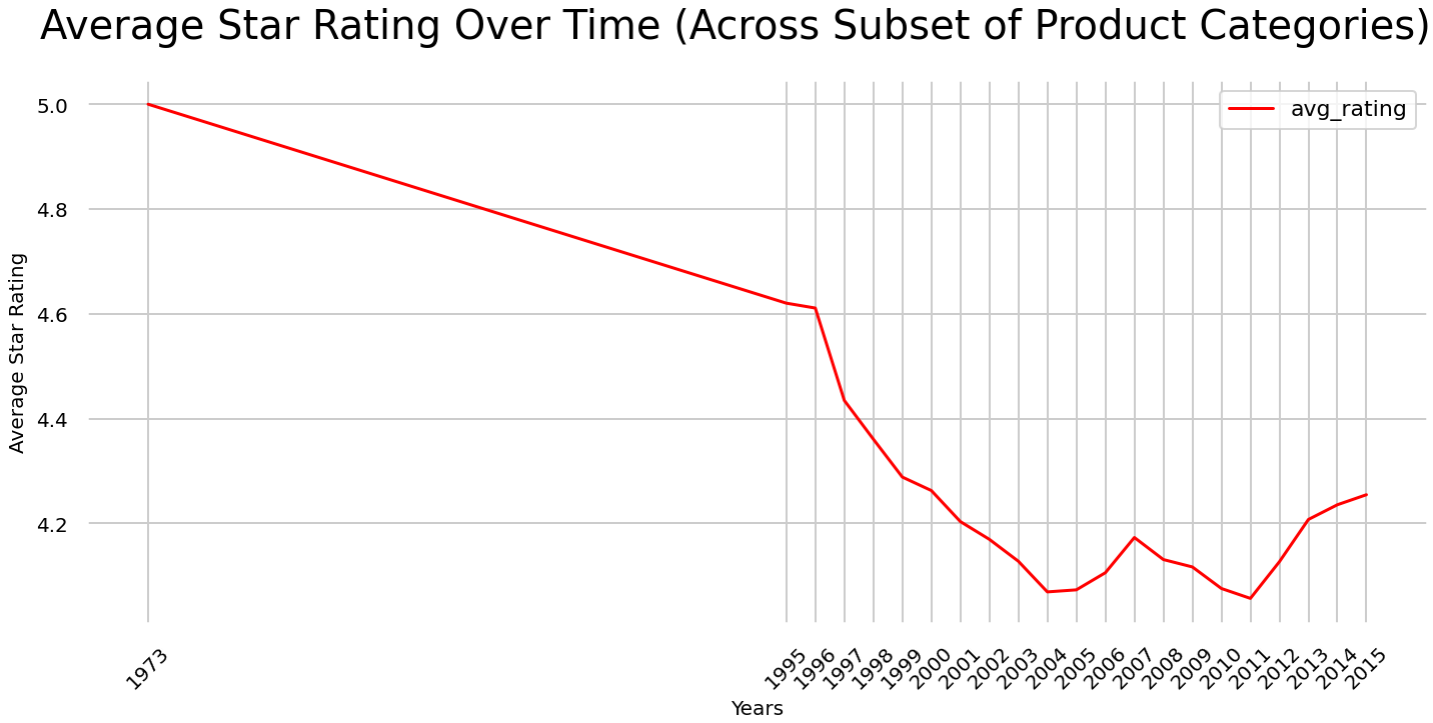
Out[49]:

	year	avg_rating
0	1973	5.0000
1	1995	4.6204
2	1996	4.6111
3	1997	4.4348
4	1998	4.3610
5	1999	4.2886
6	2000	4.2629
7	2001	4.2037
8	2002	4.1698
9	2003	4.1280
10	2004	4.0695
11	2005	4.0734
12	2006	4.1061
13	2007	4.1733
14	2008	4.1312
15	2009	4.1171
16	2010	4.0760
17	2011	4.0571
18	2012	4.1271
19	2013	4.2077
20	2014	4.2358
21	2015	4.2549

In [50]:
```
df["year"] = pd.to_datetime(df["year"], format="%Y").dt.year
```

Visualization for a Subset of Product Categories

In [51]:
```
fig = plt.gcf()
fig.set_size_inches(12, 5)

fig.suptitle("Average Star Rating Over Time (Across Subset of Product Categories)")

ax = plt.gca()
# ax = plt.gca().set_xticks(df['year'])
ax.locator_params(integer=True)
ax.set_xticks(df["year"].unique())

df.plot(kind="line", x="year", y="avg_rating", color="red", ax=ax)

# plt.xticks(range(1995, 2016, 1))
# plt.yticks(range(0,6,1))
plt.xlabel("Years")
plt.ylabel("Average Star Rating")
plt.xticks(rotation=45)

# fig.savefig('average-rating.png', dpi=300)
plt.show()
```



Average Star Rating Per Product Categories Across Time

In [52]:
```
# SQL statement
statement = """
SELECT product_category, year, ROUND(AVG(star_rating), 4) AS avg_rating_category
FROM {}.{}
GROUP BY product_category, year
ORDER BY year
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_category, year, ROUND(AVG(star_rating), 4) AS avg_rating_category
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY product_category, year
ORDER BY year
```

In [53]:
```
df = pd.read_sql(statement, conn)
df
```

Out[53]:

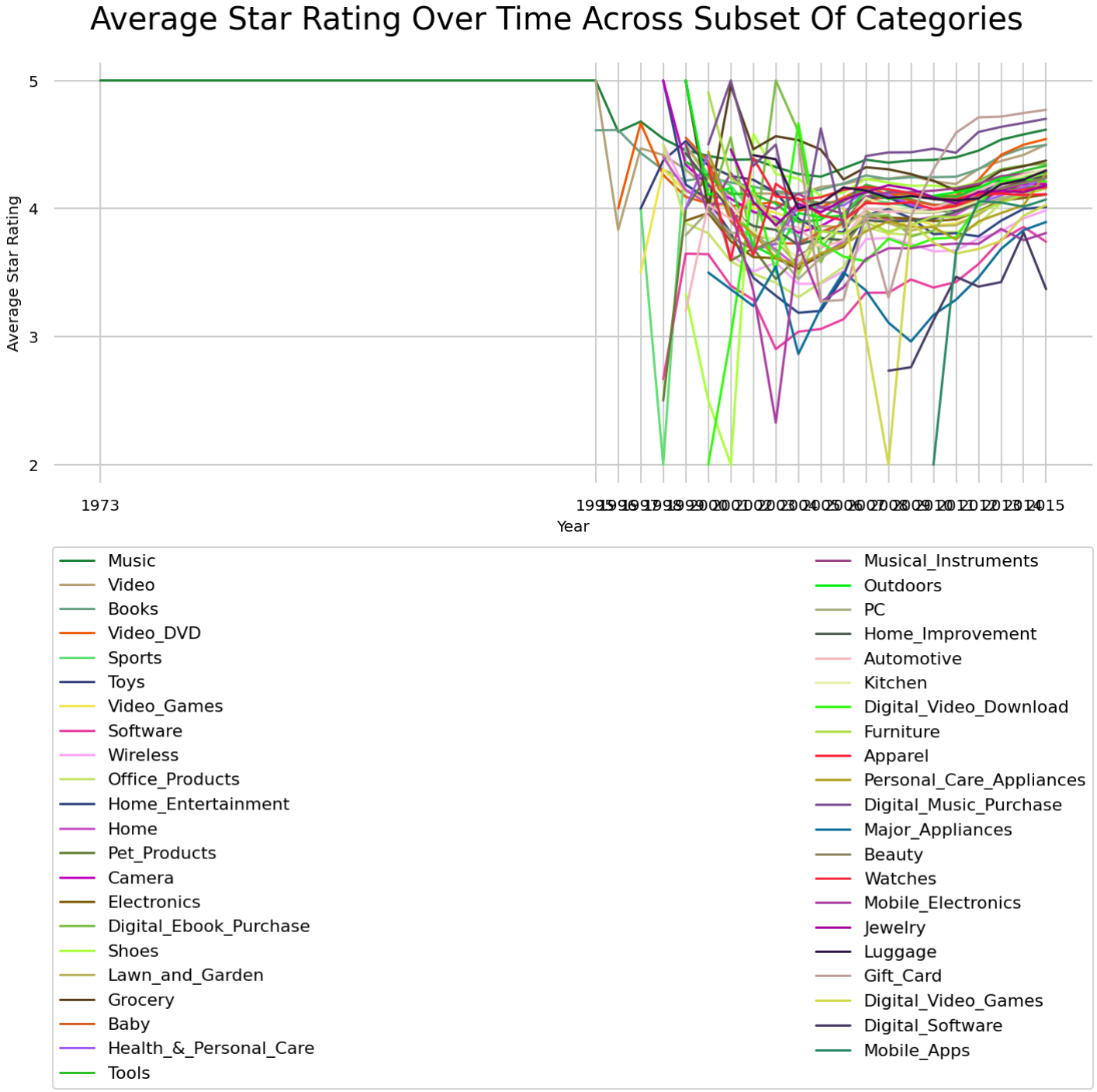
	product_category	year	avg_rating_category
0	Music	1973	5.0000
1	Video	1995	5.0000
2	Books	1995	4.6114
3	Music	1995	5.0000
4	Music	1996	4.6000
...	...	...	...
704	Software	2015	3.7407
705	Jewelry	2015	4.1817
706	Health_&_Personal_Care	2015	4.2382
707	Office_Products	2015	4.2103
708	PC	2015	4.1576

709 rows × 3 columns

Visualization

In [54]:
```
def plot_categories(df):
    df_categories = df["product_category"].unique()
    for category in df_categories:
        # print(category)
        df_plot = df.loc[df["product_category"] == category]
        df_plot.plot(
            kind="line",
            x="year",
            y="avg_rating_category",
            c=np.random.rand(
                3,
            ),
            ax=ax,
            label=category,
        )
```

In [55]:
```
fig = plt.gcf()
fig.set_size_inches(12, 5)

fig.suptitle("Average Star Rating Over Time Across Subset Of Categories")

ax = plt.gca()

ax.locator_params(integer=True)
ax.set_xticks(df["year"].unique())

plot_categories(df)

plt.xlabel("Year")
plt.ylabel("Average Star Rating")
plt.legend(bbox_to_anchor=(0, -0.15, 1, 0), loc=2, ncol=2, mode="expand", borderaxespad=0)

# fig.savefig('average_rating_category_all_data.png', dpi=300)
plt.show()
```


Average Star Rating Over Time Across Subset Of Categories
Average Star Rating
5
4
3
2
1973
Year
Music
Video
Books
Video_DVD
Sports
Toys
Video_Games
Software
Wireless
Office_Products
Home_Entertainment
Home
Pet_Products
Camera
Electronics
Digital_Ebook_Purchase
Shoes
Lawn_and_Garden
Grocery
Baby
Health_&_Personal_Care
Tools
Musical_Instruments
Outdoors
PC
Home_Improvement
Automotive
Kitchen
Digital_Video_Download
Furniture
Apparel
Personal_Care_Appliances
Digital_Music_Purchase
Major_Appliances
Beauty
Watches
Mobile_Electronics
Jewelry
Luggage
Gift_Card
Digital_Video_Games
Digital_Software
Mobile_Apps

7. Which Star Ratings (1-5) are Most Helpful?

In [56]:
```
# SQL statement
statement = """
SELECT star_rating,
         AVG(helpful_votes) AS avg_helpful_votes
FROM {}.{}
GROUP BY  star_rating
ORDER BY  star_rating ASC
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT star_rating,
         AVG(helpful_votes) AS avg_helpful_votes
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY  star_rating
ORDER BY  star_rating ASC
```

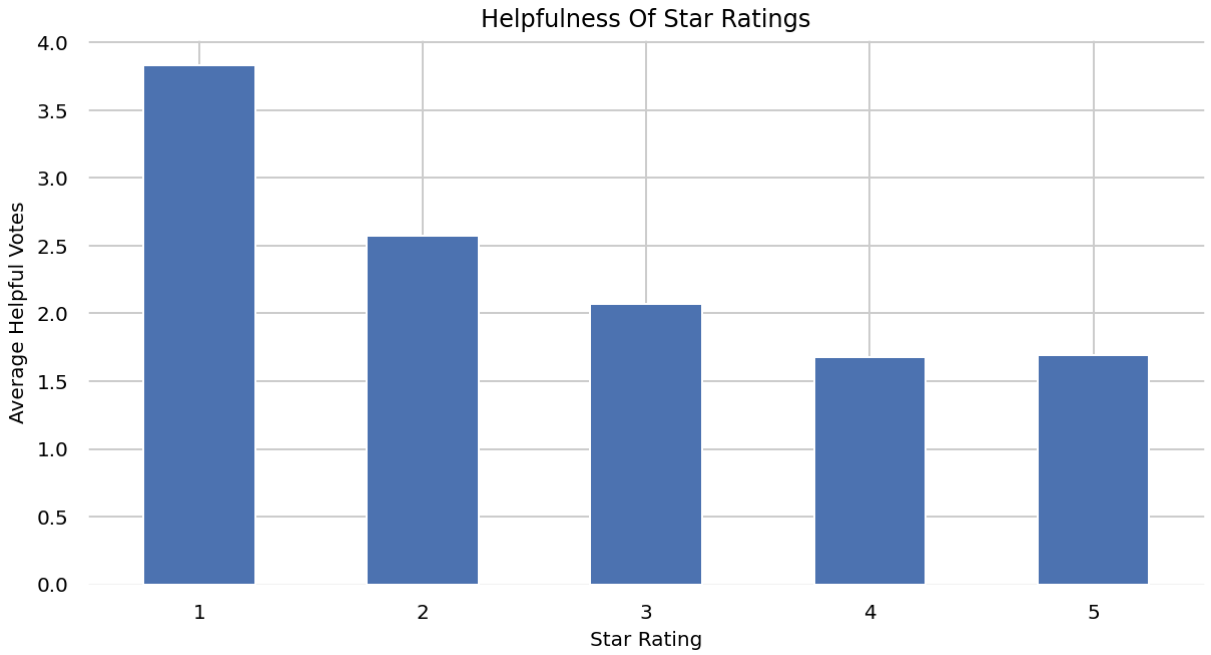
In [57]:
```
df = pd.read_sql(statement, conn)
df
```

Out[57]:

	star_rating	avg_helpful_votes
0	1	3.832321
1	2	2.570054
2	3	2.068739
3	4	1.680050
4	5	1.691202

Visualization for a Subset of Product Categories

In [58]:
```
chart = df.plot.bar(
    x="star_rating", y="avg_helpful_votes", rot="0", figsize=(10, 5), title="Helpfulness Of Star Ratings", legend=False
)

plt.xlabel("Star Rating")
plt.ylabel("Average Helpful Votes")

# chart.get_figure().savefig('helpful-votes.png', dpi=300)
plt.show(chart)
```



8. Which Products have Most Helpful Reviews? How Long are the Most Helpful Reviews?

In [59]:
```
# SQL statement
statement = """
SELECT product_title,
         helpful_votes,
         star_rating,
         LENGTH(review_body) AS review_body_length,
         SUBSTR(review_body, 1, 100) AS review_body_substr
FROM {}.{}
ORDER BY helpful_votes DESC LIMIT 10
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT product_title,
         helpful_votes,
         star_rating,
         LENGTH(review_body) AS review_body_length,
         SUBSTR(review_body, 1, 100) AS review_body_substr
FROM amazonreviewsdb.amazon_reviews_parquet
ORDER BY helpful_votes DESC LIMIT 10
```

In [60]:
```
df = pd.read_sql(statement, conn)
df
```

Out[60]:

	product_title	helpful_votes	star_rating	review_body_length	review_body_substr
0	Kindle: Amazon's Original Wireless Reading Dev...	47524	5	12906	This is less a \\"pros and cons\\" review than...
1	BIC Cristal For Her Ball Pen, 1.0mm, Black, 16...	41393	5	863	Someone has answered my gentle prayers and FIN...
2	The Mountain Kids 100% Cotton Three Wolf Moon ...	41278	5	1566	This item has wolves on it which makes it intr...
3	Kindle Keyboard 3G, Free 3G + Wi-Fi, 6" E Ink ...	31924	5	23069	UPDATE NOVEMBER 2011: My review is ...
4	Kindle Fire HD 7", Dolby Audio, Dual-Band Wi-Fi	31417	4	13594	I've been an iPad user since the original came...
5	Kindle Fire (Previous Generation - 1st)	28611	4	29778	UPDATE November 2012 - With the [[ASIN:B007T36...
6	Fifty Shades of Grey: Book One of the Fifty Sh...	27550	2	2849	I really don't like writing bad reviews. I adm...
7	Fifty Shades of Grey: Book One of the Fifty Sh...	27550	2	2849	I really don't like writing bad reviews. I adm...
8	Wheelmate Laptop Steering Wheel Desk	26132	5	572	My husband Brad always warns me not to try and...
9	Kindle Wireless Reading Device (6" Display, U....	24714	1	10222	I was DELIGHTED to upgrade my Kindle 1 to K2.....

9. What is the Ratio of Positive (5, 4) to Negative (3, 2 ,1) Reviews?

In [61]:
```
# SQL statement
statement = """
SELECT (CAST(positive_review_count AS DOUBLE) / CAST(negative_review_count AS DOUBLE)) AS positive_to
_negative_sentiment_ratio
FROM (
  SELECT count(*) AS positive_review_count
  FROM {}.{}
  WHERE star_rating >= 4
), (
  SELECT count(*) AS negative_review_count
  FROM {}.{}
  WHERE star_rating < 4
)
""".format(
    database_name, table_name, database_name, table_name
)

print(statement)
```

```
SELECT (CAST(positive_review_count AS DOUBLE) / CAST(negative_review_count AS DOUBLE)) AS positive_t
o_negative_sentiment_ratio
FROM (
  SELECT count(*) AS positive_review_count
  FROM amazonreviewsdb.amazon_reviews_parquet
  WHERE star_rating >= 4
), (
  SELECT count(*) AS negative_review_count
  FROM amazonreviewsdb.amazon_reviews_parquet
  WHERE star_rating < 4
)
```

In [62]:
```
df = pd.read_sql(statement, conn)
df
```

Out[62]:

	positive_to_negative_sentiment_ratio
0	3.8332

10. Which Customers are Abusing the Review System by Repeatedly Reviewing the Same Product? What Was Their Average Star Rating for Each Product?

In [65]:
```
# SQL statement
statement = """
SELECT customer_id, product_category, product_title,
ROUND(AVG(star_rating),4) AS avg_star_rating, COUNT(*) AS review_count
FROM {}.{}
GROUP BY customer_id, product_category, product_title
HAVING COUNT(*) > 1
ORDER BY review_count DESC
LIMIT 5
""".format(
    database_name, table_name
)

print(statement)
```

```
SELECT customer_id, product_category, product_title,
ROUND(AVG(star_rating),4) AS avg_star_rating, COUNT(*) AS review_count
FROM amazonreviewsdb.amazon_reviews_parquet
GROUP BY customer_id, product_category, product_title
HAVING COUNT(*) > 1
ORDER BY review_count DESC
LIMIT 5
```

In [66]:
```
df = pd.read_sql(statement, conn)
df
```

Out[66]:

	customer_id	product_category	product_title	avg_star_rating	review_count
0	38118182	Video_DVD	Pearl Harbor	4.2308	260
1	33132919	Video_DVD	Shania Twain - Up (Live in Chicago)	5.0000	220
2	52895956	Books	Frankenstein	3.0299	201
3	29088361	Music	In The Zone	4.9672	122
4	23974294	Video_DVD	Shania Twain - Up (Live in Chicago)	5.0000	114