In [266]:
```
import datetime
import os
from datetime import date
import dateutil.relativedelta

import pandas as pd                                    # panda's nickname is

import numpy as np                                     # numpy as np

from pandas import DataFrame, Series                   # for convenience

import matplotlib.pyplot as plt

%matplotlib inline

import pandas as pd
```

In [267]:
```
A=pd.read_csv("issues.csv")
```

In [268]:
```
#Q1 : Are there any useful patterns of outliers in the dataset?

df = A[['created_at','issue_number']]
df['created_at'] = pd.to_datetime(df['created_at'])
df.index = df['created_at']
```

```
C:\Users\mukti\anaconda3\envs\py37\lib\site-packages\ipykernel_launcher.py:
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  after removing the cwd from sys.path.
```

In [269]:
```
B = df.resample('W').count()
```

In [270]:
```
issueclosed = B['issue_number'].sum()
issueclosed
```

Out[270]: 2000

In [271]:
```
avgissueclosed = issueclosed/47
avgissueclosed
```

Out[271]: 42.5531914893617

In [272]:
```
stddevclosed = B['issue_number'].std()
stddevclosed
```

Out[272]: 55.625646504284404

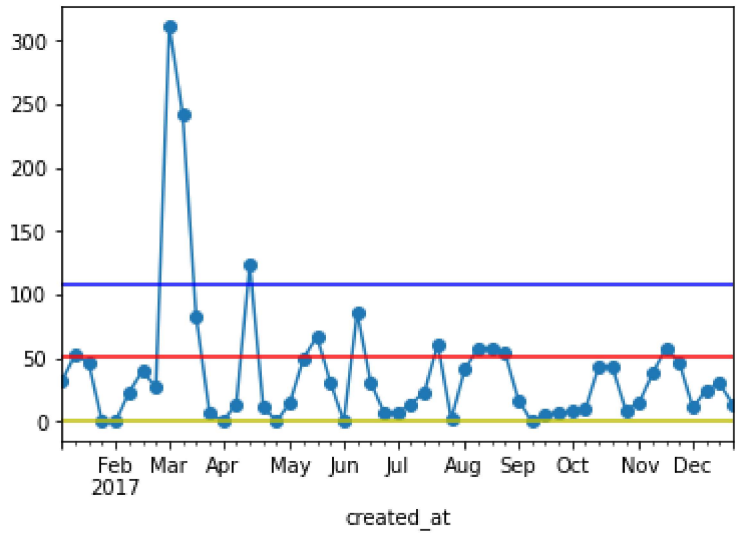
In [273]:
```
Rbar=B["issue_number"].sum()/39  ## mean value of range of each lot
UCL_R= 2.114*Rbar                   ## upper control limit of range
LCL_R= 0*Rbar
```

In [274]:
```
B.insert(2,'Rbar',Rbar)
B.insert(3,'UCL_R',UCL_R)
B.insert(4,'LCL_R',LCL_R)
```

In [275]:
```
B["issue_number"].plot(marker="o")
B["UCL_R"].plot(color='b')
B["LCL_R"].plot(color='y')
B["Rbar"].plot(color='r')

# Ans 1 part 1
```

Out[275]: <AxesSubplot:xlabel='created_at'>



In [276]:
```
A=pd.read_csv("issues.csv")
```

In [277]:
```
df = A[['closed_at','issue_number']]
df['closed_at'] = pd.to_datetime(df['closed_at'])
df.index = df['closed_at']
```

```
C:\Users\mukti\anaconda3\envs\py37\lib\site-packages\ipykernel_launcher.py:
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
```

In [278]:
```
B = df.resample('W').count()
```

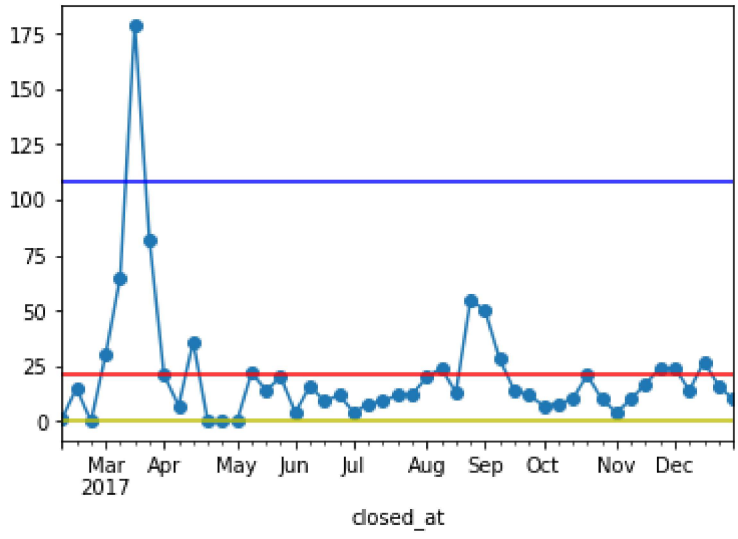
In [279]:
```
Rbar1=B["issue_number"].sum()/47  ## mean value of range of each lot
UCL_R= 2.114*Rbar                   ## upper control limit of range
LCL_R= 0*Rbar
```

In [280]:
```
B.insert(2,'Rbar1',Rbar1)
B.insert(3,'UCL_R1',UCL_R)
B.insert(4,'LCL_R1',LCL_R)
```

In [281]:
```
B["issue_number"].plot(marker="o")
B["UCL_R1"].plot(color='b')
B["LCL_R1"].plot(color='y')
B["Rbar1"].plot(color='r')

# ans 1 part 2
```

Out[281]: <AxesSubplot:xlabel='closed_at'>



In [282]:
```
# Q2 : What are the UCL (Upper Control Limit) and LCL (Lower Control Limit) f
issueclosed = B['issue_number'].sum()
issueclosed
```

Out[282]: 996

In [283]:
```
avgissueclosed = issueclosed/47
avgissueclosed
```

Out[283]: 21.19148936170213

In [284]:
```
stddevclosed = B['issue_number'].std()
stddevclosed
```

Out[284]: 28.82000987981632

In [285]:
```
# Q3 : 3. Can the outliers detect hidden problems in the given dataset?

df=pd.read_csv("issues.csv")
```

In [286]:
```
df['closed_at'] = pd.to_datetime(df['closed_at'])
df['created_at'] = pd.to_datetime(df['created_at'])
```

In [287]:
```

df['days'] = df['closed_at'] - df['created_at']

df1 = df.dropna()
```

In [288]:
```
dfq3 = df1[['issue_number','days']]
dfq3['days'].dt.days
```

Out[288]:
```
2        9
3        9
6        9
7        9
10      19
        ..
1989    17
1990    17
1993    13
1994    13
1995    59
Name: days, Length: 996, dtype: int64
```

In [289]:
```
dfq3.dtypes
```

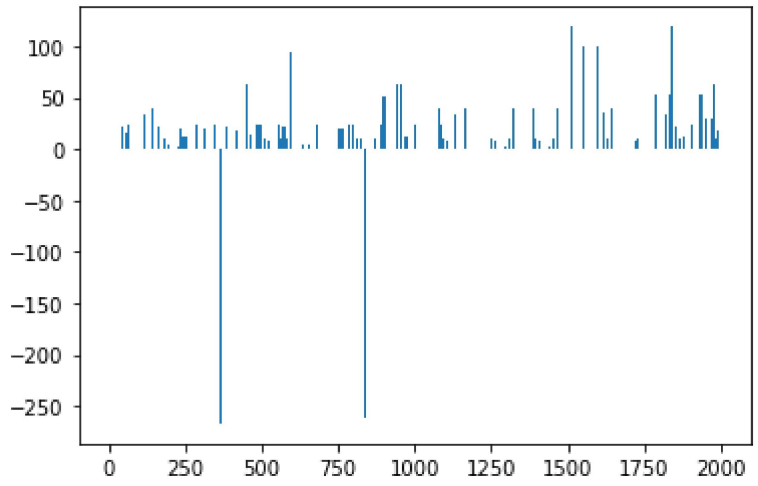
Out[289]:
```
issue_number              int64
days            timedelta64[ns]
dtype: object
```

In [290]:
```
avgof = dfq3['days'].sum()
avgofdays = avgof/996
avgofdays
```

Out[290]:
```
Timedelta('19 days 19:41:12.289156626')
```

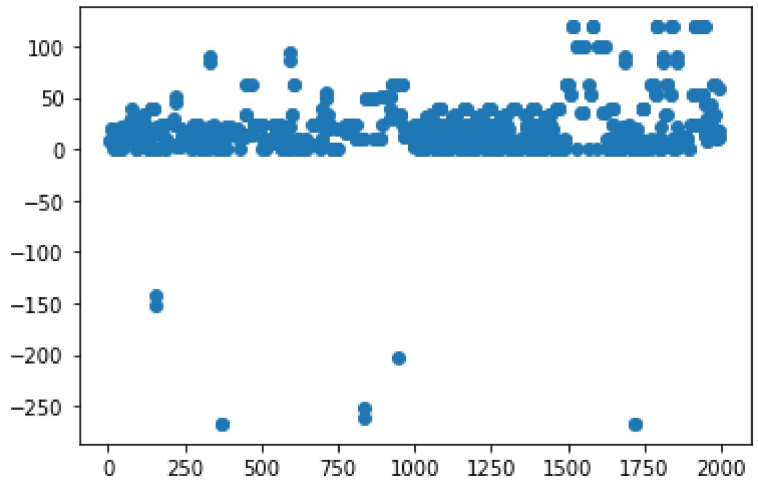
In [291]:
```
x = dfq3['issue_number']
y = dfq3['days'].dt.days
plt.bar(x,y)
```

Out[291]:
```
<BarContainer object of 996 artists>
```



In [292]:
```
plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=Non
```

Out[292]:
```
<matplotlib.collections.PathCollection at 0x1967b8a5148>
```

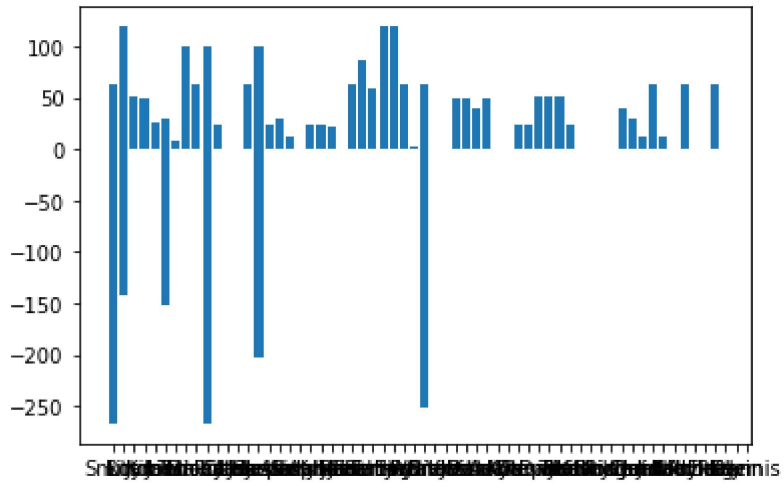


In [293]:
```
dfq4t = df[['Author','days','issue_number','created_at']]
dfq4 = dfq4t
```

In [294]:
```
xq4 = dfq4['Author']
yq4 = dfq4['days'].dt.days

plt.bar(xq4,yq4)
```

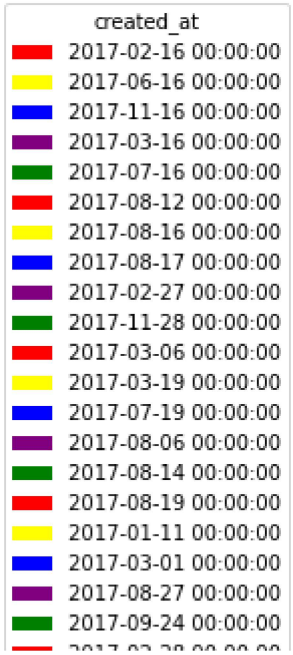
Out[294]: <BarContainer object of 2000 artists>



In [295]:
```
#Q4 . What is the correlation between outliers and hidden problems in the giv

issuesbyday = dfq4.groupby(['Author','created_at']).issue_number.count().head

dateLabelsFig = issuesbyday.unstack().plot(kind='bar',stacked=False,  color=[
```



In [296]:

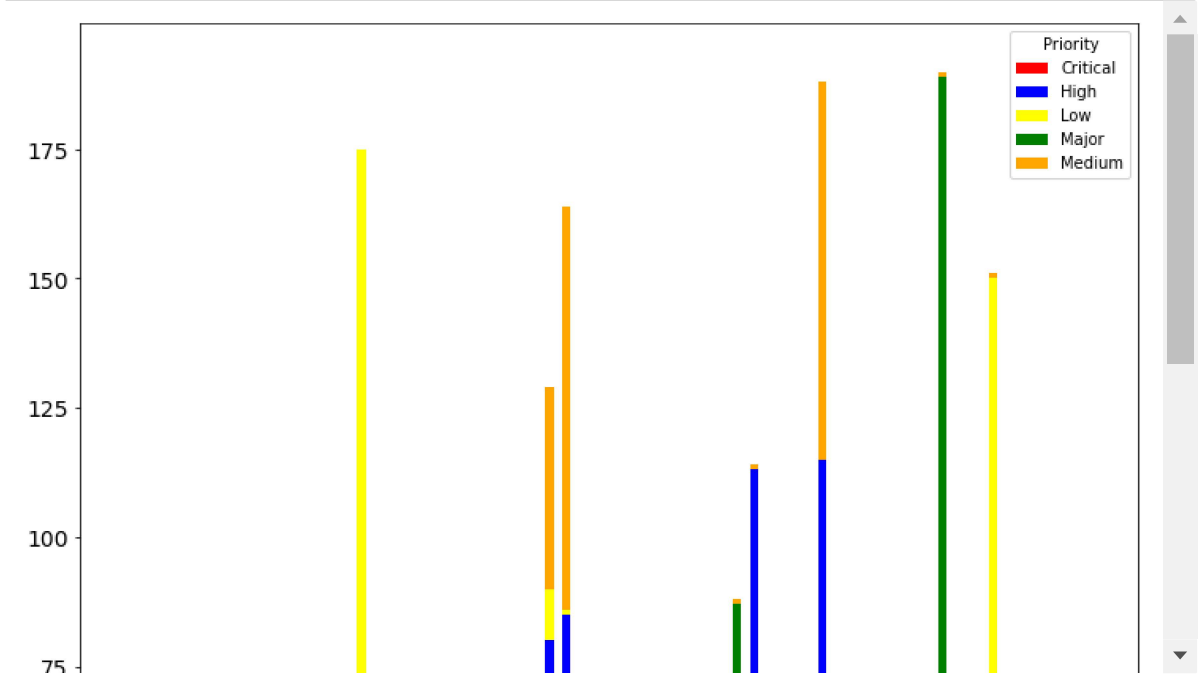
```
# Q5 How to detect if there is problem hidden in the given dataset? - explain

#Q6 How to detect if certain engineer deliberately creates issues with Priori


q5df = df[df.Priority.eq('Critical')]

LabelsReviewedByDate = df.groupby(['Author','Priority']).created_at.count()

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



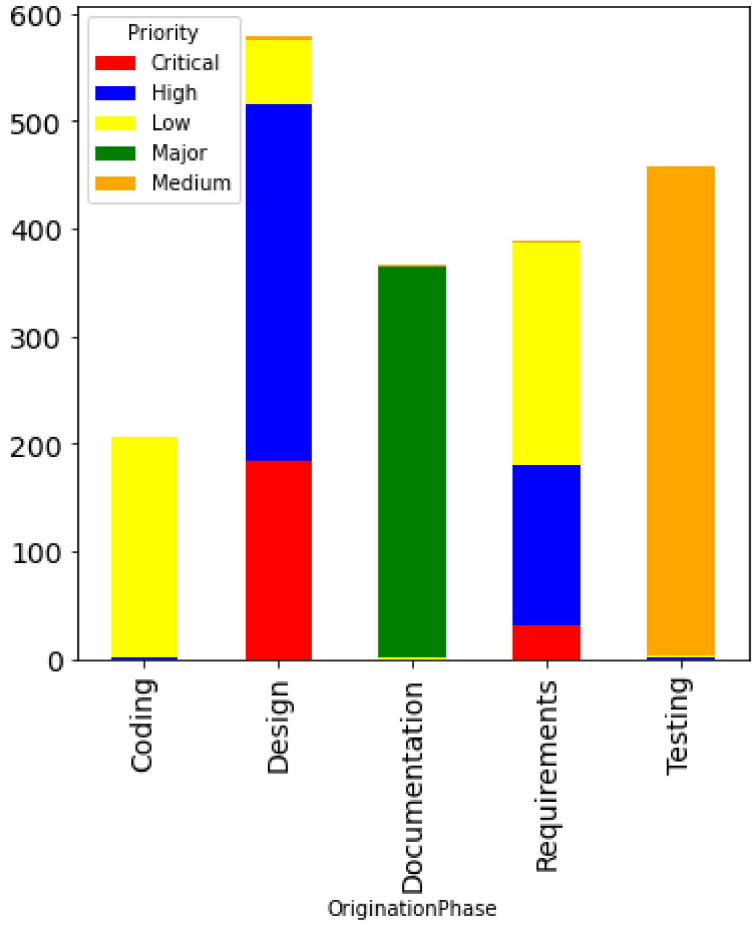
In [297]:

```
# Q7. How to detect if certain origination phase causes majority of the in pr

q6df = df[df.Priority.eq('Critical')]

LabelsReviewedByDate = df.groupby(['OriginationPhase','Priority']).created_at

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



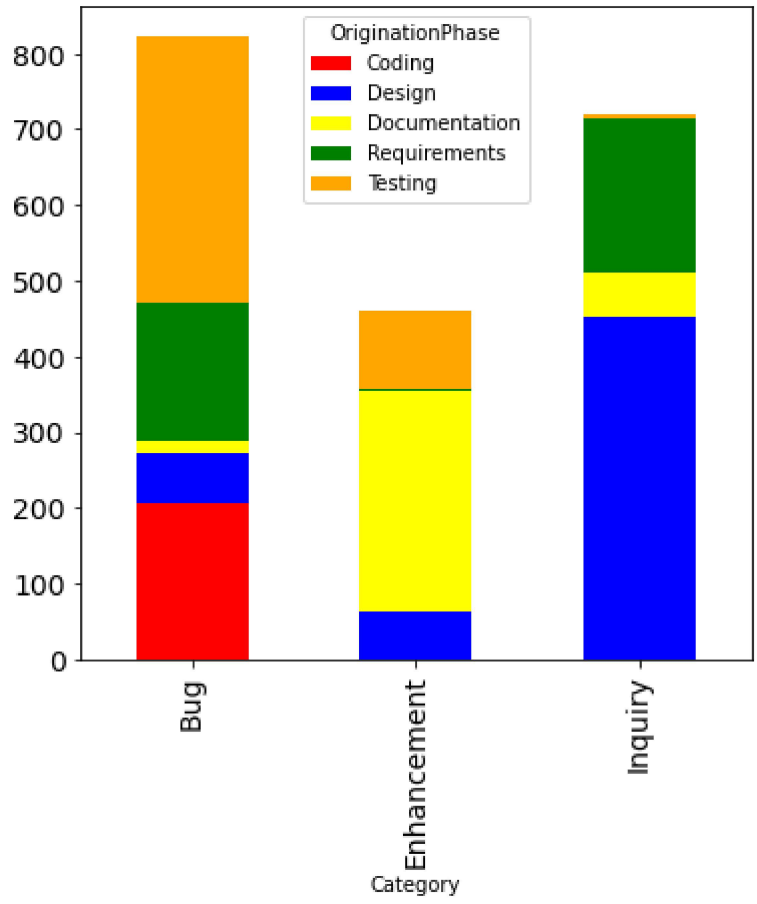
In [298]:
```
# Q8. Can you chart the patterns of outliers in the dataset? - answered in th

# Q9 Can you create the right pivot stacked bar chart? - answered in the pdf

# Q10  How to group multi-levels? Group by Origination phase or Category for

LabelsReviewedByDate = df.groupby(['Category','OriginationPhase']).created_at

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



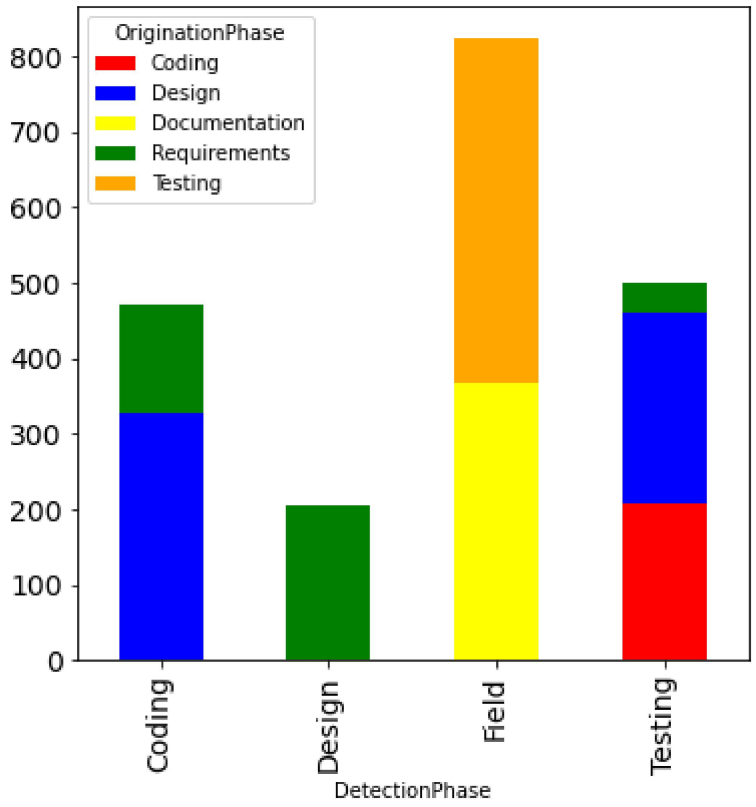
In [299]:

```
# Q11. How many Levels of indexing? Should the Priority be displayed in a piv

# Q12 What is the avg number of issues opened per DetectionPhase?

LabelsReviewedByDate = df.groupby(['DetectionPhase','OriginationPhase']).crea

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



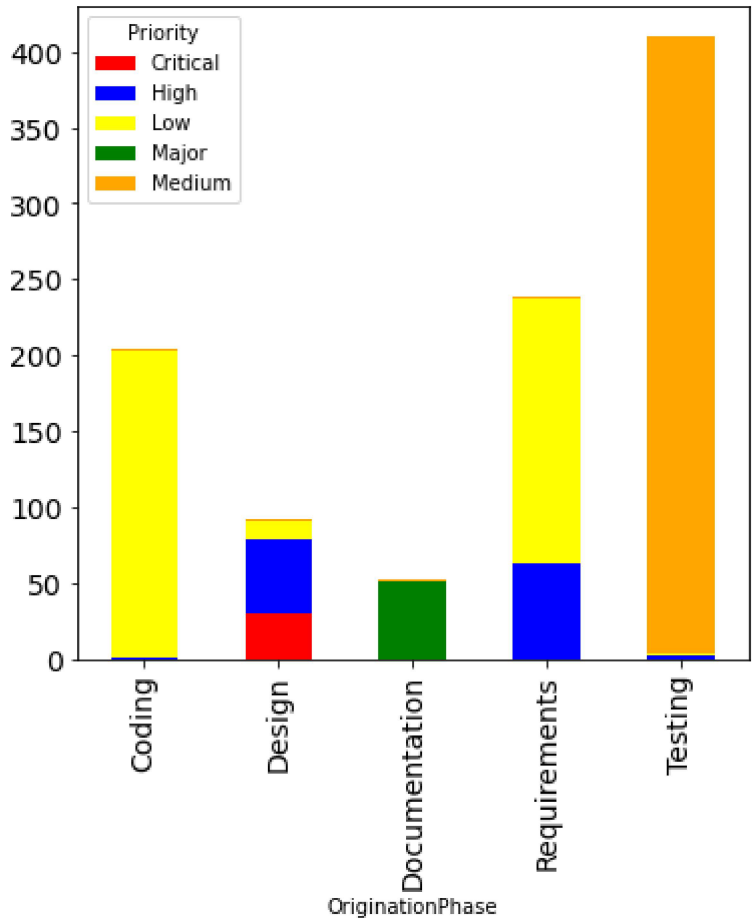
In [300]:

```
# Q13. What is the avg turn around time per issue (time from the day the issu

noissuetesting = df[df.OriginationPhase.eq('Testing')]
noissuetesting

LabelsReviewedByDate = df.groupby(['OriginationPhase','Priority']).days.count

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



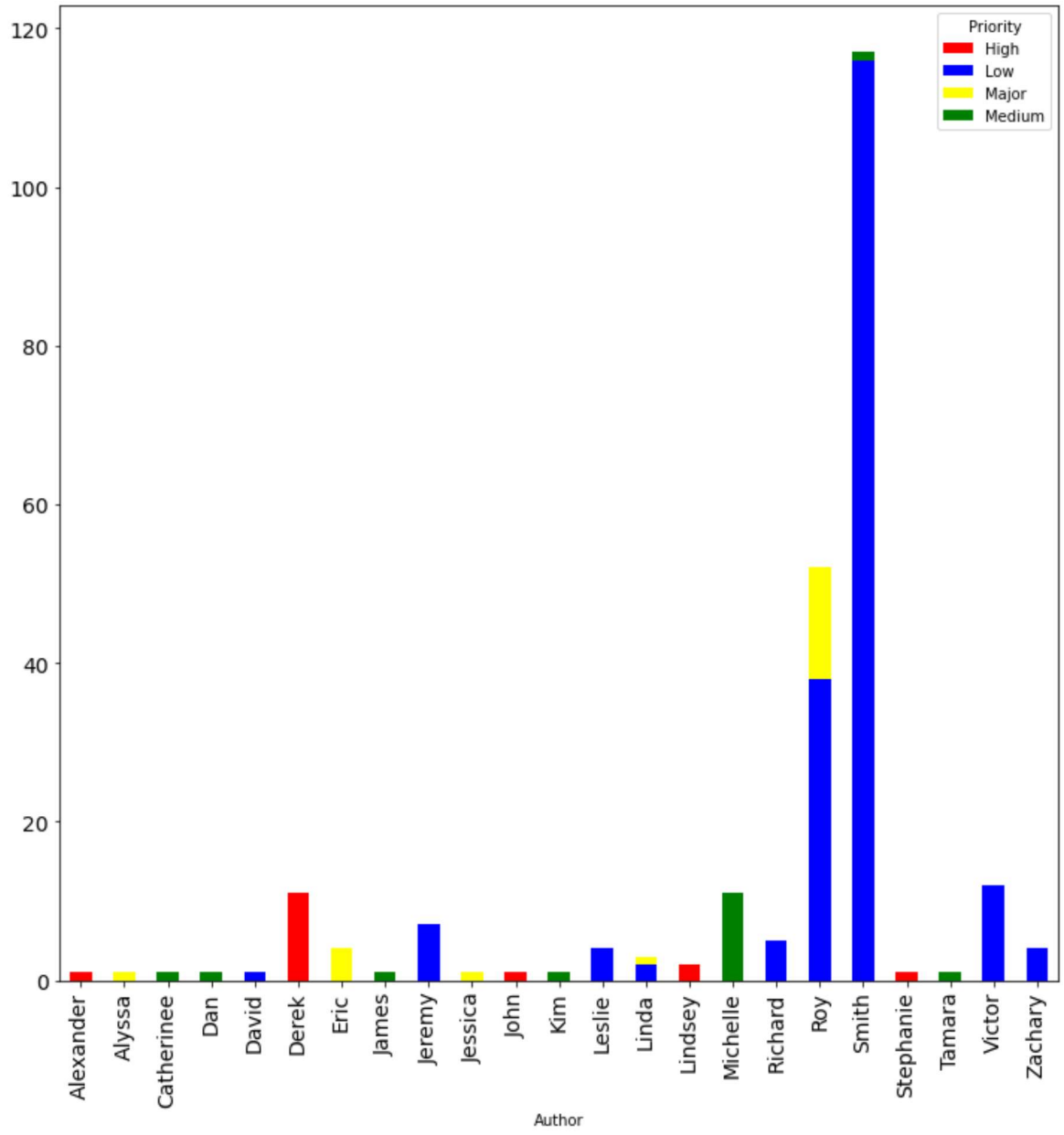
In [301]:

```
# Q14. What is the avg number of rejected issues opened per engineer?

noissuetesting = df[df.Status.eq('Rejected')]

LabelsReviewedByDate = noissuetesting.groupby(['Author','Priority']).created_

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



In [302]:
```
forcoding = df[df.OriginationPhase.eq('Coding')].count()
nodayscoding = forcoding['days']
noissuecoding = forcoding['issue_number']
avgdayscoding = noissuecoding/ nodayscoding
avgdayscoding
```

Out[302]: 1.0147058823529411

In [303]:
```
fordesign = df[df.OriginationPhase.eq('Design')].count()
nodaysdesign = fordesign['days']
noissuedesign = fordesign['issue_number']
avgdaysdesign = noissuedesign / nodaysdesign
avgdaysdesign
```

Out[303]: 6.293478260869565

In [304]:
```
fordoc = df[df.OriginationPhase.eq('Documentation')].count()
nodaysdoc = fordoc['days']
noissuedoc = fordoc['issue_number']
avgdaysdoc = noissuedoc / nodaysdoc
avgdaysdoc
```

Out[304]: 7.0576923076923075

In [305]:
```
forreq = df[df.OriginationPhase.eq('Requirements')].count()
nodaysreq = forreq['days']
noissuereq = forreq['issue_number']
avgdaysreq = noissuereq / nodaysreq
avgdaysreq
```

Out[305]: 1.634453781512605

In [306]:
```
fortesting = df[df.OriginationPhase.eq('Testing')].count()
nodaystesting = fortesting['days']
noissuetesting = fortesting['issue_number']
avgdaystesting = noissuetesting / nodaystesting
avgdaystesting
```

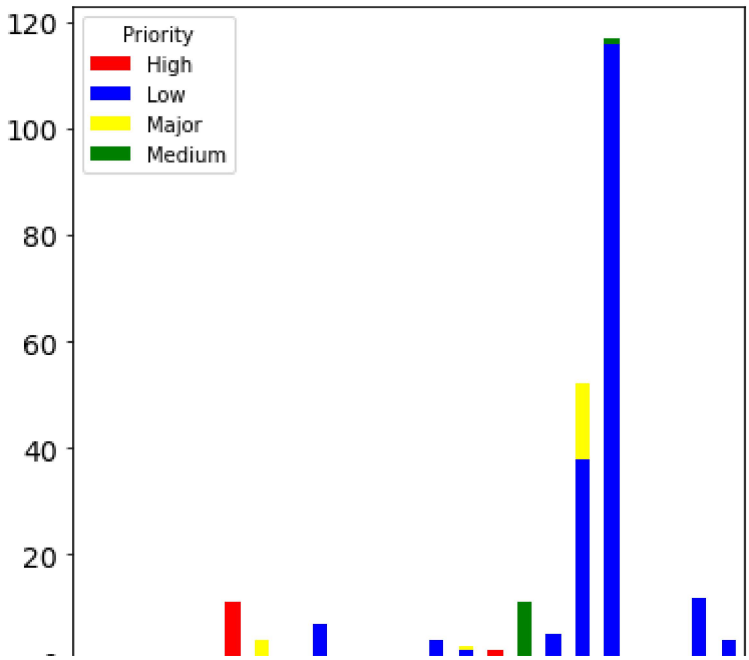
Out[306]: 1.1170731707317074

In [307]:
```
dfra = df[df.Status.eq('Rejected')]
```

In [308]:
```
# 14. What is the avg number of rejected issues opened per engineer?

LabelsReviewedByDate = dfra.groupby(['Author','Priority']).created_at.count()

dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,
```



In [309]:
```
dfra = df[df.Status.eq('Rejected')].count()
dfa = df.Author.value_counts()
authordfa = len(dfa)
rejectedissue = dfra['issue_number']
avgrejissueperauthor = rejectedissue / authordfa
avgrejissueperauthor
```

Out[309]: 3.9193548387096775

In [310]:
```
# Q15 What is the avg number of critical issues opened per engineer?

dfra = df[df.Priority.eq('Critical')].count()
dfa = df.Author.value_counts()
authordfa = len(dfa)
rejectedissue = dfra['issue_number']
avgrejissueperauthor = rejectedissue / authordfa
avgrejissueperauthor
```

Out[310]: 3.467741935483871

In [311]:
```
#Q16 What is the avg number of rejected issues per Origination Phase?

fortesting = df[df.OriginationPhase.eq('Testing')].count()
nodaystesting = fortesting['days']
noissuetesting = fortesting['issue_number']
avgdaystesting = noissuetesting / nodaystesting
avgdaystesting
```

Out[311]: 1.1170731707317074

In [312]:
```
dfra1 = df[df.Status.eq('Rejected')]
dfra = df[df.Status.eq('Rejected')].count()
rejectedissue = dfra['issue_number']
rejectedissue
```

Out[312]: 243

In [313]:
```
# Q17 What is the avg number of critical issues per Origination Phase?

fortesting = dfra1[dfra1.OriginationPhase.eq('Documentation')].count()
fortesting
```

Out[313]:
```
issue_number        21
OriginationPhase    21
DetectionPhase      21
Category            21
Priority            21
Status              21
created_at          21
closed_at           14
Author              21
days                14
dtype: int64
```

In [314]:
```
dfra2 = df[df.Priority.eq('Critical')]
dfra = df[df.Priority.eq('Critical')].count()
rejectedissue = dfra['issue_number']
rejectedissue
```

Out[314]: 215

In [315]:
```
#Q18 What is the avg number of created issues per Origination Phase? - answer

#Q19 What is the avg number of rejected critical issues per Origination Phase

fortesting = dfra2[dfra2.OriginationPhase.eq('Testing')].count()
fortesting
```

Out[315]:
```
issue_number        0
OriginationPhase    0
DetectionPhase      0
Category            0
Priority            0
Status              0
created_at          0
closed_at           0
Author              0
days                0
dtype: int64
```

In [316]:
```
# Q20. What is the ratio of total number of critical to medium issues per Ori

dfra3 = df[df.Priority.eq('Critical')]
dfra3
```

Out[316]:

	issue_number	OriginationPhase	DetectionPhase	Category	Priority	Status	created_a
0	1	Requirements	Coding	Bug	Critical	Approved	2017-02 2
8	9	Design	Testing	Inquiry	Critical	Approved	2017-03 0
18	19	Design	Testing	Inquiry	Critical	Approved	2017-03 0
27	28	Design	Testing	Inquiry	Critical	Approved	2017-08 1
38	39	Design	Testing	Inquiry	Critical	Approved	2017-03 0
...	...	...	...	...	...	...	...
1918	1919	Design	Testing	Bug	Critical	Approved	2017-11-1
1954	1955	Design	Testing	Inquiry	Critical	Approved	2017-12 1
1973	1974	Design	Testing	Inquiry	Critical	Approved	2017-12 1
1975	1976	Design	Testing	Inquiry	Critical	Approved	2017-10 1
1987	1988	Design	Testing	Inquiry	Critical	Approved	2017-05 0

215 rows × 10 columns

In [317]:
```
dfra4 = dfra3[dfra3.Status.eq('Rejected')]
dfra4
```

Out[317]:

	issue_number	OriginationPhase	DetectionPhase	Category	Priority	Status	created_at	clos

In [318]:
```
df5 = df[df.Priority.eq('Critical')]
df6 = df[df.Priority.eq('Medium')]
dfopc = df5[df5.OriginationPhase.eq('Testing')].count()
dfopm = df6[df6.OriginationPhase.eq('Testing')].count()
dfopc
```

Out[318]:
```
issue_number        0
OriginationPhase    0
DetectionPhase      0
Category            0
Priority            0
Status              0
created_at          0
closed_at           0
Author              0
days                0
dtype: int64
```

In [319]:
```
dfopm
```

Out[319]:
```
issue_number        454
OriginationPhase    454
DetectionPhase      454
Category            454
Priority            454
Status              454
created_at          454
closed_at           406
Author              454
days                406
dtype: int64
```

In [320]:
```
df = pd.read_csv("issues.csv")
```

In [321]:
```
# Q21. Which month got the maximum number of Critical issues created?

A = df[df.Priority.eq('Critical')]
```

In [322]:
```
A = df[df.Priority.eq('Critical')]
dfmax = A[['created_at','issue_number']]

dfmax['created_at'] = pd.to_datetime(dfmax['created_at'])
dfmax
# df.index = df['created_at']
```

```
C:\Users\mukti\anaconda3\envs\py37\lib\site-packages\ipykernel_launcher.py:
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  after removing the cwd from sys.path.
```

Out[322]:

	created_at	issue_number
0	2017-02-24	1
8	2017-03-04	9
18	2017-03-06	19
27	2017-08-15	28
38	2017-03-06	39
...	...	...
1918	2017-11-15	1919
1954	2017-12-15	1955
1973	2017-12-15	1974
1975	2017-10-19	1976
1987	2017-05-06	1988

215 rows × 2 columns

In [323]:
```
dfmax.groupby(dfmax['created_at'].dt.strftime('%B'))['issue_number'].count().
```

Out[323]:
```
created_at
September     3
April         4
July          5
December      7
February      8
October      10
June         12
January      13
May          17
November     18
August       30
March        88
Name: issue_number, dtype: int64
```

In [324]:
```
#df['created_at'] = pd.to_datetime(df['created_at']) - pd.to_timedelta(7, uni
#df.groupby([pd.Grouper(key='created_at', freq='W-MON')])['issue_number'].cou
```

In [325]:
```
# Q22. Which week got the minimum number of issues created?

df = df[['created_at','issue_number']]
df['created_at'] = pd.to_datetime(df['created_at'])
df.index = df['created_at']
```

In [326]:
```
B = df.resample('W').count()

B.sort_values(by=['issue_number'])
```

Out[326]:

created_at	created_at	issue_number
2017-04-30	0	0
2017-06-04	0	0
2017-04-02	0	0
2017-01-29	0	0
2017-02-05	0	0
2017-09-10	0	0
2017-07-30	2	2
2017-09-17	6	6
2017-03-26	7	7
2017-06-25	7	7
2017-09-24	7	7
2017-07-02	7	7
2017-10-01	8	8
2017-10-29	9	9
2017-10-08	10	10
2017-12-03	11	11
2017-04-23	12	12
2017-12-24	13	13
2017-04-09	14	14
2017-07-09	14	14
2017-11-05	15	15
2017-05-07	15	15
2017-09-03	17	17
2017-02-12	23	23
2017-07-16	23	23
2017-12-10	25	25
2017-02-26	27	27
2017-12-17	30	30
2017-05-28	31	31
2017-06-18	31	31

created_at	created_at	issue_number
2017-01-08	32	32
2017-11-12	38	38
2017-02-19	40	40
2017-08-06	41	41
2017-10-22	43	43
2017-10-15	44	44
2017-11-26	46	46
2017-01-22	46	46
2017-05-14	50	50
2017-01-15	53	53
2017-08-27	54	54
2017-11-19	58	58
2017-08-20	58	58
2017-08-13	58	58
2017-07-23	61	61
2017-05-21	67	67
2017-03-19	83	83
2017-06-11	86	86
2017-04-16	124	124
2017-03-12	242	242
2017-03-05	312	312

In [ ]: