FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG
Department of Computer Engineering

Experiment 8 - Exploratory Data Analysis and Sentiment Analysis of Drug Reviews

Course Details:

Academic Year	**2023 - 24**	**Estimated Time**	**Experiment No. 8 – 02 Hours**
Course & Semester	**B.E. (COMP) – Sem. VII**	**Subject Name**	**Data Science for Health and Social Care Lab**
Experiment Type	**Software Performance**	**Subject Code**	**HDSSBL701**

Name of Student	Atharva Pawar	**Roll No.**	9427
Date of Performance.:		**Date of Submission.:**	
CO Mapping	**HDSSBL701.3 -** Demonstrate statistical data analysis and visualization techniques on healthcare data.		

Aim: Exploratory Data Analysis and Sentiment Analysis of Drug reviews.

Objective:

1. To plot a scatter plot on rating verses useful Count.
2. To draw bar graph of number of reviews per condition (depression , panic condition etc.)
3. To Perform sentiment analysis using TextBlob module of python for text classification as positive, negative and neutral.

Steps:

1. Search and collect Drug Review data set from the UCI machine learning repository
2. Perform EDA and SA on the drug review dataset.

Result:

Reference:

http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462022000301191

Exp - 8

About Dataset

- The Drug Review Dataset is taken from the UCI Machine Learning Repository.**It has 7 features** including the review and **161297 Data Points** or entries.

Attributes :

1. **drugName :** name of drug 2.**condition :** name of condition
2. **review :** patient review
3. **rating :** 10 star patient rating
4. **date :** date of review entry
5. **usefulCount :** number of users who found review useful

In [1]:
```
# Load EDA Pkgs
import pandas as pd
import numpy as np
```

In [2]:
```
# Load Data Viz
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as px
plt.style.use('fivethirtyeight')
%matplotlib inline
```

In [3]:
```
# Load Sentiment Pkgs
from textblob import TextBlob
```

Loading the Datset

In [4]:
```
# Load Dataset
df = pd.read_csv("../input/kuc-hackathon-winter-2018/drugsComTrain_raw.csv")
```

In [5]:
```
# Preview Dataset
df.head()
```

Out[5]:

	uniqueID	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8	3-Nov-15	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9	27-Nov-16	37

Data Exploration

In [6]:
```
# Columns
df.columns
```

Out[6]:
```
Index(['uniqueID', 'drugName', 'condition', 'review', 'rating', 'date',
       'usefulCount'],
      dtype='object')
```

In [7]:
```
# Missing Values
df.isnull().sum()
```

Out[7]:
```
uniqueID         0
drugName         0
condition      899
review           0
rating           0
date             0
usefulCount      0
dtype: int64
```

Observation :

- Most of the missing values are in the condition column
- This implies that most people don't know their condition by name or privacy

In [8]:
```
df.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
uniqueID	161297.0	115923.585305	67004.445170	2.0	58063.0	115744.0	173776.0	232291.0
rating	161297.0	6.994377	3.272329	1.0	5.0	8.0	10.0	10.0
usefulCount	161297.0	28.004755	36.403742	0.0	6.0	16.0	36.0	1291.0

In [9]:
```
df.info()
print('==============='*10)
df.dtypes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161297 entries, 0 to 161296
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   uniqueID     161297 non-null  int64 
 1   drugName     161297 non-null  object
 2   condition    160398 non-null  object
 3   review       161297 non-null  object
 4   rating       161297 non-null  int64 
 5   date         161297 non-null  object
 6   usefulCount  161297 non-null  int64 
dtypes: int64(3), object(4)
memory usage: 8.6+ MB
==========================================================================================================================================
=============
```

Out[9]:
```
uniqueID        int64
drugName       object
condition      object
review         object
rating          int64
date           object
usefulCount     int64
dtype: object
```

EDA and Data Visualization

-- How many drugs do we have?

In [11]:
```
# How many drugs do we have?
num = len(df['drugName'].unique().tolist())
print('Number of Drugs are -',num )
```

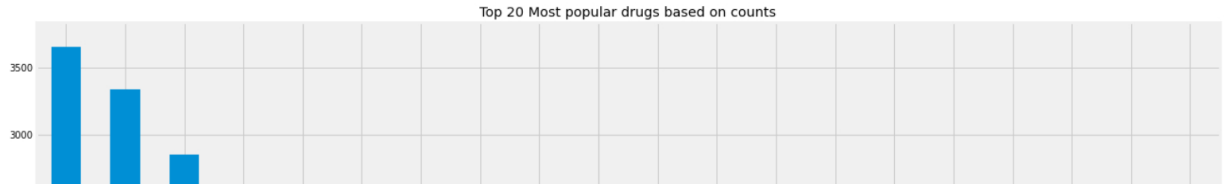
```
Number of Drugs are - 3436
```

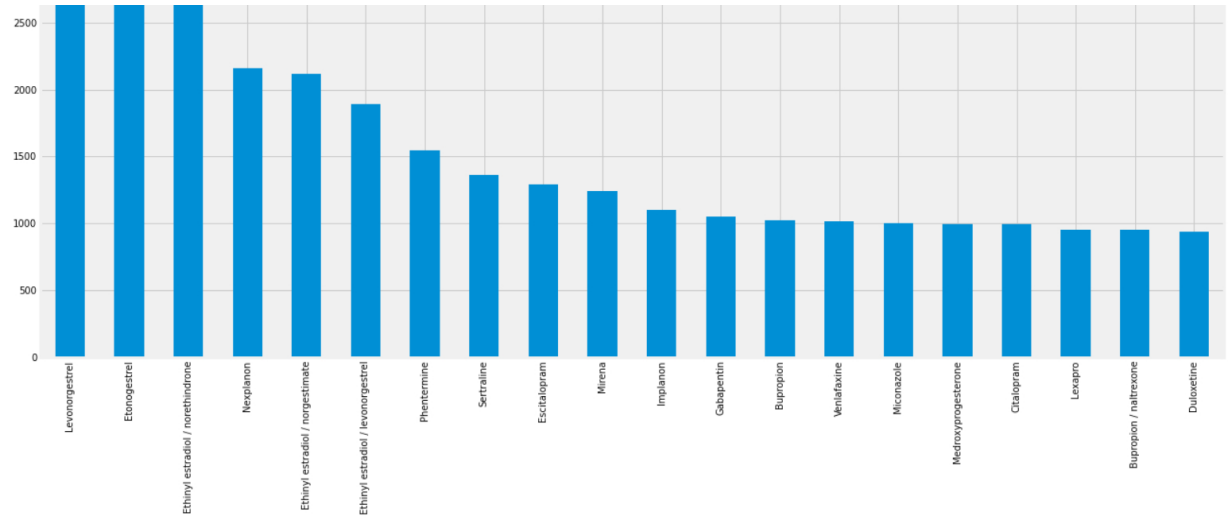
--- What are the Most popular Drug?

In [12]:
```
# What is the most popular drug?
# Top 20 Drugs (Most Popular)
df['drugName'].value_counts().nlargest(20)
```

Out[12]:
```
Levonorgestrel                      3657
Etonogestrel                        3336
Ethinyl estradiol / norethindrone    2850
Nexplanon                           2156
Ethinyl estradiol / norgestimate     2117
Ethinyl estradiol / levonorgestrel   1888
Phentermine                         1543
Sertraline                          1360
Escitalopram                        1292
Mirena                              1242
Implanon                            1102
Gabapentin                          1047
Bupropion                           1022
Venlafaxine                         1016
Miconazole                          1000
Medroxyprogesterone                  995
Citalopram                           995
Lexapro                              952
Bupropion / naltrexone               950
Duloxetine                           934
Name: drugName, dtype: int64
```

In [13]:
```
# Top 20 Drugs (Most Popular)
plt.figure(figsize=(20,10))
df['drugName'].value_counts().nlargest(20).plot(kind='bar')
plt.title("Top 20 Most popular drugs based on counts")
plt.show()
```

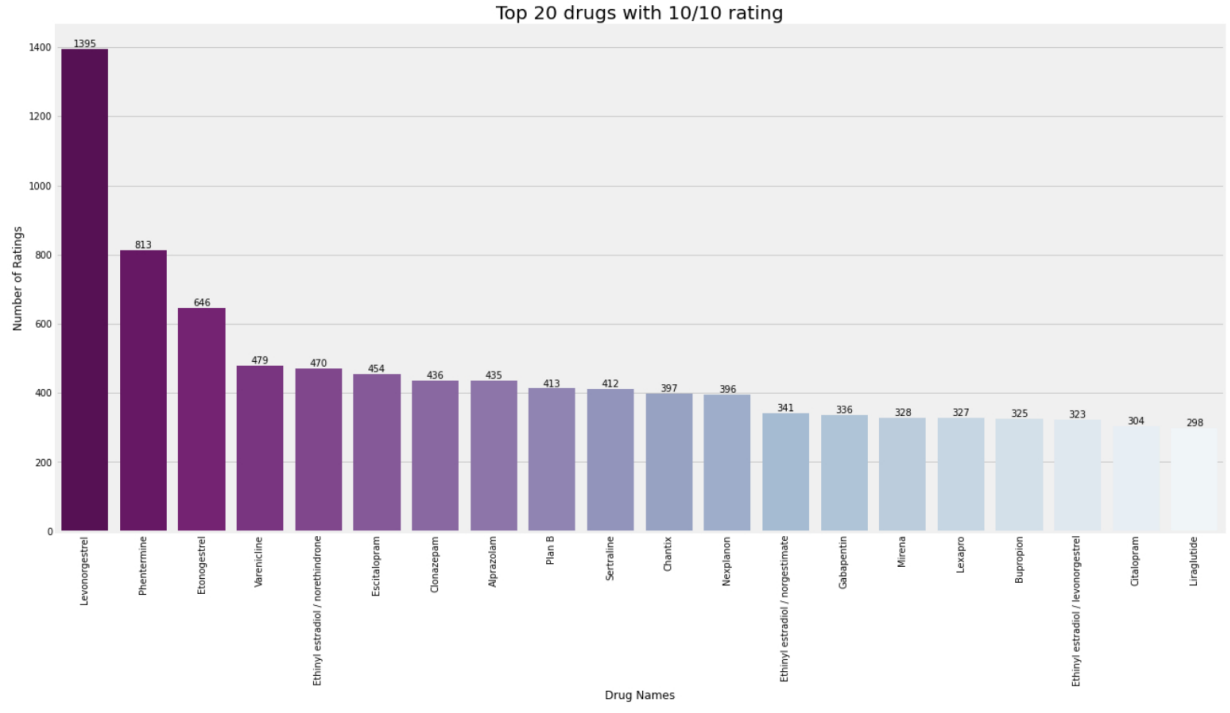




Observation

- Most of the commonest drugs are hormonal drugs

In [14]:
```
# Top 20 Drugs (Most Popular)
plt.figure(figsize=(20,10))
rating = dict(df.loc[df.rating == 10, "drugName"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())

sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20], palette = 'BuPu_r')

for i in sns_rating.containers:
    sns_rating.bar_label(i,)

sns_rating.set_title('Top 20 drugs with 10/10 rating', fontsize=20)
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
plt.setp(sns_rating.get_xticklabels(), rotation=90)
plt.show()
```

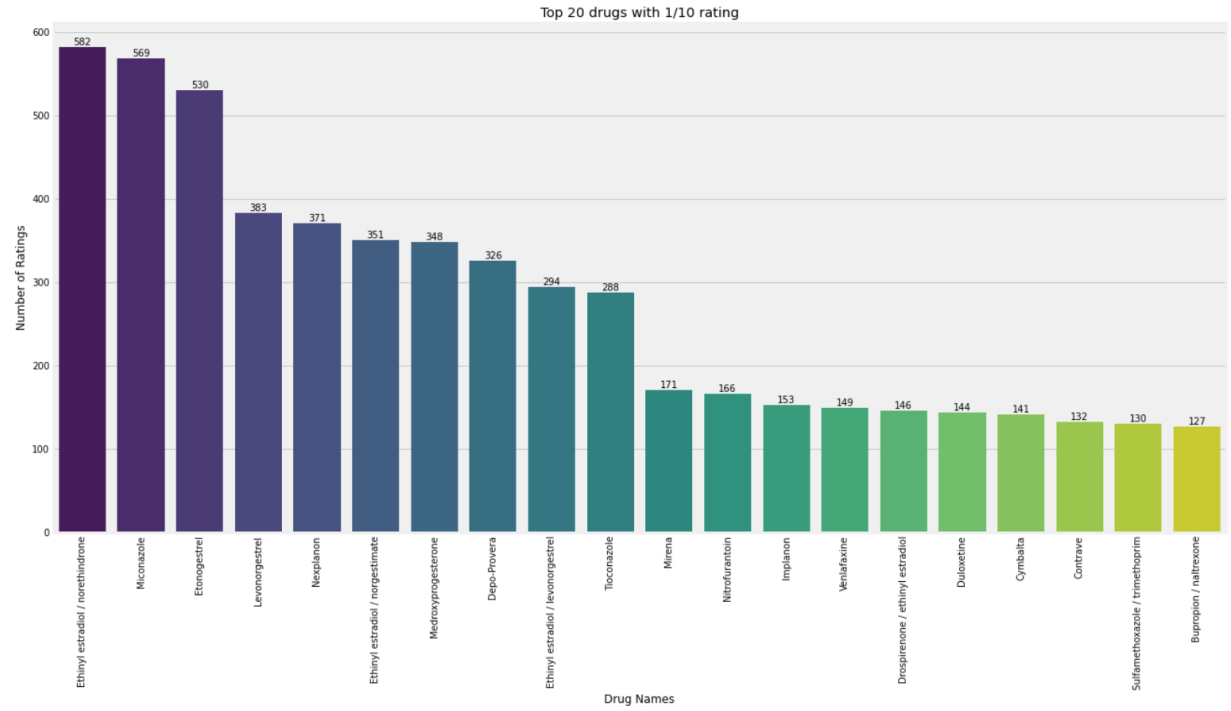


Observation :

Levonorgestrel, Phentemine, Etonogestrel are the Durgs with the **Heighest Ratings**

In [15]:
```
# Least 20 Drugs (Most Popular)
plt.figure(figsize=(20,10))
rating = dict(df.loc[df.rating == 1, "drugName"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())

sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20], palette = 'viridis')
```

```
for i in sns_rating.containers:
    sns_rating.bar_label(i,)

sns_rating.set_title('Top 20 drugs with 1/10 rating')
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
plt.setp(sns_rating.get_xticklabels(), rotation=90)
plt.show()
```

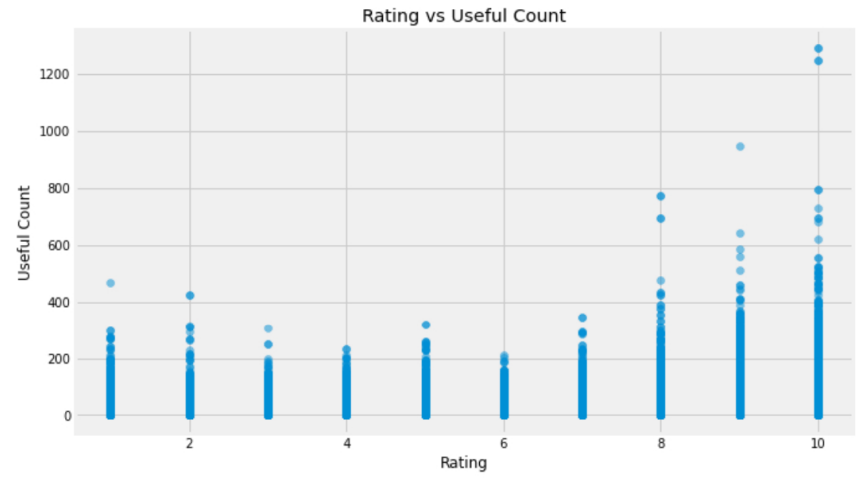


Observation :

Norethindrone, Miconzole are the Drugs with the **Least ratings** given by users

Plot a scatter plot on rating verses useful Count.

In [72]:
```
def plot_scatter_rating_vs_useful_count(dataframe):
    plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
    plt.scatter(dataframe['rating'], dataframe['usefulCount'], alpha=0.5)
    plt.title('Rating vs Useful Count')
    plt.xlabel('Rating')
    plt.ylabel('Useful Count')
    plt.grid(True)
    plt.show()

# Call the function to create the scatter plot
plot_scatter_rating_vs_useful_count(df)
```



-- What are the groups/classification of drugs used?

- suffix or endings

In [16]:
```
drug_suffix = {"azole":"antifungal (except metronidazole)",
"caine":"anesthetic",
"cillin":"antibiotic(penicillins)",
"mycin":"antibiotic",
"micin":"antibiotic",
"cycline":"antibiotic",
"oxacin":"antibiotic",
"ceph":"antibiotic(cephalosporins)",
"cef":"antibiotic (cephalosporins)",
"dine":"h2 blockers (anti-ulcers)",
"done":"opiod analgesics",
"ide":"oral hypoglycemics",
"lam":"anti-anxiety",
"pam":"anti-anxiety",
"mide":"diuretics",
"zide":"diuretics",
"nium":"neuromuscular blocking agents",
"olol":"beta blockers",
"tidine":"h2 antagonist",
"tropin":"pituitary hormone",
"zosin":"alpha blocker",
"ase":"thrombolytics",
"plase":"thrombolytics",
"azepam":"anti-anziety(benzodiazepine)",
"azine":"antipyschotics (phenothiazine)",
"barbital":"barbiturate",
"dipine":"calcium channel blocker",
"lol":"beta blocker",
"zolam":"cns depressants",
"pril":"ace inhibitor",
"artan":"arb blocker",
"statins":"lipid-lowering drugs",
"parin":"anticoagulants",
"sone":"corticosteroid (prednisone)"}
```

In [17]:
```
def classify_drug(drugname):
    for i in drug_suffix.keys():
        if drugname.endswith(i):
            return drug_suffix[i]
```

In [18]:
```
classify_drug('valsartan')
```

Out[18]:
```
'arb blocker'
```

In [19]:
```
df['drug_class'] = df['drugName'].apply(classify_drug)
```

In [20]:
```
df[['drugName','drug_class']]
```

Out[20]:

	drugName	drug_class
0	Valsartan	arb blocker
1	Guanfacine	None
2	Lybrel	None
3	Ortho Evra	None
4	Buprenorphine / naloxone	None
...	...	...
161292	Campral	None
161293	Metoclopramide	oral hypoglycemics
161294	Orencia	None
161295	Thyroid desiccated	None
161296	Lubiprostone	None

161297 rows × 2 columns

In [21]:
```
# How many Groups of Drugs By Class
df['drug_class'].unique().tolist()
```

Out[21]:
```
['arb blocker',
 None,
 'antifungal (except metronidazole)',
 'oral hypoglycemics',
 'opiod analgesics',
 'antibiotic',
 'anti-anxiety',
 'h2 blockers (anti-ulcers)',
 'beta blockers',
 'ace inhibitor',
 'thrombolytics',
 'alpha blocker',
 'corticosteroid (prednisone)',
 'antipyschotics (phenothiazine)',
 'antibiotic(penicillins)',
 'barbiturate',
 'calcium channel blocker',
 'anesthetic',
 'pituitary hormone',
 'antibiotic (cephalosporins)',
 'beta blocker',
```

```
 'neuromuscular blocking agents',
 'anticoagulants']
```

In [22]:
```
# How many Groups of Drugs By Class
grp_drugs = len(df['drug_class'].unique().tolist())
print('Groups of Drugs by Class - ',grp_drugs)
```

```
Groups of Drugs by Class -  23
```

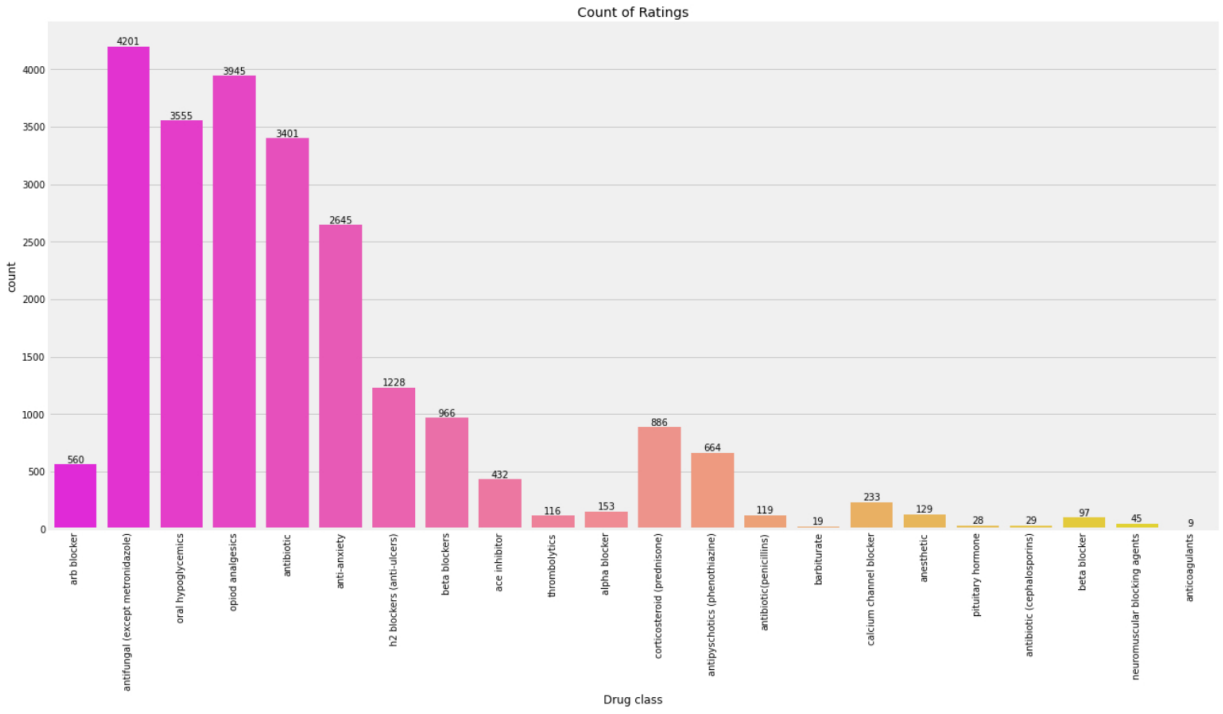
-- Which class of Drug is the most common ?

In [23]:
```
# Which of class of drug  is the most commonest
df['drug_class'].value_counts()
```

Out[23]:
```
antifungal (except metronidazole)    4201
opiod analgesics                     3945
oral hypoglycemics                   3555
antibiotic                           3401
anti-anxiety                         2645
h2 blockers (anti-ulcers)            1228
beta blockers                         966
corticosteroid (prednisone)           886
antipyschotics (phenothiazine)        664
arb blocker                           560
ace inhibitor                         432
calcium channel blocker               233
alpha blocker                         153
anesthetic                            129
antibiotic(penicillins)               119
thrombolytics                         116
beta blocker                           97
neuromuscular blocking agents          45
antibiotic (cephalosporins)            29
pituitary hormone                      28
barbiturate                            19
anticoagulants                          9
Name: drug_class, dtype: int64
```

In [24]:
```
# Which of class of drug  is the most commonest

plt.figure(figsize=(20,10))
sns_1 = sns.countplot(df['drug_class'], palette = 'spring')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Count of Ratings')
sns_1.set_xlabel("Rating")
sns_1.set_xlabel("Drug class")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```



Observations :

Observations :

- The most commonest class/group of drugs used are
  - **Antifungal**
  - **Opiod Analgesics(Pain Killers)**
  - **Oral Hypoglycemics (DM)**
  - **Antibiotic**

-- Distribution of Drugs(class) Per Drug Group based on size

In [25]:
```
# Distribution of Drugs Per Drug Group based on size
drug_groups = df.groupby('drug_class').size()
```

In [26]:
```
type(drug_groups)
```

Out[26]:
```
pandas.core.series.Series
```

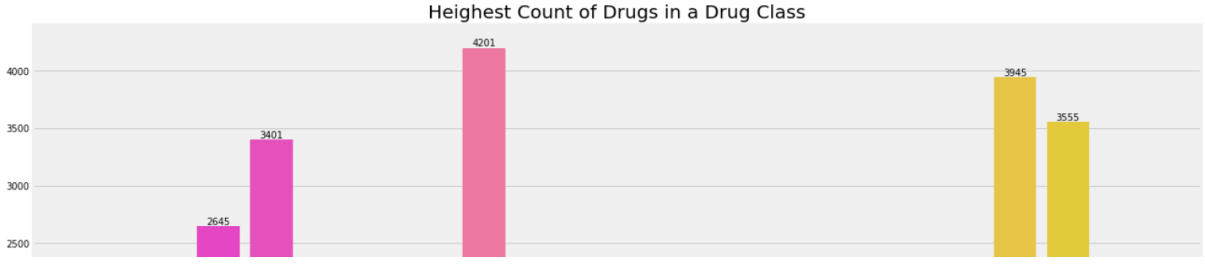
In [27]:
```
# Convert to DF
# Method 1
drug_groups.to_frame()
```

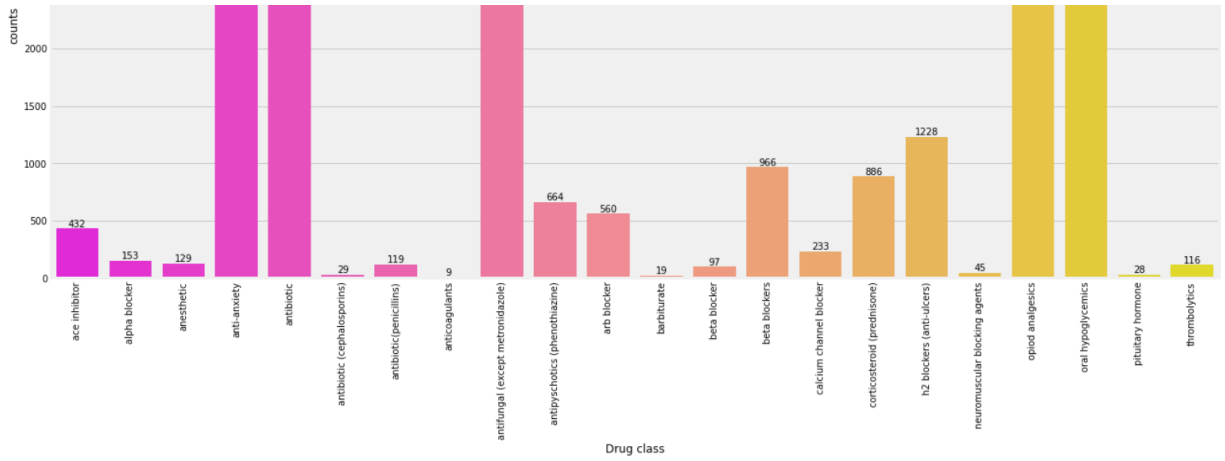
Out[27]:

drug_class	0
ace inhibitor	432
alpha blocker	153
anesthetic	129
anti-anxiety	2645
antibiotic	3401
antibiotic (cephalosporins)	29
antibiotic(penicillins)	119
anticoagulants	9
antifungal (except metronidazole)	4201
antipyschotics (phenothiazine)	664
arb blocker	560
barbiturate	19
beta blocker	97
beta blockers	966
calcium channel blocker	233
corticosteroid (prednisone)	886
h2 blockers (anti-ulcers)	1228
neuromuscular blocking agents	45
opiod analgesics	3945
oral hypoglycemics	3555
pituitary hormone	28
thrombolytics	116

In [28]:
```
# Convert to DF
# Method 2
drug_groups_df = pd.DataFrame({'drug_class':drug_groups.index,'counts':drug_groups.values})
```

In [29]:
```
# Seaborn Plot
plt.figure(figsize=(20,10))
# g = sns.barplot(data=drug_groups_df,x='drug_class',y='counts', palette='gnuplot_r')

sns_1 = sns.barplot(data=drug_groups_df,x='drug_class',y='counts', palette = 'spring')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Heighest Count of Drugs in a Drug Class', fontsize=20)
sns_1.set_xlabel("count")
sns_1.set_xlabel("Drug class")


sns_1.set_xticklabels(drug_groups_df['drug_class'].values,rotation=90)
plt.show()
```





Observations :

Here we can see **Antifungal, Opiod Analgesics, Oral Hypoglycemeics** is the Drug Class with **heighest count of Drugs**

-- How many Conditions are suffered by patients?

In [30]:
```
len(df['condition'].unique().tolist())
```

Out[30]:
```
885
```

Observation :

- We have 885 different conditions

In [31]:
```
#### Distribution of Conditions
df['condition'].value_counts()
```

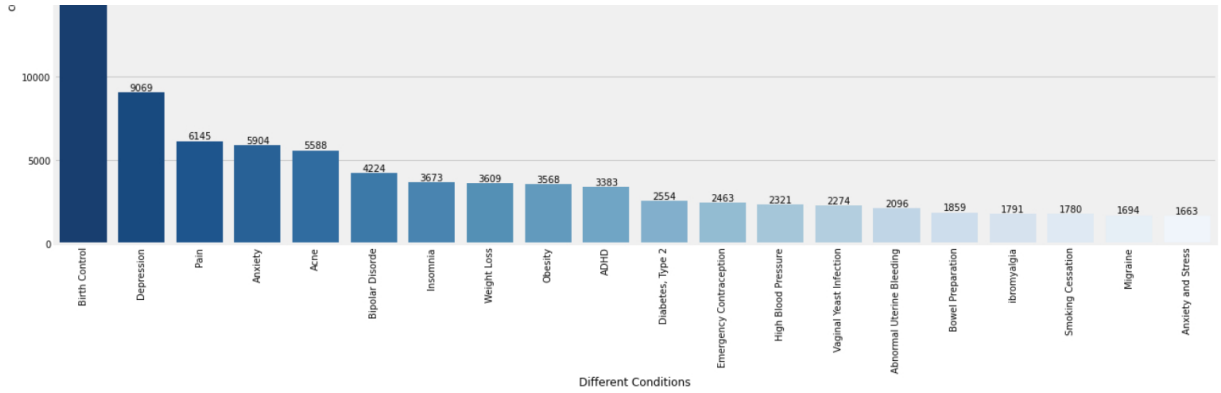
Out[31]:
```
Birth Control                                        28788
Depression                                            9069
Pain                                                  6145
Anxiety                                               5904
Acne                                                  5588
                                                     ...
Dissociative Identity Disorde                            1
Hydrocephalus                                            1
Hyperlipoproteinemia Type III, Elevated beta-VLDL   IDL      1
Q Feve                                                   1
Neutropenia                                              1
Name: condition, Length: 884, dtype: int64
```

-- Most common condition suffered

In [32]:
```
#### Most commonest Conditions
common_conditions = df['condition'].value_counts().nlargest(20)
```

In [33]:
```
# Change the Variable Name
new = pd.DataFrame({'drug_class':common_conditions.index,'counts':common_conditions.values})
```

In [34]:
```
#### Most commonest Conditions
plt.figure(figsize=(20,10))

sns_1 = sns.barplot(data=new,x='drug_class',y='counts', palette = 'Blues_r')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Most suffered Conditions', fontsize=20)
sns_1.set_xlabel("Rating")
sns_1.set_xlabel("Different Conditions")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```



Observations :

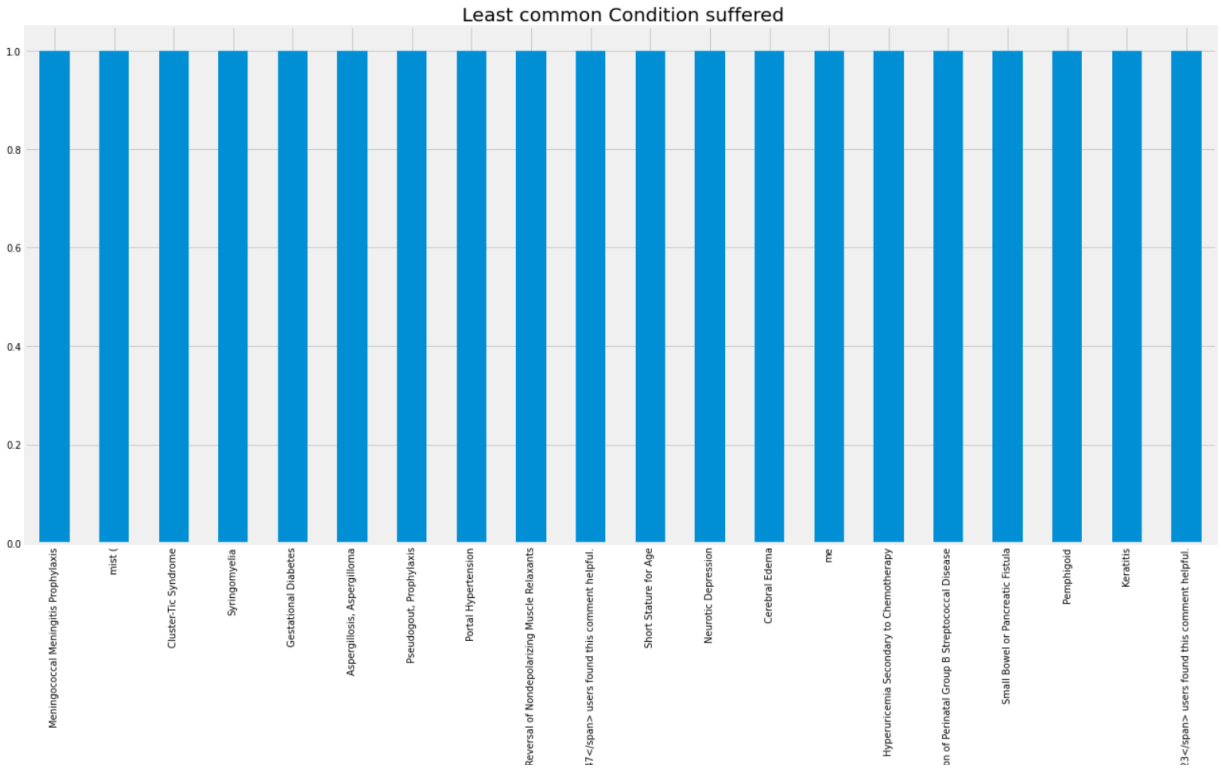
- The most commonest condition is **Birth Control**,followed by **Depression** and **Pain and Anxiety**
- Makes sense as compared to the drug distribution

-- Least common Condition suffered

In [35]:
```
df['condition'].value_counts().nsmallest(20)
```

Out[35]:
```
Meningococcal Meningitis Prophylaxis                    1
mist (                                                  1
Cluster-Tic Syndrome                                    1
Syringomyelia                                           1
Gestational Diabetes                                    1
Aspergillosis, Aspergilloma                             1
Pseudogout, Prophylaxis                                 1
Portal Hypertension                                     1
Reversal of Nondepolarizing Muscle Relaxants            1
47</span> users found this comment helpful.             1
Short Stature for Age                                   1
Neurotic Depression                                     1
Cerebral Edema                                          1
me                                                      1
Hyperuricemia Secondary to Chemotherapy                 1
Prevention of Perinatal Group B Streptococcal Disease   1
Small Bowel or Pancreatic Fistula                       1
Pemphigoid                                              1
Keratitis                                               1
123</span> users found this comment helpful.            1
Name: condition, dtype: int64
```

In [36]:
```
#### Least commonest Conditions
df['condition'].value_counts().nsmallest(20).plot(kind='bar',figsize=(20,10))
plt.title(' Least common Condition suffered', fontsize=20)
plt.show()
```

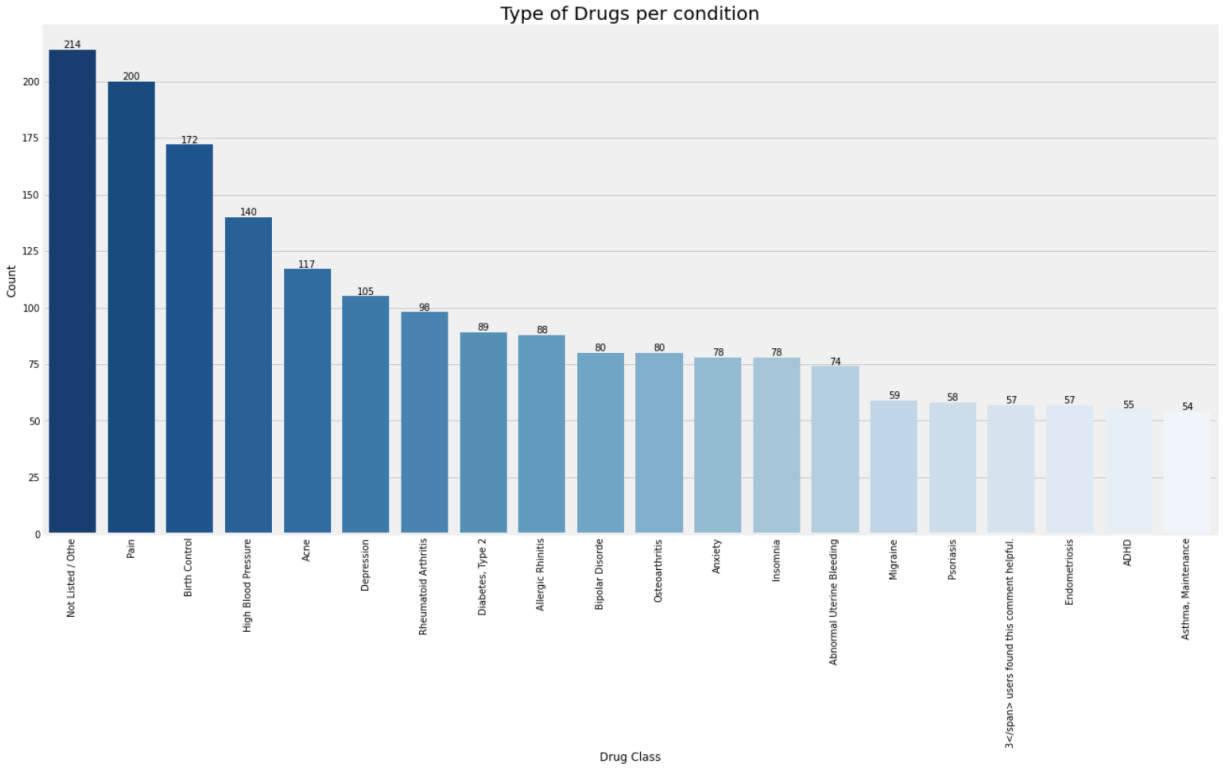


-- How many type of Drugs per condition (Top 20)

In [37]:
```
drug_per_cond = df.groupby('condition')['drugName'].nunique().nlargest(20)
```

In [38]:
```
drug_per_cond_df = pd.DataFrame({'drug_class':drug_per_cond.index,'counts':drug_per_cond.values})
```

In [39]:
```
# How many Drugs per condition (Top 20)
plt.figure(figsize=(20,10))

sns_1 = sns.barplot(data=drug_per_cond_df,x='drug_class',y='counts', palette = 'Blues_r')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Type of Drugs per condition', fontsize=20)
sns_1.set_xlabel("Drug Class")
sns_1.set_ylabel("Count")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```

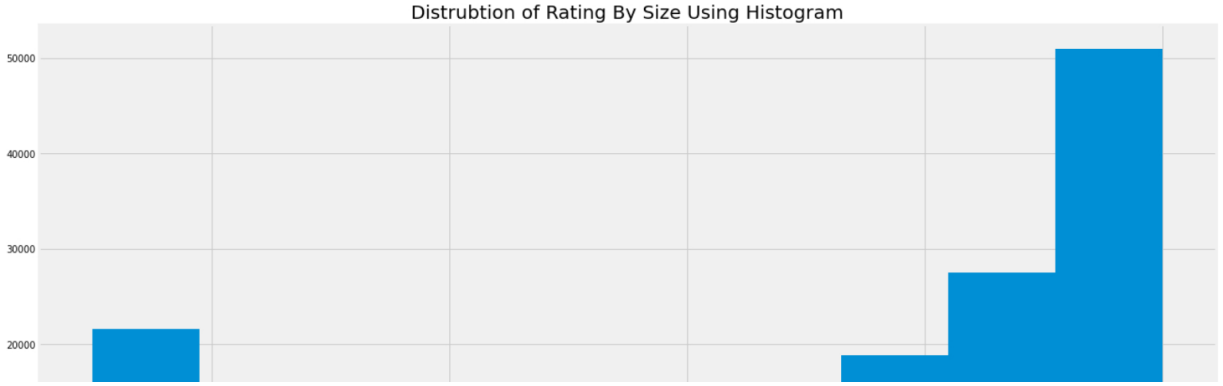


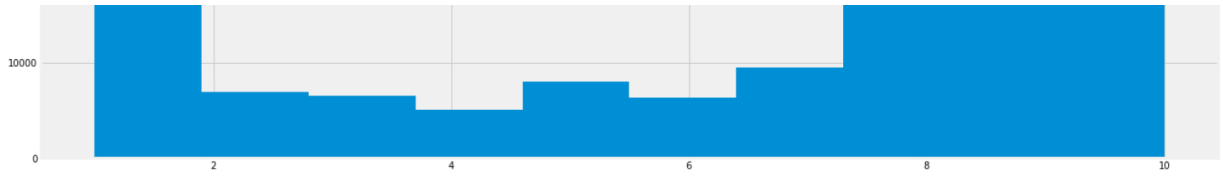
Observation :

- **Pain, Birth Control** and **HBP** have the highest number of different/unique drugs for their condition

-- Distrubtion of Rating given by Users

In [40]:
```
plt.figure(figsize=(20,10))
df['rating'].hist()
plt.title("Distrubtion of Rating By Size Using Histogram", fontsize=20)
plt.show()
```





Observation :

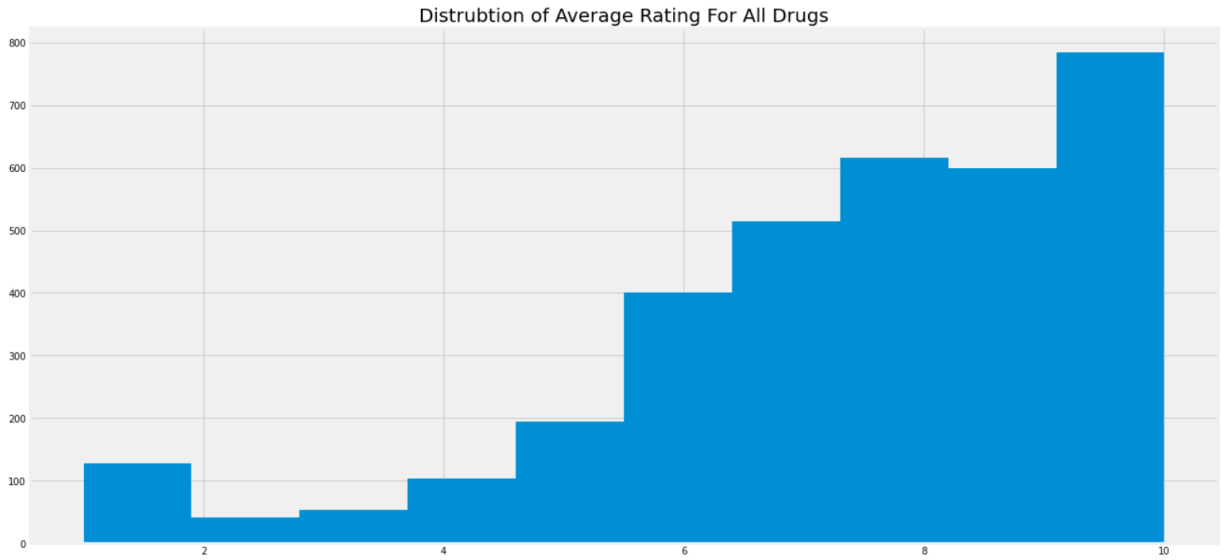
- Most people rated at the **extremes i.e (0 and 10)**

-- Average Rating of Drugs

In [41]:
```
# Average Rating of Drugs
avg_rating = (df['rating'].groupby(df['drugName']).mean())
avg_rating
```

Out[41]:
```
drugName
A + D Cracked Skin Relief              10.000000
A / B Otic                             10.000000
Abacavir / dolutegravir / lamivudine    8.211538
Abacavir / lamivudine / zidovudine      9.000000
Abatacept                               7.157895
                                          ...
Zyvox                                   9.000000
ZzzQuil                                 2.500000
depo-subQ provera 104                   1.000000
ella                                    6.980392
femhrt                                  4.000000
Name: rating, Length: 3436, dtype: float64
```

In [42]:
```
# Average Rating For All Drugs
plt.figure(figsize=(20,10))
avg_rating.hist()
plt.title("Distrubtion of Average Rating For All Drugs", fontsize=20)
plt.show()
```



Observation :

The **Majority of Rating** given by the users are **from 6 to 10**

Insights on Date Column :

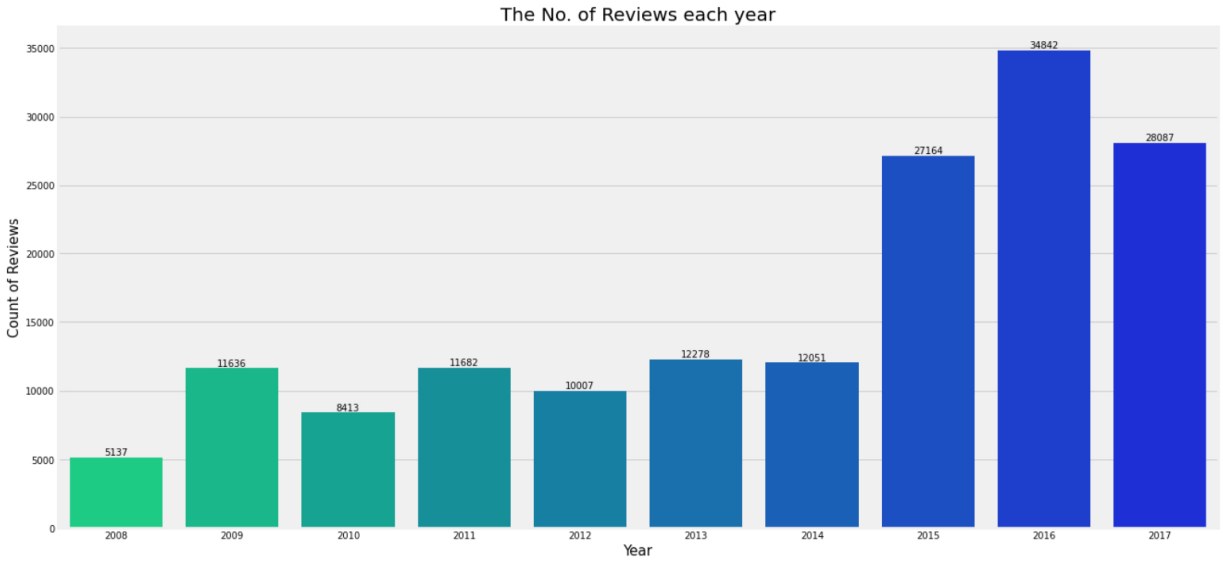
Transforming date columns

In [43]:
```
# converting the date into datetime format
df['date'] = pd.to_datetime(df['date'], errors = 'coerce')

# extracting year from date
df['Year'] = df['date'].dt.year

# extracting the month from the date
df['month'] = df['date'].dt.month

# extracting the days from the date
df['day'] = df['date'].dt.day
```

-- Number of Reviews each Year

In [44]:
```
plt.figure(figsize=(20,10))
sns_=sns.countplot(df['Year'], palette ='winter_r')

for i in sns_.containers:
    sns_.bar_label(i,)

plt.title('The No. of Reviews each year', fontsize = 20)
plt.xlabel('Year', fontsize = 15)
plt.ylabel('Count of Reviews', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```



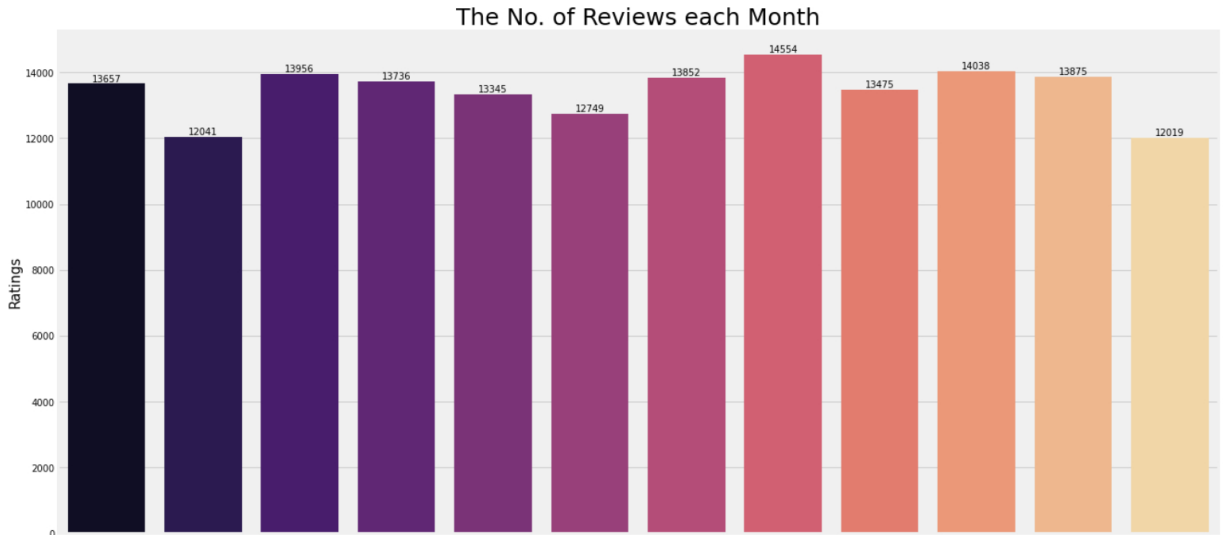
Observations :

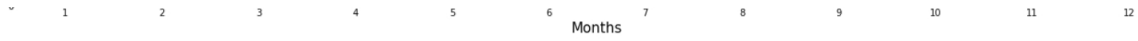
From above plot we can see that **Year 2016 had the heighest number of reviews** followed by **Year 2017 and 2015**

-- Number of Reviews each Month

In [45]:
```
plt.figure(figsize=(20,10))
sns_=sns.countplot(df['month'], palette ='magma')

for i in sns_.containers:
    sns_.bar_label(i,)

plt.title('The No. of Reviews each Month', fontsize = 25)
plt.xlabel('Months', fontsize = 15)
plt.ylabel('Ratings', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```



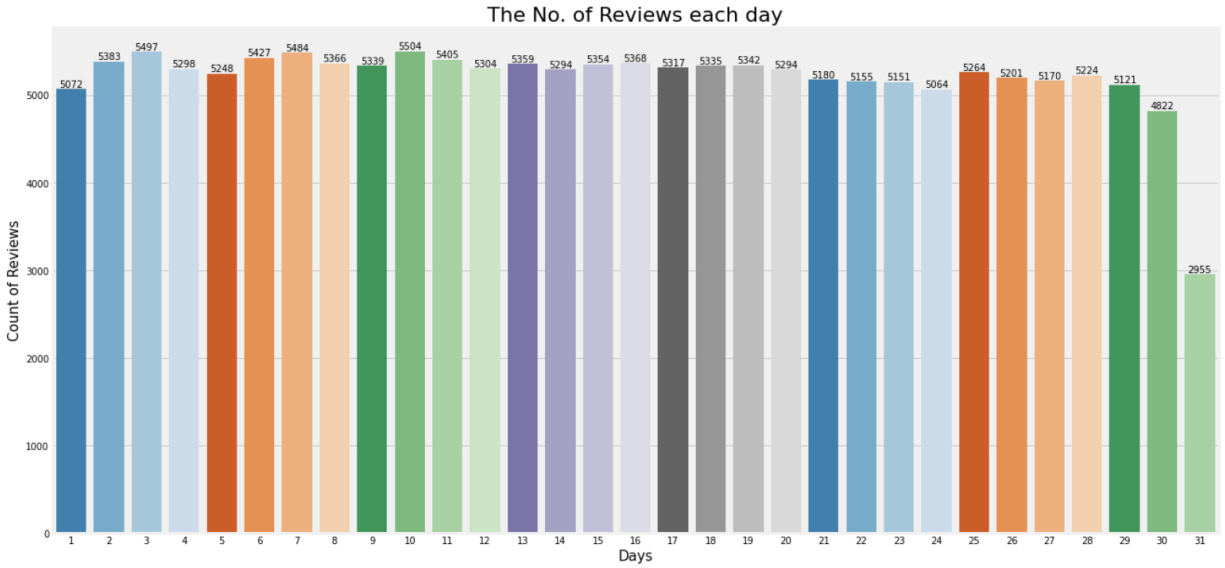


Observations :

August month has the heighest reviwes **followed by October** month.

In [46]:
```
plt.figure(figsize=(20,10))
sns_=sns.countplot(df['day'], palette ='tab20c')

for i in sns_.containers:
    sns_.bar_label(i)

plt.title('The No. of Reviews each day', fontsize = 22)
plt.xlabel('Days', fontsize = 15)
plt.ylabel('Count of Reviews', fontsize = 15)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



-- How genuine is the review? (Using sentiment analysis)

In [47]:
```
from textblob import TextBlob
```

In [48]:
```
df['review'].head()
```

Out[48]:
```
0    "It has no side effect, I take it in combinati...
1    "My son is halfway through his fourth week of ...
2    "I used to take another oral contraceptive, wh...
3    "This is my first time using any form of birth...
4    "Suboxone has completely turned my life around...
Name: review, dtype: object
```

In [49]:
```
def get_sentiment(text):
    blob = TextBlob(text)
    return blob.polarity

def get_sentiment_label(text):
    blob = TextBlob(text)
    if blob.polarity > 0:
        result = 'positive'
    elif blob.polarity < 0:
        result = 'negative'
    else:
        result = 'neutral'
    return result
```

In [50]:
```
# text fxn
get_sentiment("I love apples")
```

Out[50]:
```
0.5
```

In [51]:
```
# text fxn
get_sentiment_label("I love apples")
```

Out[51]:
```
'positive'
```

In [52]:
```
# Sentiment Score for Review
df['sentiment'] = df['review'].apply(get_sentiment)
```

In [53]:
```
# Sentiment Labels for Review
df['sentiment_label'] = df['review'].apply(get_sentiment_label)
```

In [54]:
```
df[['review','sentiment','sentiment_label']].head()
```

Out[54]:

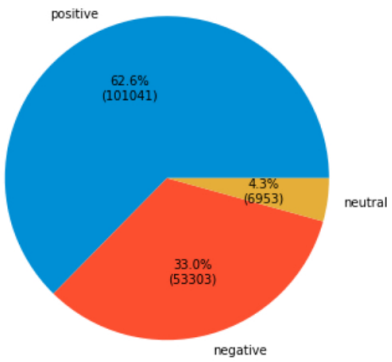
	review	sentiment	sentiment_label
0	"It has no side effect, I take it in combinati...	0.000000	neutral
1	"My son is halfway through his fourth week of ...	0.168333	positive
2	"I used to take another oral contraceptive, wh...	0.067210	positive
3	"This is my first time using any form of birth...	0.179545	positive
4	"Suboxone has completely turned my life around...	0.194444	positive

-- How many positive and negative and neutral reviews?

In [55]:
```
df['sentiment_label'].value_counts()
```

Out[55]:
```
positive    101041
negative     53303
neutral       6953
Name: sentiment_label, dtype: int64
```

In [56]:
```
plt.figure(figsize=(12,6))

def autopct_format(values):
        def my_format(pct):
            total = sum(values)
            val = int(round(pct*total/100.0))
            return '{:.1f}%\n({v:d})'.format(pct, v=val)
        return my_format

s = df['sentiment_label'].value_counts()
plt.pie(s,labels = s.index, autopct=autopct_format(s))
plt.show()
```

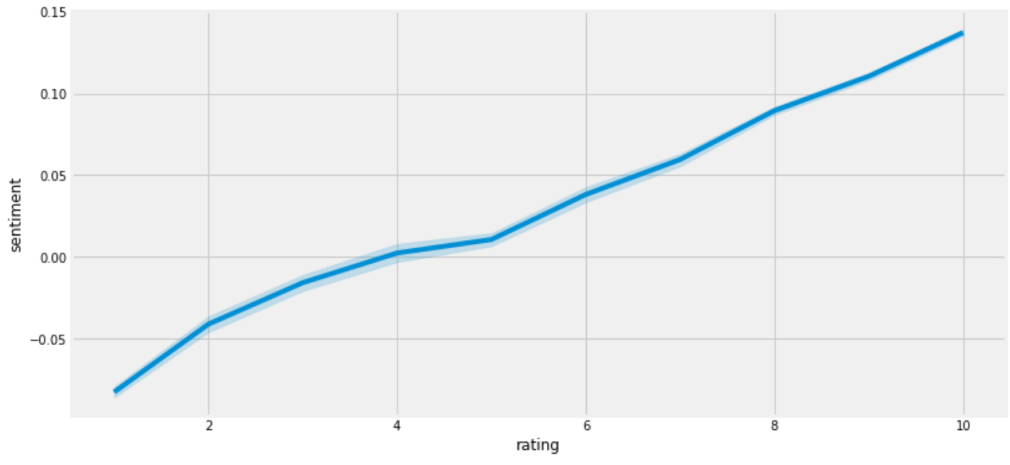


Obserevation :

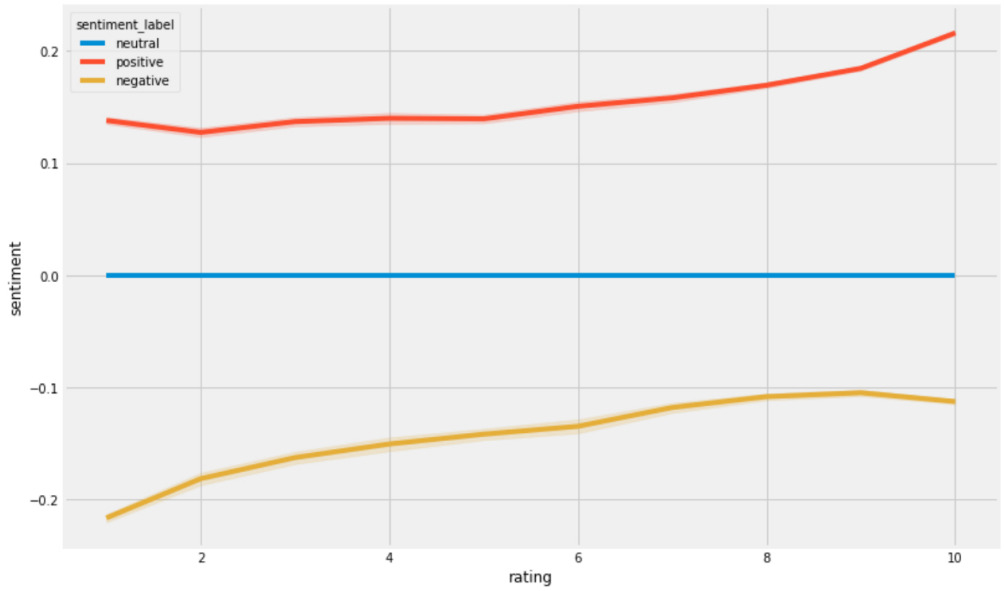
We can see that Majority of reviews given by users are **Positive(62.6%)** and **Negative(33%)**, while **Neutral reviews are (4.3%)**

-- Correlation Between Our sentiment and rating

In [57]:
```
#### Correlation Between Our sentiment and rating
plt.figure(figsize=(12,6))
sns.lineplot(data=df,x='rating',y='sentiment')
plt.show()
```



In [58]:
```
plt.figure(figsize=(12,8))
sns.lineplot(data=df,x='rating',y='sentiment',hue='sentiment_label')
plt.show()
```



Observation :

- The positive Ratings are increasing gradually which indicates that the patients are getting better by the use of the prescribed Drugs
- The Negative Ratings are decreasing which is a good sign that the Drugs given to patients are effective

-- How many reviews are genuine as compared to the rating

- Genuine **good rating** =positive + rating 10-6
- Genuine **bad rating** = negative + rating 4-1

In [59]:
```
# Genuine Good  Rating Per Review
good_review =  len(df[(df['rating'] >= 6) & (df['sentiment_label'] == 'positive')])
print('According to Sentiment and Rating the number of Good Ratings are -', good_review)
```

```
According to Sentiment and Rating the number of Good Ratings are - 81044
```

In [60]:
```
# Genuine Bad  Rating Per Review
bad_review = len(df[(df['rating'] <= 4) & (df['sentiment_label'] == 'negative')])
print('According to Sentiment and Rating the number of Good Ratings are -', bad_review)
```

```
According to Sentiment and Rating the number of Good Ratings are - 21995
```

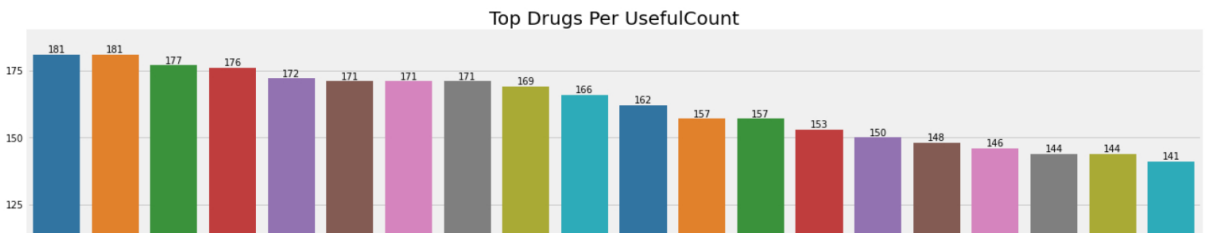
In [61]:
```
# df.groupby('drugName')['usefulCount'].value_counts()
```

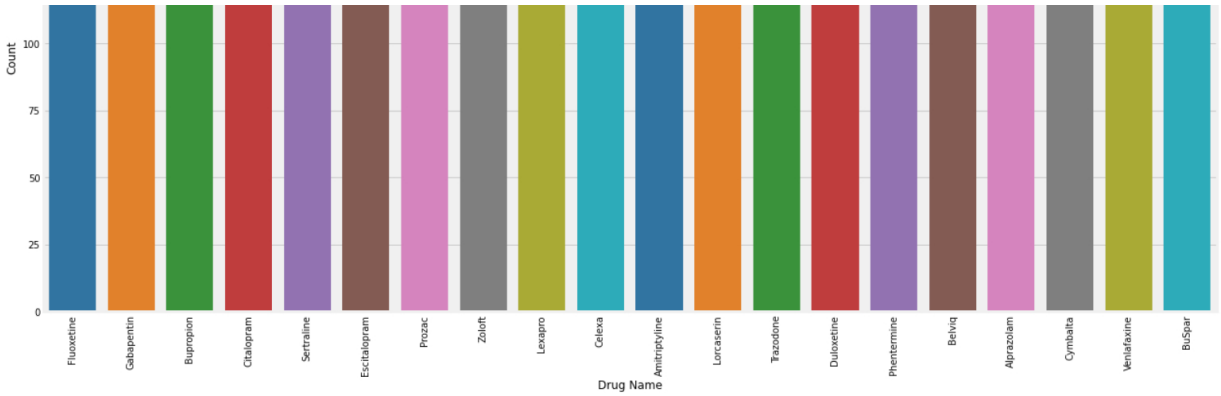
-- Top Drugs name Per UsefulCount

In [62]:
```
drug_usefulcount = df.groupby('drugName')['usefulCount'].nunique().nlargest(20)
```

In [63]:
```
drug_usefulcount_df = pd.DataFrame({'drug_name':drug_usefulcount.index,'counts':drug_usefulcount.values})
```

In [64]:
```
plt.figure(figsize=(20,10))

sns_1 = sns.barplot(data=drug_usefulcount_df,x='drug_name',y='counts', palette = 'tab10')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Top Drugs Per UsefulCount', fontsize=20)
sns_1.set_xlabel("Drug Name")
sns_1.set_ylabel("Count")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```





Observation :

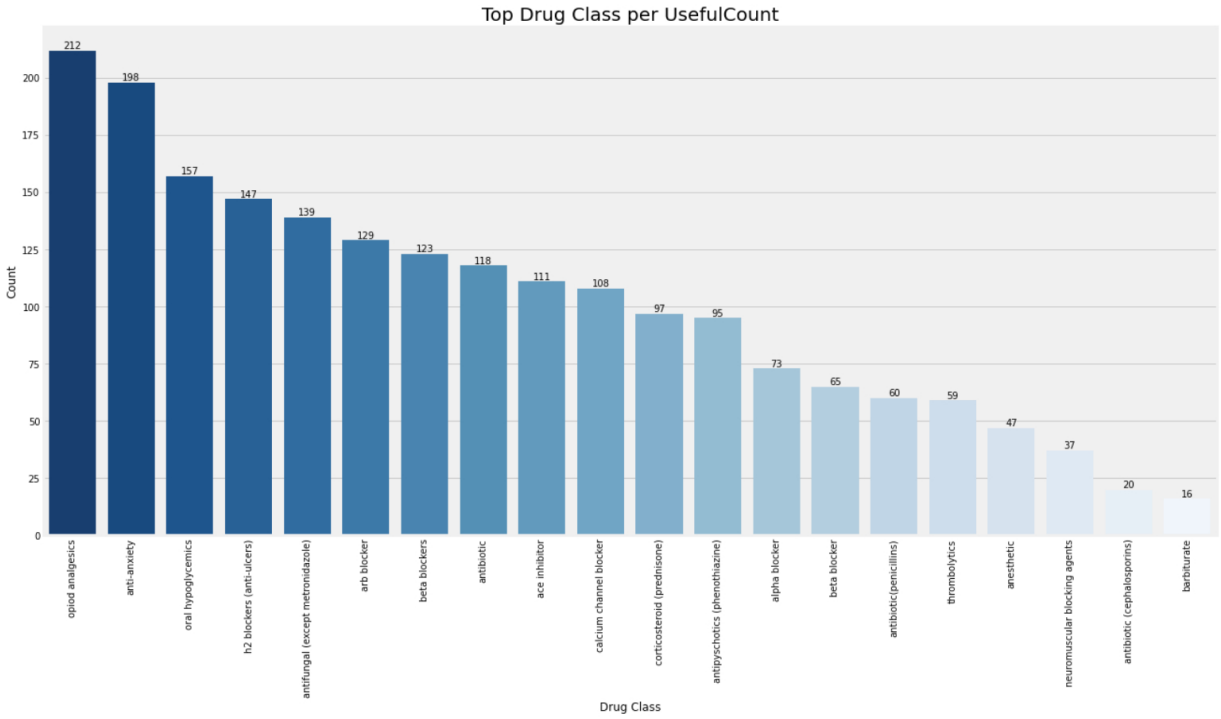
Fluoxetine, Gabapentin and **Bupropion** are the Drugs which are most useful for treating people

-- Top Drug Class per UsefulCount

In [65]:
```
# Top Drugs Class Per UsefulCount
drug_class_usefulcount = df.groupby('drug_class')['usefulCount'].nunique().nlargest(20)
```

In [66]:
```
drug_class_usefulcount_df = pd.DataFrame({'drug_class':drug_class_usefulcount.index,'counts':drug_class_usefulcount.values})
```

In [67]:
```
plt.figure(figsize=(20,10))

sns_1 = sns.barplot(data=drug_class_usefulcount_df,x='drug_class',y='counts', palette = 'Blues_r')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Top Drug Class per UsefulCount', fontsize=20)
sns_1.set_xlabel("Drug Class")
sns_1.set_ylabel("Count")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```



Observations :

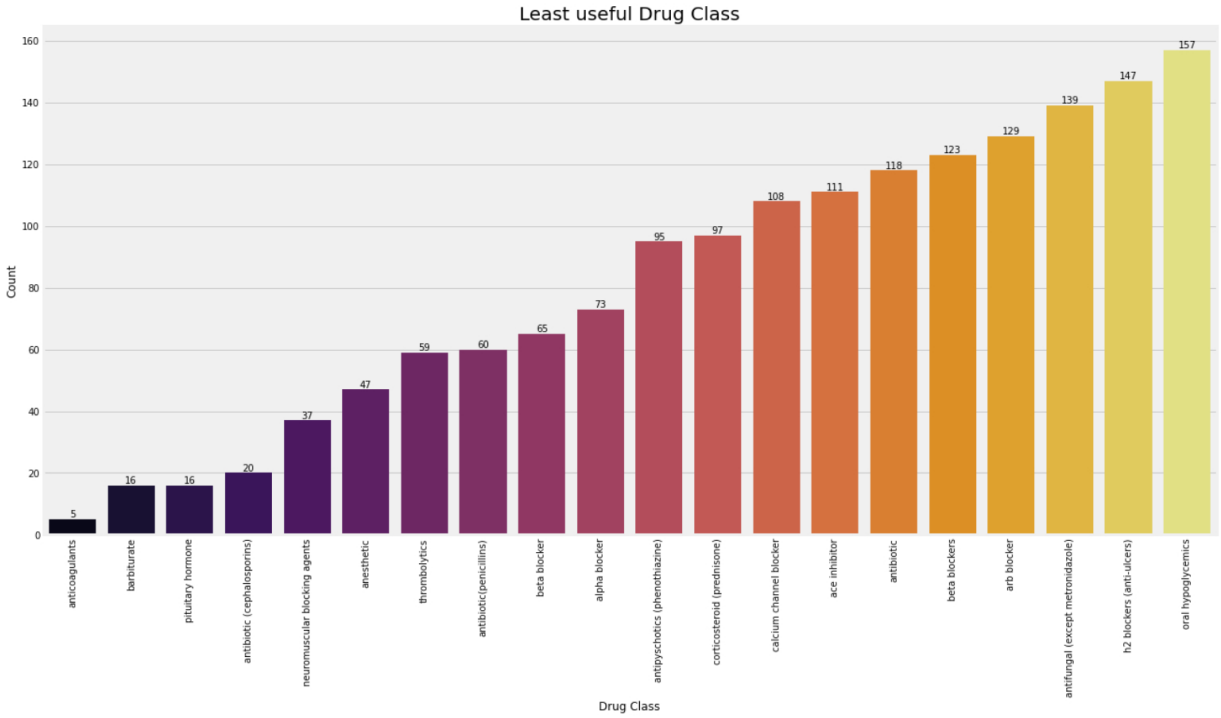
Opiod analgesics, anti-anxiety and **oral hypoglycemics** are the top 3 Drug Classes used to treat the people

-- Least Useful Drug Class

In [68]:
```
# Top Drugs Class Per UsefulCount
least_drug_class_usefulcount = df.groupby('drug_class')['usefulCount'].nunique().nsmallest(20)
```

In [69]:
```
least_drug_class_usefulcount_df = pd.DataFrame({'drug_class':least_drug_class_usefulcount.index,'counts':least_drug_class_usefulc
```

In [70]:
```
plt.figure(figsize=(20,10))

sns_1 = sns.barplot(data=least_drug_class_usefulcount_df,x='drug_class',y='counts', palette = 'inferno')

for i in sns_1.containers:
    sns_1.bar_label(i,)

sns_1.set_title('Least useful Drug Class', fontsize=20)
sns_1.set_xlabel("Drug Class")
sns_1.set_ylabel("Count")
plt.setp(sns_1.get_xticklabels(), rotation=90)
plt.show()
```

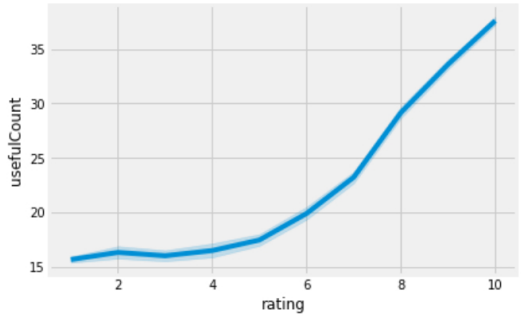


Observation :

Anticoagulant, barbiturate and **pitutary harmone** is the Least used Drug Class for perscribed for peoples

In [71]:
```
### Correlation between Rating and Usefulcount
sns.lineplot(data=df,x='rating',y='usefulCount')
```

Out[71]:
```
<AxesSubplot:xlabel='rating', ylabel='usefulCount'>
```



Observation :

- As the rating goes up the usefulcount goes up