Plot the Graph for the total number of issues with different labels created and closed on every date on GitHub for SPM587SP19 issues project

```
SPM Assignment: 5
Phase 3
Name: Ritika Kumari
CWID: A20414073
```

In [1]:
```
import os

#import cPickle as pickle

import pandas as pd                                    # panda's nickname is pd

import numpy as np                                     # numpy as np

from pandas import DataFrame, Series                   # for convenience

import matplotlib.pyplot as plt

%matplotlib inline
```

Requirement : Get the total number of issues with different labels for every date and plot them in a stacked chart

In [2]:
```
# Read the JSON file into a list of dictionaries

import json
list_of_issues_dict_data = [json.loads(line) for line in open('SPM587SP19issues.
```

In [3]:
```
# Create the DataFrame object for the list_of_issues_dict_data object

issues_df = DataFrame(list_of_issues_dict_data)
```

In [4]:
```
# Sanity test: print first 10 rows in our DataFrame

issues_df
```

Out[4]:

	Author	State	closed_at	created_at	issue_number	labels
0	DSP19SCM782	open	None	2019-04-21	491	[Category:Bug, DetectionPhase:Field, Originati...
1	DSP19SCM782	open	None	2019-04-21	490	[Category:Bug, DetectionPhase:Design, Originat...
2	DSP19SCM781	open	None	2019-04-21	489	[Category:Enhancement, DetectionPhase:Coding, ...
3	DSP19SCM781	open	None	2019-04-21	488	[Category:Bug, DetectionPhase:Testing, Origina...
4	DSP19SCM781	open	None	2019-04-21	487	[Category:Inquiry, DetectionPhase:Design, Orig...
5	SSP19SCM781	open	None	2019-04-20	486	[Category:Enhancement, DetectionPhase:Design, ...
6	SSP19SCM781	open	None	2019-04-20	485	[Category:Enhancement, DetectionPhase:Field, O...
7	SSP19SCM781	open	None	2019-04-20	484	[Category:Inquiry, DetectionPhase:Field, Origi...
8	SSP19SCM782	open	None	2019-04-20	483	[Category:Inquiry, DetectionPhase:Field, Origi...
9	SSP19SCM782	open	None	2019-04-20	482	[Category:Bug, DetectionPhase:Field, Originati...
10	JSP19SCM03G	closed	2019-04-20	2019-04-20	480	[Category:Enhancement, DetectionPhase:Testing,...
11	ZSP19SCM65P	open	None	2019-04-20	479	[Address:McDonalds 2525 S King Dr Chicago 6061...
12	YSP19SCM26X	closed	2019-04-19	2019-04-18	477	[Category:Inquiry, DetectionPhase:Field, Origi...
13	RSP19SCM781	open	None	2019-04-18	476	[Category:Bug, DetectionPhase:Field, Originati...
14	RSP19SCM781	open	None	2019-04-18	475	[Category:Inquiry, DetectionPhase:Design, Orig...
15	TSP19SCM782	open	None	2019-04-18	474	[Category:Enhancement, DetectionPhase:Design, ...
16	TSP19SCM782	open	None	2019-04-18	473	[Category:Bug, DetectionPhase:Design, Originat...
17	TSP19SCM782	open	None	2019-04-18	472	[Category:Enhancement, DetectionPhase:Testing,...
18	DSP19SCM782	open	None	2019-04-18	471	[Category:Inquiry, DetectionPhase:Design, Orig...
19	DSP19SCM782	open	None	2019-04-18	470	[Category:Bug, DetectionPhase:Testing, Origina...

	Author	State	closed_at	created_at	issue_number	labels
20	TSP19SCM781	open	None	2019-04-18	469	[Category:Enhancement, DetectionPhase:Field, O...
21	TSP19SCM781	open	None	2019-04-18	468	[Category:Bug, DetectionPhase:Field, Originati...
22	TSP19SCM781	open	None	2019-04-18	467	[Category:Inquiry, DetectionPhase:Field, Oriti...
23	YSP19SCM26X	closed	2019-04-18	2019-04-18	465	[Address:2525 S Martin Luther King Dr Chicago ...
24	SSP19SCM782	open	None	2019-04-17	464	[Category:Bug, DetectionPhase:Coding, Originat...
25	SSP19SCM782	open	None	2019-04-17	463	[Category:Inquiry, DetectionPhase:Testing, Ori...
26	SSP19SCM781	open	None	2019-04-17	462	[Category:Enhancement, DetectionPhase:Design, ...
27	SSP19SCM781	open	None	2019-04-17	461	[Category:Inquiry, DetectionPhase:Field, Origi...
28	SSP19SCM781	closed	2019-04-21	2019-04-17	460	[Category:Enhancement, DetectionPhase:Design, ...
29	DSP19SCM782	closed	2019-04-21	2019-04-17	459	[Category:Enhancement, DetectionPhase:Field, O...
...	...	...	...	...	...	...
326	SPM587SP19	closed	2019-04-03	2019-03-31	31	[Category:Bug, DetectionPhase:Design, Originat...
327	SPM587SP19	closed	2019-04-02	2019-03-31	30	[Category:Inquiry, DetectionPhase:Coding, Orit...
328	SPM587SP19	closed	2019-04-07	2019-03-31	29	[Category:Bug, DetectionPhase:Design, Originat...
329	SPM587SP19	closed	2019-04-02	2019-03-31	28	[Category:Enhancement, DetectionPhase:Design, ...
330	SPM587SP19	closed	2019-04-03	2019-03-31	27	[Category:Inquiry, DetectionPhase:Design, Orit...
331	SPM587SP19	closed	2019-03-31	2019-03-31	26	[Category:Enhancement, DetectionPhase:Coding, ...
332	SPM587SP19	closed	2019-03-31	2019-03-31	25	[Category:Inquiry, DetectionPhase:Design, Orig...
333	SPM587SP19	closed	2019-03-31	2019-03-31	24	[Category:Enhancement, DetectionPhase:Coding, ...
334	SPM587SP19	closed	2019-03-31	2019-03-31	23	[Category:Bug, DetectionPhase:Design, Originat...
335	SPM587SP19	closed	2019-03-31	2019-03-31	22	[Category:Inquiry, DetectionPhase:Coding, Orig...
336	SPM587SP19	closed	2019-03-31	2019-03-31	21	[Category:Bug, DetectionPhase:Design, Originat...
337	SPM587SP19	closed	2019-03-31	2019-03-31	20	[Category:Bug, DetectionPhase:Coding, Oritinat...
338	SPM587SP19	closed	2019-03-31	2019-03-31	19	[Category:Enhancement, DetectionPhase:Design, ...

	Author	State	closed_at	created_at	issue_number	labels
339	SPM587SP19	closed	2019-03-31	2019-03-31	18	[Category:Inquiry, DetectionPhase:Coding, Orig...
340	SPM587SP19	closed	2019-03-31	2019-03-31	17	[Category:Bug, DetectionPhase:Field, Originati...
341	SPM587SP19	closed	2019-03-31	2019-03-31	16	[Category:Inquiry, DetectionPhase:Design, Orig...
342	SPM587SP19	closed	2019-03-31	2019-03-31	15	[Category:Enhancement, DetectionPhase:Coding, ...
343	SPM587SP19	closed	2019-03-31	2019-03-31	14	[Category:Inquiry, DetectionPhase:Design, Orig...
344	SPM587SP19	closed	2019-03-31	2019-03-30	13	[Category:Enhancement, DetectionPhase:Design, ...
345	SPM587SP19	closed	2019-03-31	2019-03-30	12	[Category:Enhancement, DetectionPhase:Coding, ...
346	SPM587SP19	closed	2019-03-31	2019-03-30	11	[Category:Enhancement, DetectionPhase:Design, ...
347	SPM587SP19	closed	2019-03-31	2019-03-30	10	[Category:Bug, DetectionPhase:Coding, Originat...
348	SPM587SP19	closed	2019-03-31	2019-03-30	9	[Category:Enhancement, DetectionPhase:Design, ...
349	SPM587SP19	closed	2019-03-31	2019-03-30	8	[Category:Bug, DetectionPhase:Coding, Originat...
350	SPM587SP19	closed	2019-03-31	2019-03-30	7	[Category:Inquiry, DetectionPhase:Design, Orig...
351	SPM587SP19	closed	2019-03-31	2019-03-30	6	[Category:Inquiry, DetectionPhase:Field, Origi...
352	SPM587SP19	closed	2019-03-31	2019-03-30	5	[Category:Inquiry, DetectionPhase:Field, Origi...
353	SPM587SP19	closed	2019-03-31	2019-03-29	4	[Category:Inquiry, DetectionPhase:Field, Origi...
354	SPM587SP19	closed	2019-03-31	2019-03-29	3	[Category:Inquiry, DetectionPhase:Field, Origi...
355	SPM587SP19	closed	2019-03-31	2019-03-24	1	[Category:Inquiry, DetectionPhase:Field, Origi...

356 rows × 6 columns

In [5]:
```
# Prepare and Clean the dataframe object

wrangled_issues_df = issues_df[['Author','State','closed_at','created_at','issue_
wrangled_issues_df.loc[0:len(wrangled_issues_df), 'OriginationPhase']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'DetectionPhase']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Category']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Priority']= np.NaN
wrangled_issues_df.loc[0:len(wrangled_issues_df),'Status']= np.NaN
```

In [6]:
```
wrangled_issues_df
```

Out[6]:

	Author	State	closed_at	created_at	issue_number	labels	Orig
0	DSP19SCM782	open	None	2019-04-21	491	[Category:Bug, DetectionPhase:Field, Originati...	
1	DSP19SCM782	open	None	2019-04-21	490	[Category:Bug, DetectionPhase:Design, Originat...	
2	DSP19SCM781	open	None	2019-04-21	489	[Category:Enhancement, DetectionPhase:Coding, ...	
3	DSP19SCM781	open	None	2019-04-21	488	[Category:Bug, DetectionPhase:Testing, Origina...	
4	DSP19SCM781	open	None	2019-04-21	487	[Category:Inquiry, DetectionPhase:Design, Orig...	

In [7]:
```
for i in range(0, len(wrangled_issues_df)):
   if wrangled_issues_df.iloc[i]['labels']:
        for label in wrangled_issues_df.iloc[i]['labels']:
            label_name= (label.split(':'))[0]
            label_value= (label.split(':'))[1]
            wrangled_issues_df.loc[i, label_name]=label_value
```

In [8]:
```
wrangled_issues_df
```

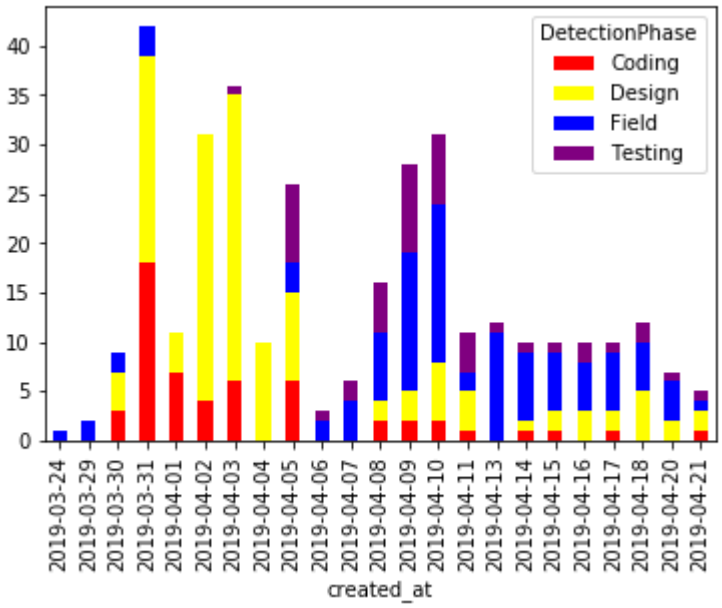
Out[8]:

	Author	State	closed_at	created_at	issue_number	labels	Orig
0	DSP19SCM782	open	None	2019-04-21	491	[Category:Bug, DetectionPhase:Field, Originati...	
1	DSP19SCM782	open	None	2019-04-21	490	[Category:Bug, DetectionPhase:Design, Originat...	
2	DSP19SCM781	open	None	2019-04-21	489	[Category:Enhancement, DetectionPhase:Coding, ...	
3	DSP19SCM781	open	None	2019-04-21	488	[Category:Bug, DetectionPhase:Testing, Origina...	
4	DSP19SCM781	open	None	2019-04-21	487	[Category:Inquiry, DetectionPhase:Design, Orig...	[

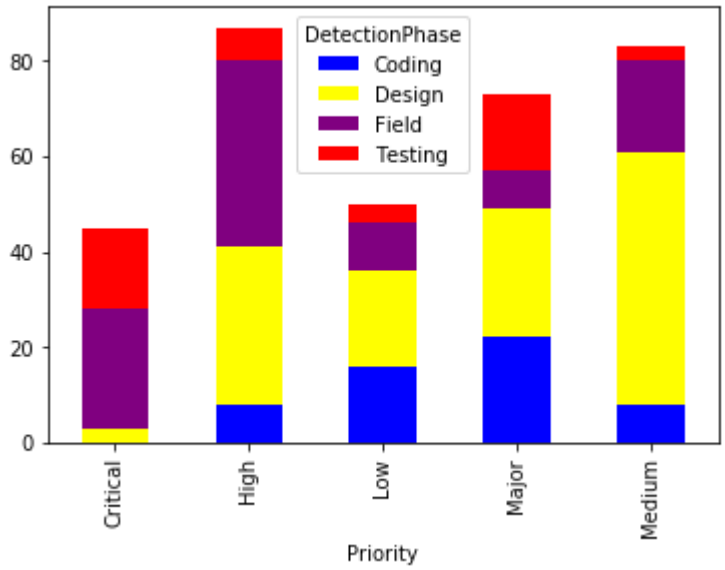
In [9]:
```
wrangled_issues_df['labels'][0]
```

Out[9]:
```
['Category:Bug',
 'DetectionPhase:Field',
 'OriginationPhase:Field',
 'Priority:Major',
 'Status:pendingReview']
```

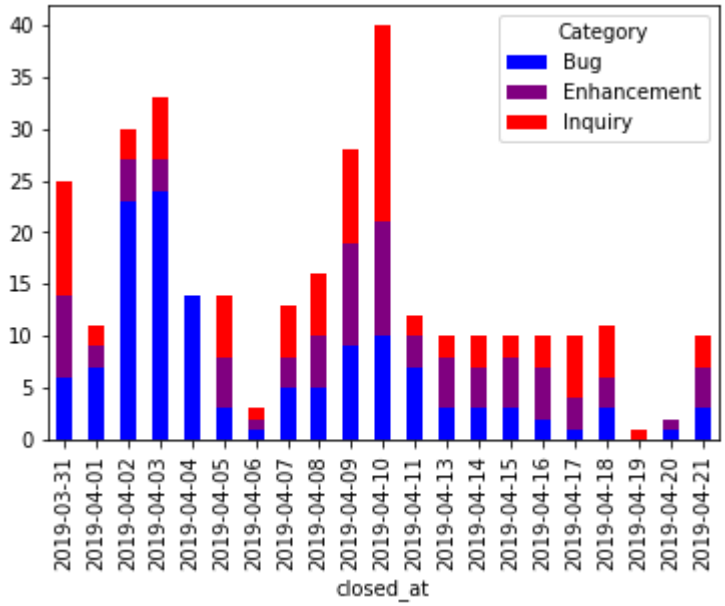
In [10]:
```
# Plot in Bar Chart the total number of issues created every day for every Detaci

LabelsReviewedByDate = wrangled_issues_df.groupby(['created_at','DetectionPhase'

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



In [11]:
```
# Plot in Bar Chart the total number of issues created for every Phase based on t

LabelsReviewedByDate = wrangled_issues_df.groupby(['Priority','DetectionPhase'])

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



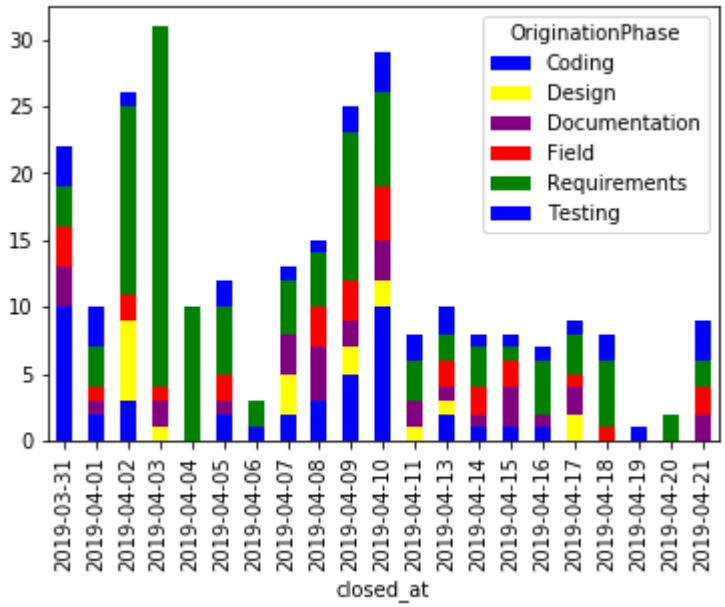
In [12]:
```
# Plot in Bar Chart the total number of issues closed every day for every Categor

LabelsReviewedByDate = wrangled_issues_df.groupby(['closed_at','Category']).close

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



Requirement #1: Plot in Bar Chart the total

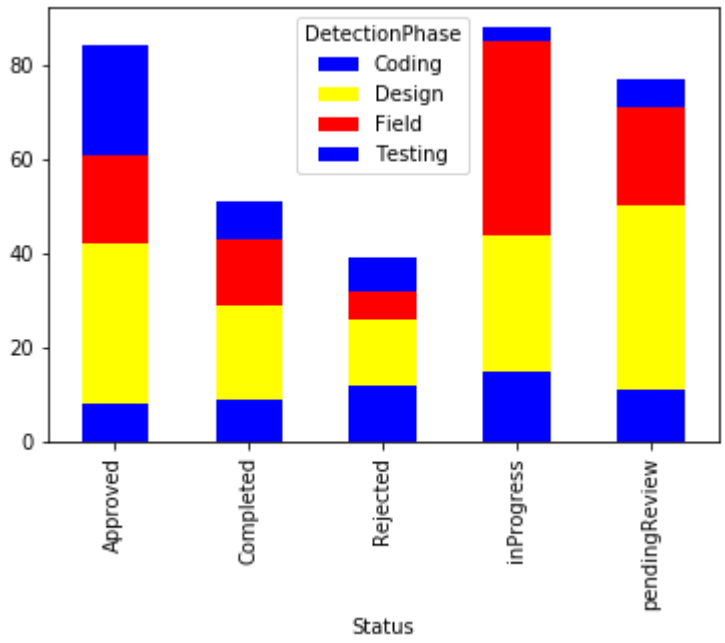
number of issues closed every day for every Origination Phase

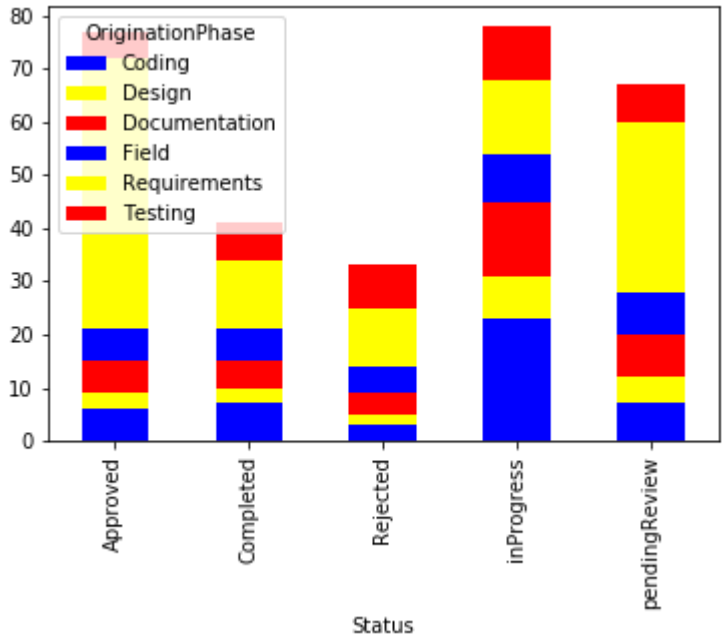
In [13]:
```
# Requirement #1:
# Plot in Bar Chart the total number of issues closed every day for every day for

LabelsReviewedByDate = wrangled_issues_df.groupby(['closed_at','OriginationPhase

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



Requirement #2: # Plot in Bar Chart the total number of issues created for every Phase based on their Status

In [14]:
```
# Requirement #2:
# Plot in Bar Chart the total number of issues created for every Phase based on t
#Is seen in 1st graph
LabelsReviewedByDate = wrangled_issues_df.groupby(['Status','DetectionPhase']).cu
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  col

#Is seen in 2nd graph
LabelsReviewedByDate = wrangled_issues_df.groupby(['Status','OriginationPhase'])
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  col
```

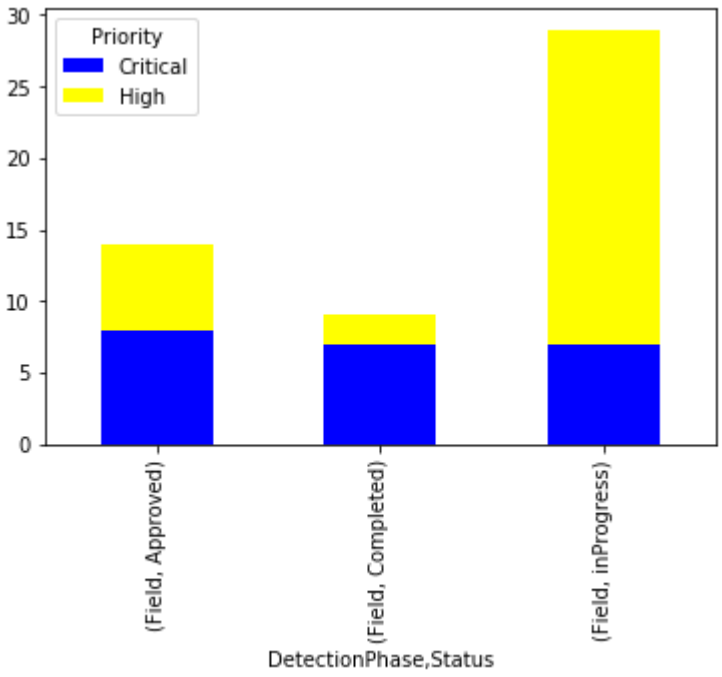


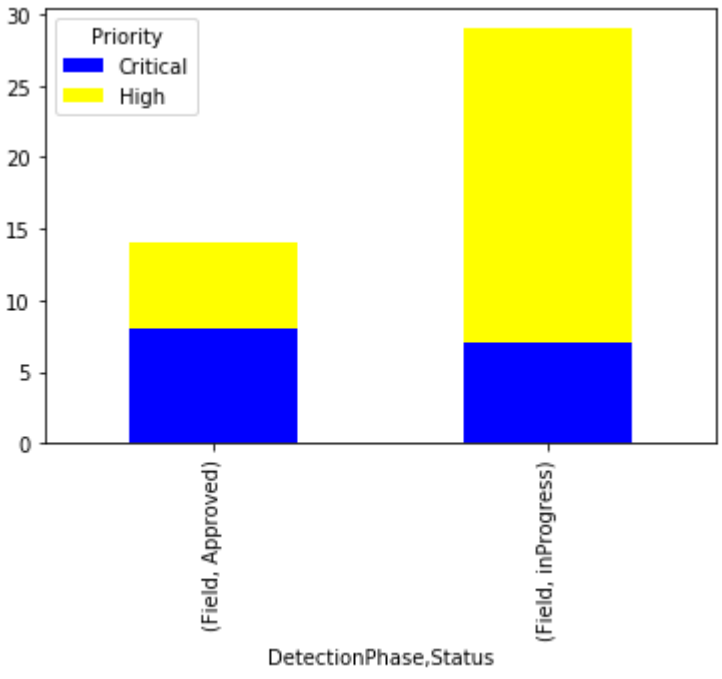


Requirement#3:Plot in Bar Chart the total number of issues for

1) DetectionPhase is Field AND Priority is Critical 2) DetectionPhase is Field AND Status is Completed 3) DetectionPhase is Field AND Priority is Critical AND Status is Approved 4) DetectionPhase is Field AND Priority is Critical or High AND Status is Approved or inProgress

In [15]:
```
filteredData = wrangled_issues_df[wrangled_issues_df.Priority.isin(['Critical','I
filteredData = filteredData[filteredData.Status.isin(['Approved', 'Completed', '
filteredData = filteredData[filteredData.DetectionPhase.isin(['Field'])]
resultGroup = filteredData.groupby(['DetectionPhase', 'Status', 'Priority']).crea
LabelsReviewedByDate = resultGroup
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  co

filteredData = wrangled_issues_df[wrangled_issues_df.Priority.isin(['Critical','I
filteredData = filteredData[filteredData.Status.isin(['Approved','inProgress'])]
filteredData = filteredData[filteredData.DetectionPhase.isin(['Field'])]
resultGroup = filteredData.groupby(['DetectionPhase', 'Status', 'Priority']).crea
LabelsReviewedByDate = resultGroup
dateLabelsFig = LabelsReviewedByDate.unstack().plot(kind='bar',stacked=True,  co
```





Use Facebook/Prophet package (https://facebook.github.io/prophet/docs/quick_sta (https://facebook.github.io/prophet/docs/quick_sta) to forecast the following for our repo

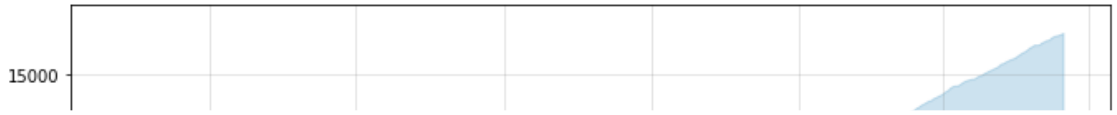
In [16]:
```
wrangled_issues_df.to_csv('SCM587SP19.csv', sep=',', encoding='utf-8', index=Fals
```

In [17]:
```
from fbprophet import Prophet
```

1. The day of the week maximum number of issues created

In [18]:
```
df = pd.read_csv('SCM587SP19.csv')
df.head()
df = df.rename(columns={'created_at':'ds', 'issue_number':'y'})
print(df)
m = Prophet()
m.fit(df)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
fig1 = m.plot(forecast)
fig2 = m.plot_components(forecast)
```

```
344         NaN         NaN         NaN
345         NaN         NaN         NaN
346         NaN         NaN         NaN
347         NaN         NaN         NaN
348         NaN         NaN         NaN
349         NaN         NaN         NaN
350         NaN         NaN         NaN
351         NaN         NaN         NaN
352         NaN         NaN         NaN
353         NaN         NaN         NaN
354         NaN         NaN         NaN
355         NaN         NaN         NaN

[356 rows x 17 columns]
```

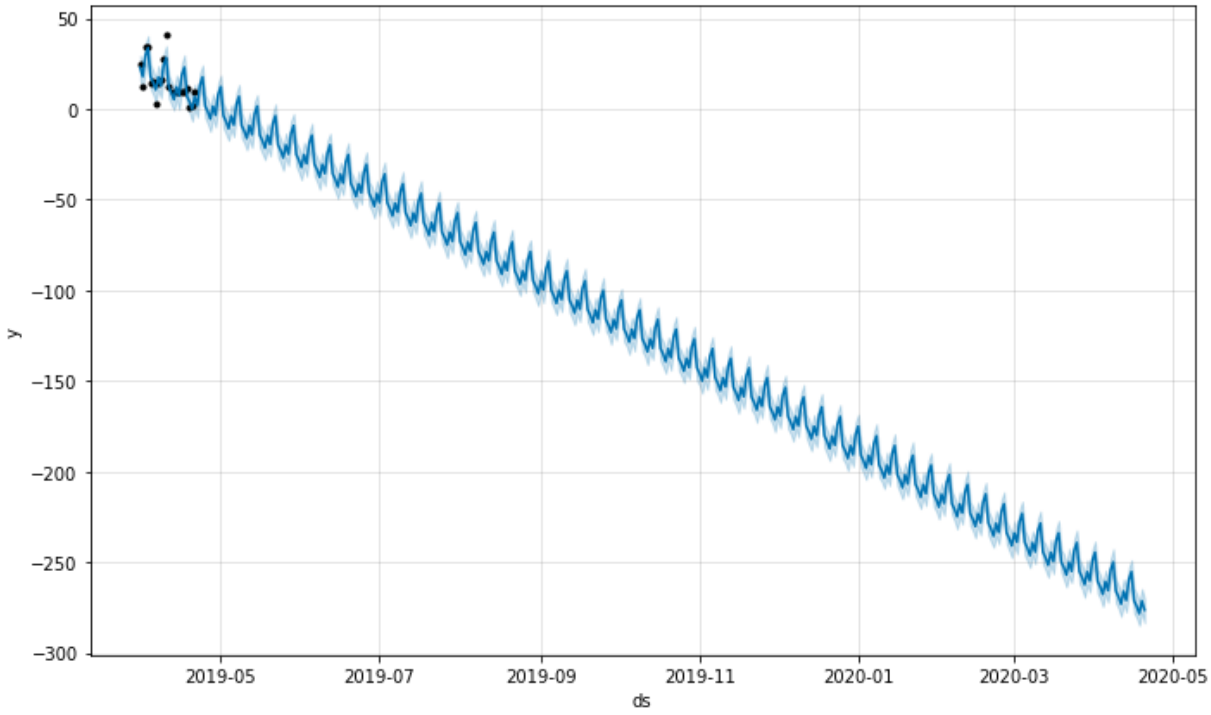


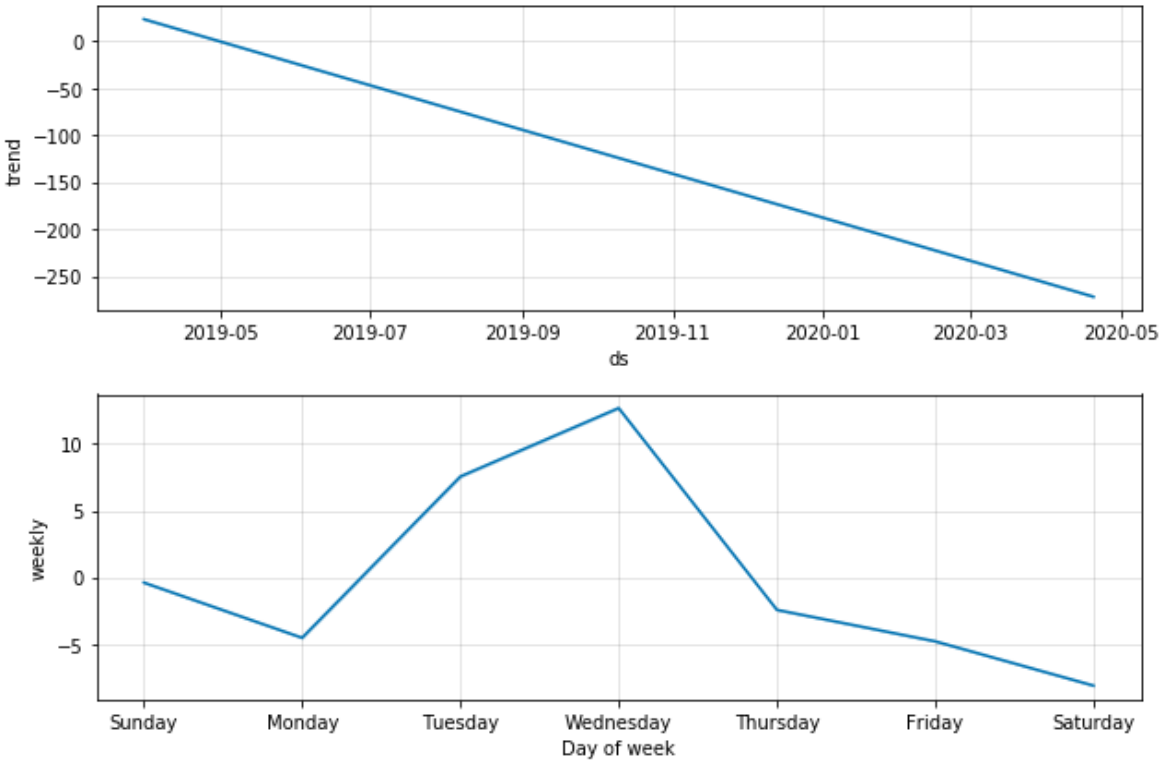
2. The day of the week maximum number of

issues closed

In [19]:
```
df = pd.read_csv('SCM587SP19.csv')
df.head()
df_closed = issues_df['closed_at'].value_counts().rename_axis('ds').reset_index(
m = Prophet()
m.fit(df_closed)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
fig1 = m.plot(forecast)
fig2 = m.plot_components(forecast)
```

```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 15.0.
```



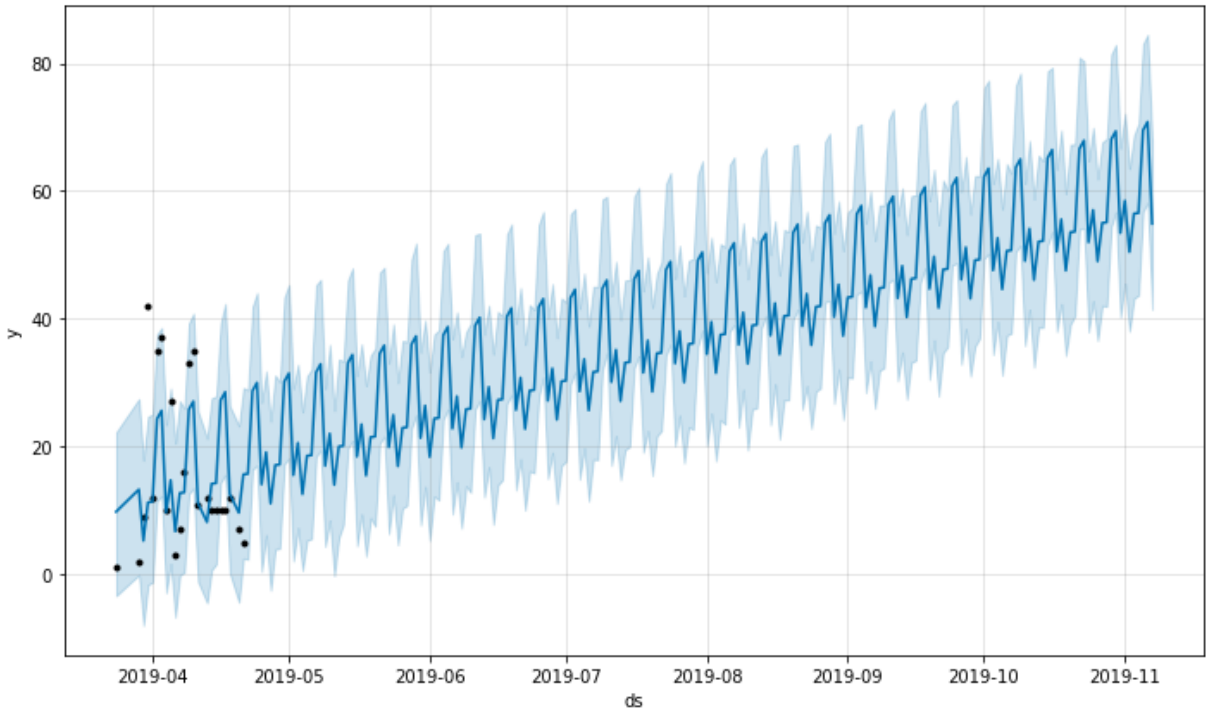


3. Plot the created issues forecast by calling the Prophet.plot method and passing in your forecast dataframe.

In [20]:
```
def predict_plot(file):
    m = Prophet()
    df = pd.read_csv(file)
    df_created = df['created_at'].value_counts().rename_axis('ds').reset_index(na
    m.fit(df_created)
    future = m.make_future_dataframe(periods=200)
    forecast = m.predict(future)
    createdIssueForecast = m.plot(forecast)
```

In [21]:
```
predict_plot('SCM587SP19.csv')
```

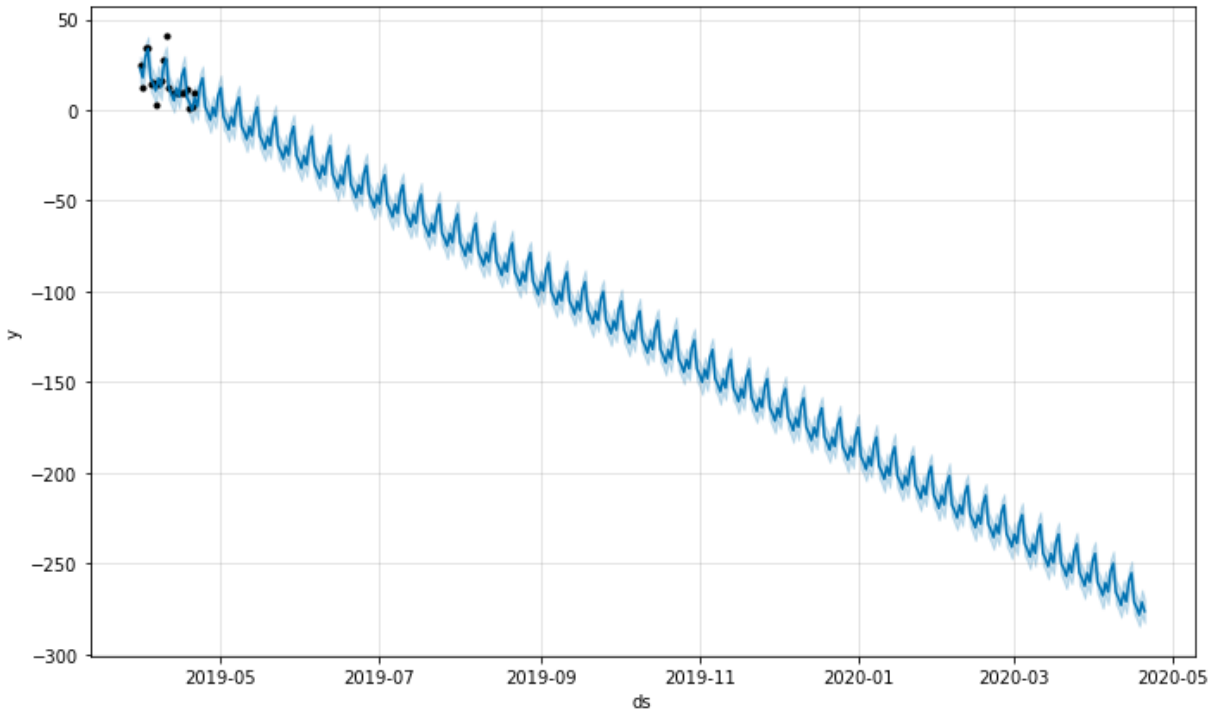
```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 17.0.
```



4. Plot the closed issues forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

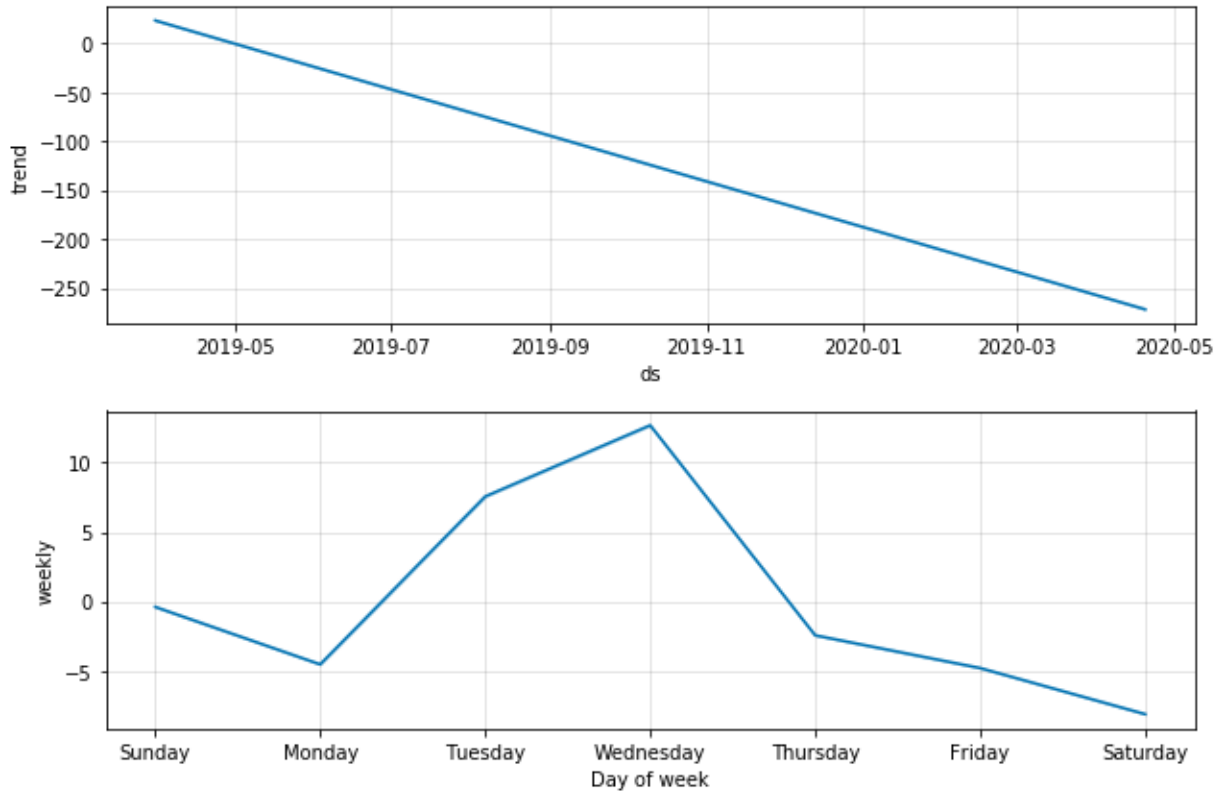
In [22]:
```
m = Prophet()
m.fit(df_closed)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
closedIssueForecast = m.plot(forecast)
```

```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 15.0.
```



5. Plot the pulls forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

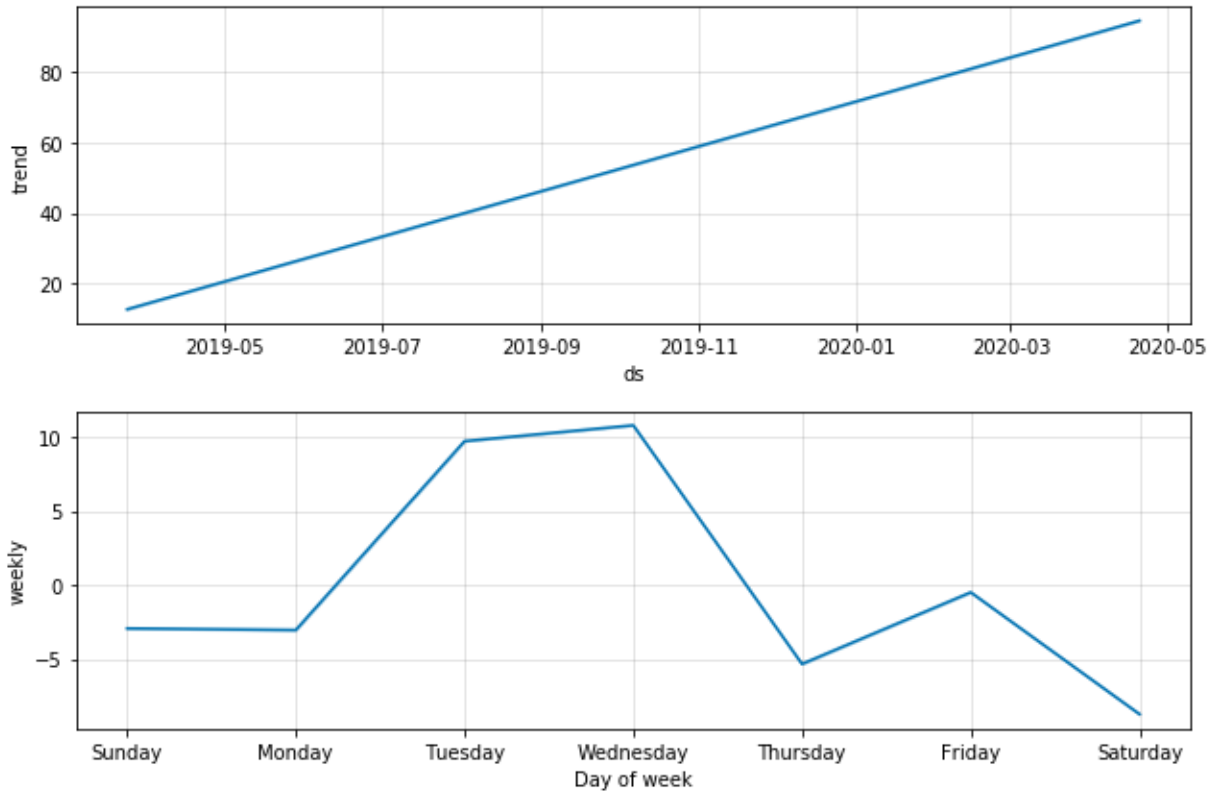
In [23]:
```
pullForecast = m.plot_components(forecast)
```



6. Plot the commits forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

In [24]:
```
df_created = issues_df['created_at'].value_counts().rename_axis('ds').reset_index
m = Prophet()
m.fit(df_created)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
commitForecast = m.plot_components(forecast)
```

```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 17.0.
```



Re-implement the above 6 requirements (listed for Facebook prophet package) using TensorFlow Time Series (TFTS) https://github.com/tensorflow/tensorflow/tree/mas (https://github.com/tensorflow/tensorflow/tree/mas ntrib/timeseries

In [25]:
```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

from os import path
import tempfile
import random
import numpy
import tensorflow as tf
from dateutil import parser
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

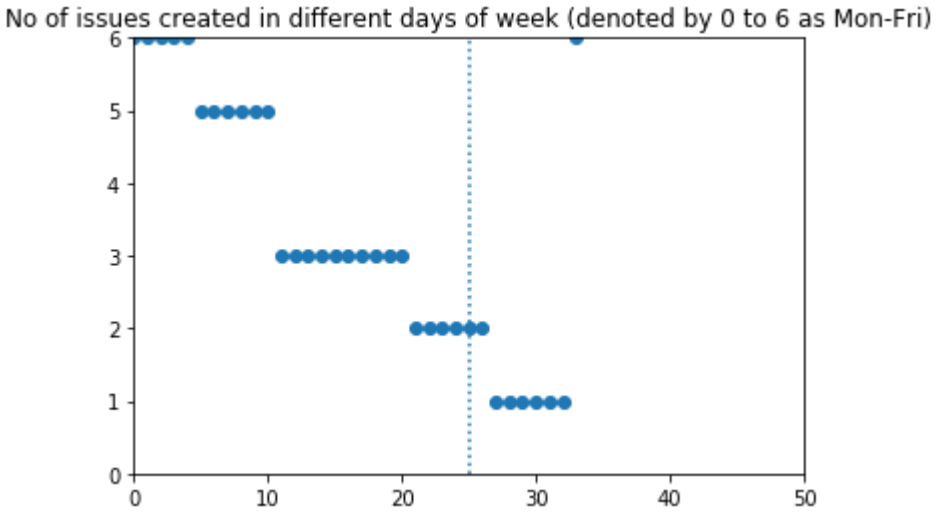
In [26]:
```
try:
  import matplotlib  # pylint: disable=g-import-not-at-top
  matplotlib.use("TkAgg")  # Need Tk for interactive plots.
  from matplotlib import pyplot  # pylint: disable=g-import-not-at-top
  HAS_MATPLOTLIB = True
except ImportError:
  # Plotting requires matplotlib, but the unit test running this code may
  # execute in an environment without it (i.e. matplotlib is not a build
  # dependency). We'd still like to test the TensorFlow-dependent parts of this
  # example, namely train_and_predict.
  HAS_MATPLOTLIB = False

def multivariate_train_and_sample(export_directory=None, training_steps=500):
  """Trains, evaluates, and exports a multivariate model."""
  estimator = tf.contrib.timeseries.StructuralEnsembleRegressor(
      periodicities=[], num_features=1)

  mys = wrangled_issues_df
  openarr = []
  for index, row in wrangled_issues_df.iterrows():
    if(row["State"] == 'open'):
        openarr.append(parser.parse(row['created_at']).weekday())

  openres = pd.DataFrame(openarr)
  openres.to_csv('RKTopen.csv', header=None)
  reader = tf.contrib.timeseries.CSVReader(
      'RKTopen.csv',
      column_names=('times', 'values'))
  train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(
      # Larger window sizes generally produce a better covariance matrix.
      reader, batch_size=4, window_size=64)
  estimator.train(input_fn=train_input_fn, steps=training_steps)
  evaluation_input_fn = tf.contrib.timeseries.WholeDatasetInputFn(reader)
  current_state = estimator.evaluate(input_fn=evaluation_input_fn, steps=1)
  values = [current_state["observed"]]
  times = [current_state[tf.contrib.timeseries.FilteringResults.TIMES]]
  # Export the model so we can do iterative prediction and filtering without
  # reloading model checkpoints.
  if export_directory is None:
    export_directory = tempfile.mkdtemp()
  input_receiver_fn = estimator.build_raw_serving_input_receiver_fn()
  export_location = estimator.export_saved_model(export_directory,
                                                 input_receiver_fn)
  with tf.Graph().as_default():
    numpy.random.seed(1)  # Make the example a bit more deterministic
    with tf.Session() as session:
      signatures = tf.saved_model.loader.load(
          session, [tf.saved_model.tag_constants.SERVING], export_location)
      for _ in range(7):
        current_prediction = (
            tf.contrib.timeseries.saved_model_utils.predict_continuation(
                continue_from=current_state, signatures=signatures,
                session=session, steps=1))
        next_sample = numpy.random.multivariate_normal(
            # Squeeze out the batch and series length dimensions (both 1).
            mean=numpy.squeeze(current_prediction["mean"], axis=(0, 1)),
```

```
            cov=numpy.squeeze(current_prediction["covariance"], axis=(0, 1)))
        # Update model state so that future predictions are conditional on the
        # value we just sampled.
        filtering_features = {
            tf.contrib.timeseries.TrainEvalFeatures.TIMES: current_prediction[
                tf.contrib.timeseries.FilteringResults.TIMES],
            tf.contrib.timeseries.TrainEvalFeatures.VALUES: next_sample[
                None, None, :]}
        current_state = (
            tf.contrib.timeseries.saved_model_utils.filter_continuation(
                continue_from=current_state,
                session=session,
                signatures=signatures,
                features=filtering_features))
        values.append(next_sample[None, None, :])
        times.append(current_state["times"])
  all_observations = numpy.squeeze(numpy.concatenate(values, axis=1), axis=0)
  all_times = numpy.squeeze(numpy.concatenate(times, axis=1), axis=0)
  return all_times, all_observations


if not HAS_MATPLOTLIB:
 raise ImportError("Please install matplotlib to generate a plot from this exampl
all_times, all_observations = multivariate_train_and_sample()
```

```
github.com/tensorflow/tensorboard-plugin-example/blob/master/README.md) for
 more information).
INFO:tensorflow:Summary for np.ndarray is not visible in Tensorboard by defa
ult. Consider using a Tensorboard plugin for visualization (see https://gith
ub.com/tensorflow/tensorboard-plugin-example/blob/master/README.md (https://
github.com/tensorflow/tensorboard-plugin-example/blob/master/README.md) for
 more information).
INFO:tensorflow:Summary for np.ndarray is not visible in Tensorboard by defa
ult. Consider using a Tensorboard plugin for visualization (see https://gith
ub.com/tensorflow/tensorboard-plugin-example/blob/master/README.md (https://
github.com/tensorflow/tensorboard-plugin-example/blob/master/README.md) for
 more information).
INFO:tensorflow:Summary for np.ndarray is not visible in Tensorboard by defa
ult. Consider using a Tensorboard plugin for visualization (see https://gith
ub.com/tensorflow/tensorboard-plugin-example/blob/master/README.md (https://
github.com/tensorflow/tensorboard-plugin-example/blob/master/README.md) for
 more information).
WARNING:tensorflow:Skipping summary for start_tuple, must be a float, np.flo
at32, np.int64, np.int32 or int or np.ndarray or a serialized string of Summ
```

1. The day of the week maximum number of issues created

In [27]:
```
matplotlib.pyplot.ylim(0, 6)
matplotlib.pyplot.xlim(0, 50)
pyplot.axvline(25, linestyle="dotted")
pyplot.title('No of issues created in different days of week (denoted by 0 to 6 a
pyplot.scatter(all_times, all_observations)
pyplot.show()
```



In [28]:

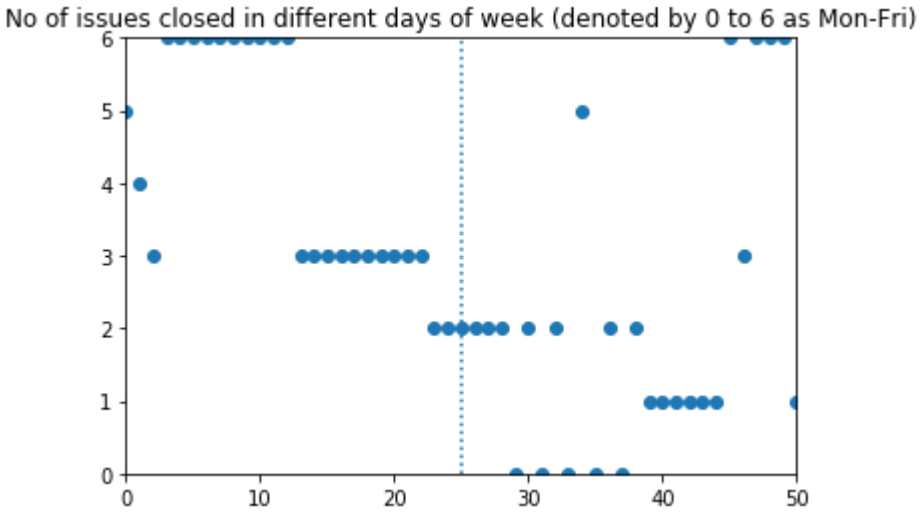
```
try:
  import matplotlib  # pylint: disable=g-import-not-at-top
  matplotlib.use("TkAgg")  # Need Tk for interactive plots.
  from matplotlib import pyplot  # pylint: disable=g-import-not-at-top
  HAS_MATPLOTLIB = True
except ImportError:
  # Plotting requires matplotlib, but the unit test running this code may
  # execute in an environment without it (i.e. matplotlib is not a build
  # dependency). We'd still like to test the TensorFlow-dependent parts of this
  # example, namely train_and_predict.
  HAS_MATPLOTLIB = False

def multivariate_train_and_sample(export_directory=None, training_steps=500):
  """Trains, evaluates, and exports a multivariate model."""
  estimator = tf.contrib.timeseries.StructuralEnsembleRegressor(
      periodicities=[], num_features=1)

  mys = wrangled_issues_df
  closedarr = []
  for index, row in wrangled_issues_df.iterrows():
    if(row["State"] == 'closed'):
        closedarr.append(parser.parse(row['closed_at']).weekday())

  closedres = pd.DataFrame(closedarr)
  closedres.to_csv('RKTclosed.csv', header=None)
  reader = tf.contrib.timeseries.CSVReader(
      'RKTclosed.csv',
      column_names=('times', 'values'))
  train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(
      # Larger window sizes generally produce a better covariance matrix.
      reader, batch_size=4, window_size=64)
  estimator.train(input_fn=train_input_fn, steps=training_steps)
  evaluation_input_fn = tf.contrib.timeseries.WholeDatasetInputFn(reader)
  current_state = estimator.evaluate(input_fn=evaluation_input_fn, steps=1)
  values = [current_state["observed"]]
  times = [current_state[tf.contrib.timeseries.FilteringResults.TIMES]]
  # Export the model so we can do iterative prediction and filtering without
  # reloading model checkpoints.
  if export_directory is None:
    export_directory = tempfile.mkdtemp()
  input_receiver_fn = estimator.build_raw_serving_input_receiver_fn()
  export_location = estimator.export_saved_model(export_directory,
                                                 input_receiver_fn)
  with tf.Graph().as_default():
    numpy.random.seed(1)  # Make the example a bit more deterministic
    with tf.Session() as session:
      signatures = tf.saved_model.loader.load(
          session, [tf.saved_model.tag_constants.SERVING], export_location)
      for _ in range(7):
        current_prediction = (
            tf.contrib.timeseries.saved_model_utils.predict_continuation(
                continue_from=current_state, signatures=signatures,
                session=session, steps=1))
        next_sample = numpy.random.multivariate_normal(
            # Squeeze out the batch and series length dimensions (both 1).
```

```
            mean=numpy.squeeze(current_prediction["mean"], axis=(0, 1)),
            cov=numpy.squeeze(current_prediction["covariance"], axis=(0, 1)))
        # Update model state so that future predictions are conditional on the
        # value we just sampled.
        filtering_features = {
            tf.contrib.timeseries.TrainEvalFeatures.TIMES: current_prediction[
                tf.contrib.timeseries.FilteringResults.TIMES],
            tf.contrib.timeseries.TrainEvalFeatures.VALUES: next_sample[
                None, None, :]}
        current_state = (
            tf.contrib.timeseries.saved_model_utils.filter_continuation(
                continue_from=current_state,
                session=session,
                signatures=signatures,
                features=filtering_features))
        values.append(next_sample[None, None, :])
        times.append(current_state["times"])
  all_observations = numpy.squeeze(numpy.concatenate(values, axis=1), axis=0)
  all_times = numpy.squeeze(numpy.concatenate(times, axis=1), axis=0)
  return all_times, all_observations


if not HAS_MATPLOTLIB:
 raise ImportError("Please install matplotlib to generate a plot from this exampl
all_times, all_observations = multivariate_train_and_sample()
```

```
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]
  [6.]

  [6.]
  [6.]
  [6.]
  [6.]
```

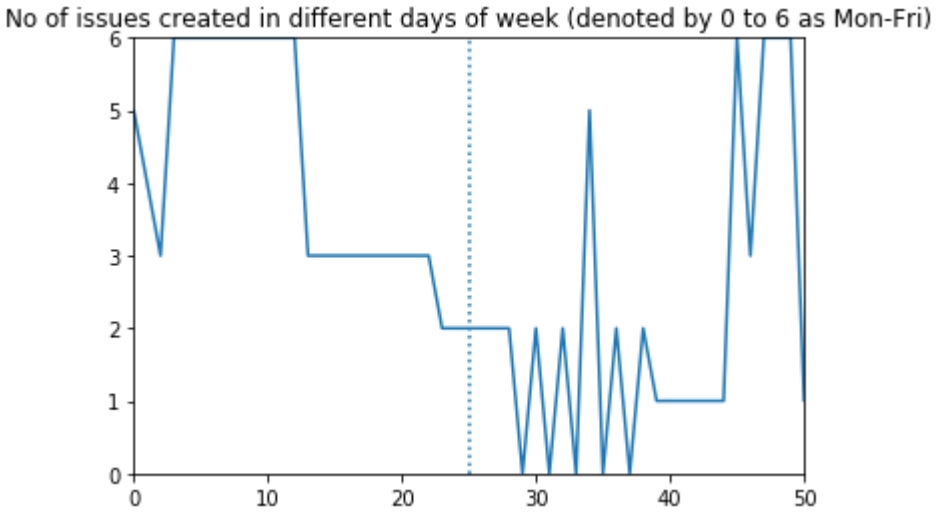
2. The day of the week maximum number of issues closed

In [29]:
```
matplotlib.pyplot.ylim(0, 6)
matplotlib.pyplot.xlim(0, 50)
pyplot.axvline(25, linestyle="dotted")
pyplot.title('No of issues closed in different days of week (denoted by 0 to 6 as
pyplot.scatter(all_times, all_observations)
pyplot.show()
```



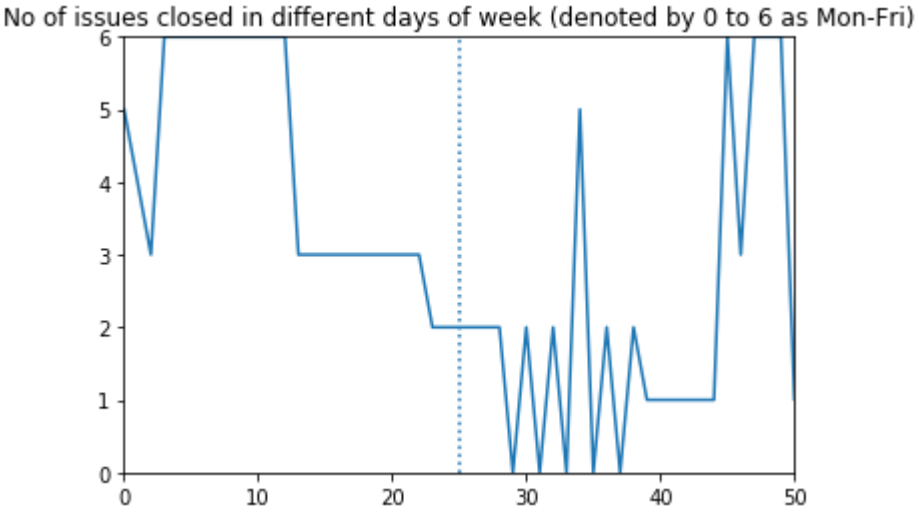
3. Plot the created issues forecast by calling the Prophet.plot method and passing in your forecast dataframe.

In [30]:
```
matplotlib.pyplot.ylim(0, 6)
matplotlib.pyplot.xlim(0, 50)
pyplot.axvline(25, linestyle="dotted")
pyplot.title('No of issues created in different days of week (denoted by 0 to 6 a
pyplot.plot(all_times, all_observations)
pyplot.show()
```



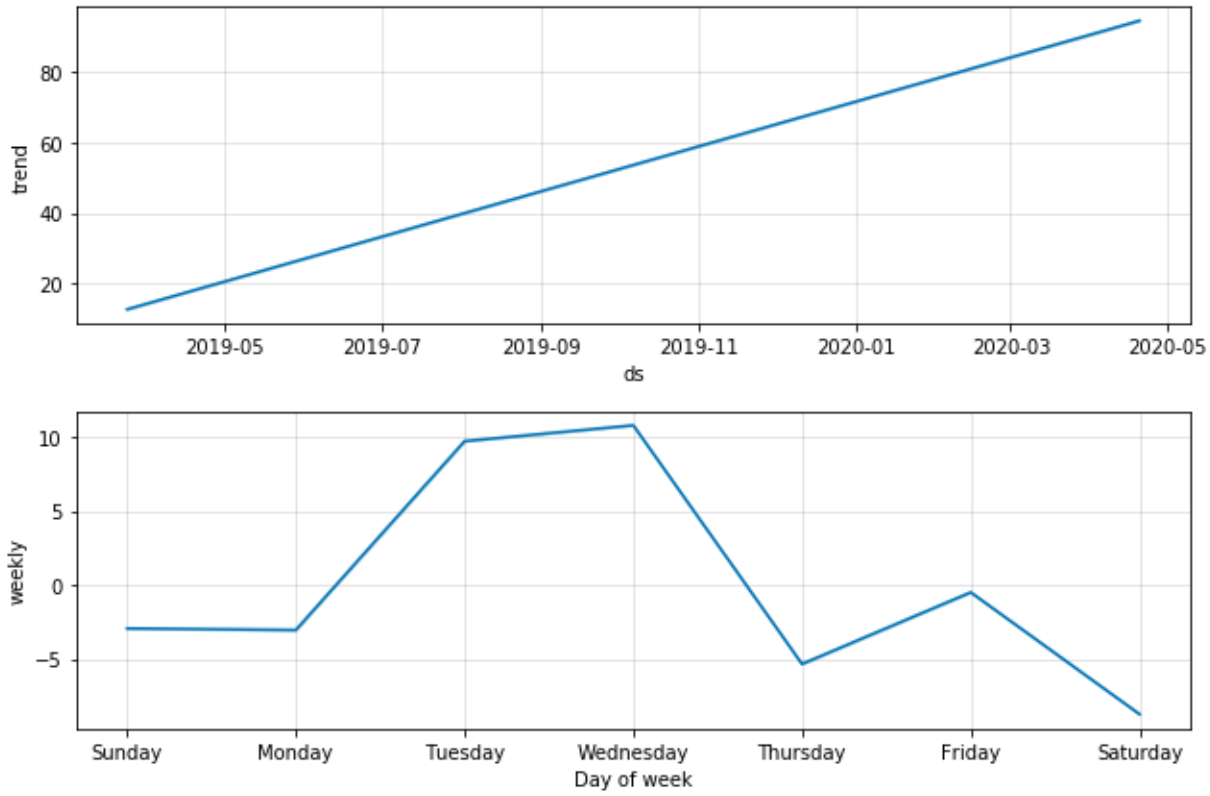
4. Plot the closed issues forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

In [31]:
```
matplotlib.pyplot.ylim(0, 6)
matplotlib.pyplot.xlim(0, 50)
pyplot.axvline(25, linestyle="dotted")
pyplot.title('No of issues closed in different days of week (denoted by 0 to 6 as
pyplot.plot(all_times, all_observations)
pyplot.show()
```



5. Plot the pulls forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

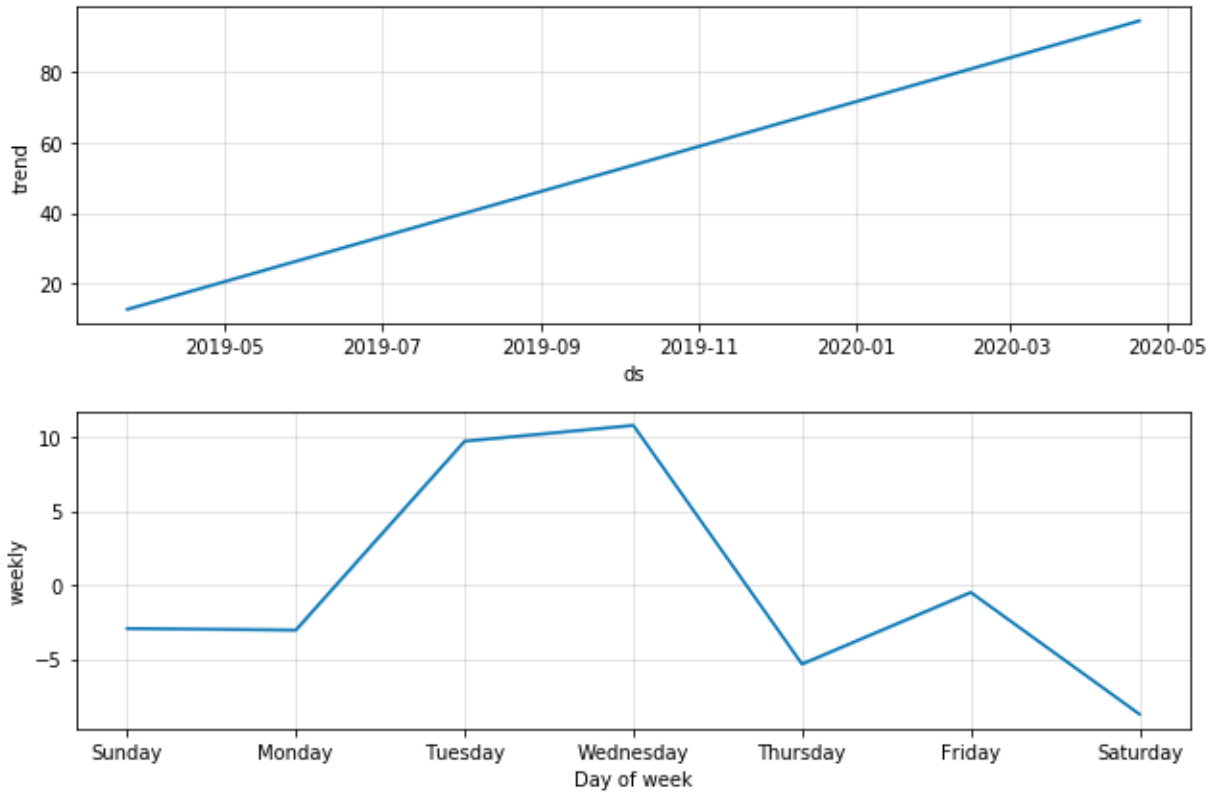
In [32]:
```
pullForecast = m.plot_components(forecast)
```



6. Plot the commits forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

In [33]:
```
df_created = issues_df['created_at'].value_counts().rename_axis('ds').reset_index
m = Prophet()
m.fit(df_created)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
commits_forecast = m.plot_components(forecast)
```

```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 17.0.
```



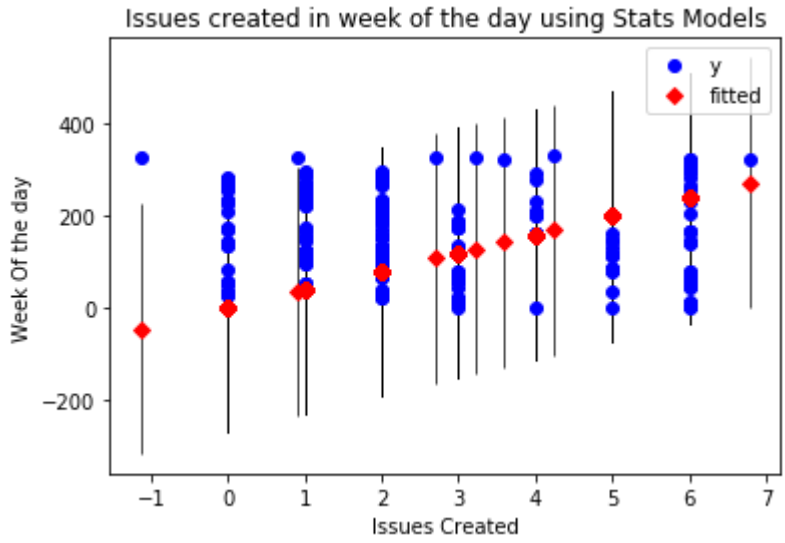
Re-implement the above 6 requirements (listed for Facebook prophet package) using StatsModel : https://www.statsmodels.org/stable/index.html (https://www.statsmodels.org/stable/index.html)

In [34]:
```
import numpy as np
import statsmodels.api as sm
from datetime import datetime
```

1. The day of the week maximum number of issues created

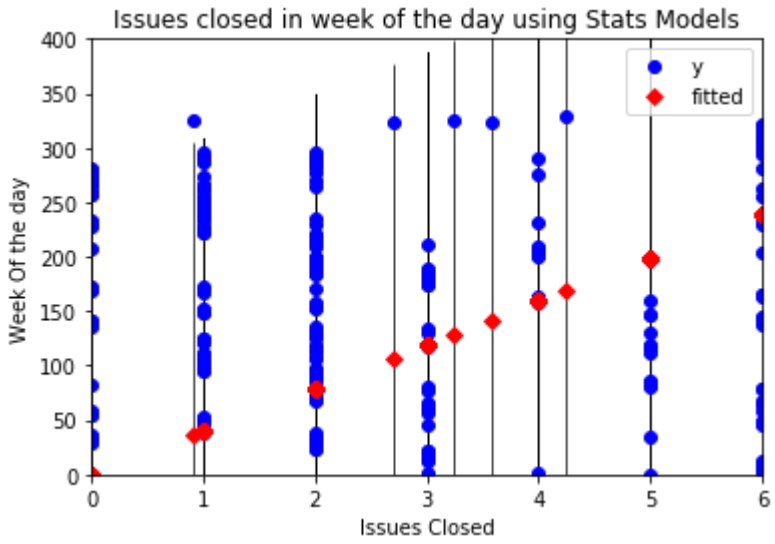
In [35]:
```
model = sm.OLS(all_times, all_observations)
results = model.fit()
fig, ax = plt.subplots()
fig = sm.graphics.plot_fit(results, 0, ax=ax)
ax.set_ylabel("Week Of the day")
ax.set_xlabel("Issues Created")
ax.set_title("Issues created in week of the day using Stats Models")
```

Out[35]:
```
Text(0.5, 1.0, 'Issues created in week of the day using Stats Models')
```



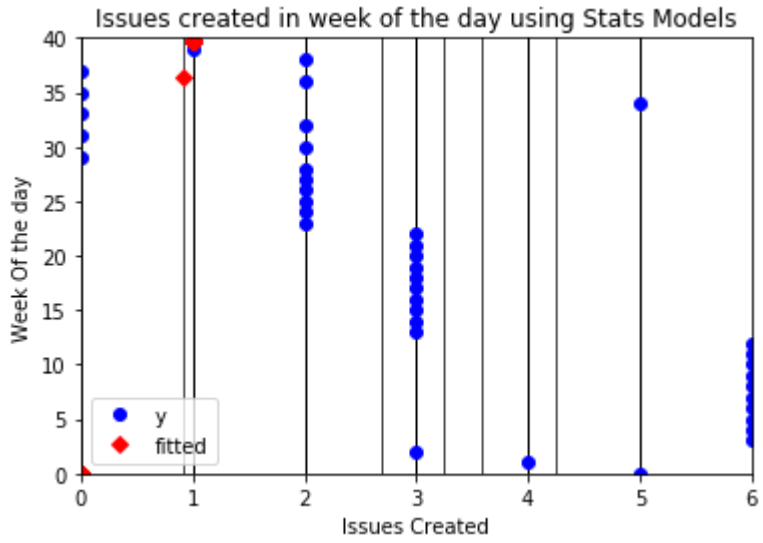
2. The day of the week maximum number of issues closed

In [36]:
```
model = sm.OLS(all_times, all_observations)
results = model.fit()
fig, ax = plt.subplots()
fig = sm.graphics.plot_fit(results, 0, ax=ax)
ax.set_ylabel("Week Of the day")
ax.set_xlabel("Issues Closed")
ax.set_title("Issues closed in week of the day using Stats Models")
matplotlib.pyplot.ylim(0, 400)
matplotlib.pyplot.xlim(0, 6)
plt.show()
```



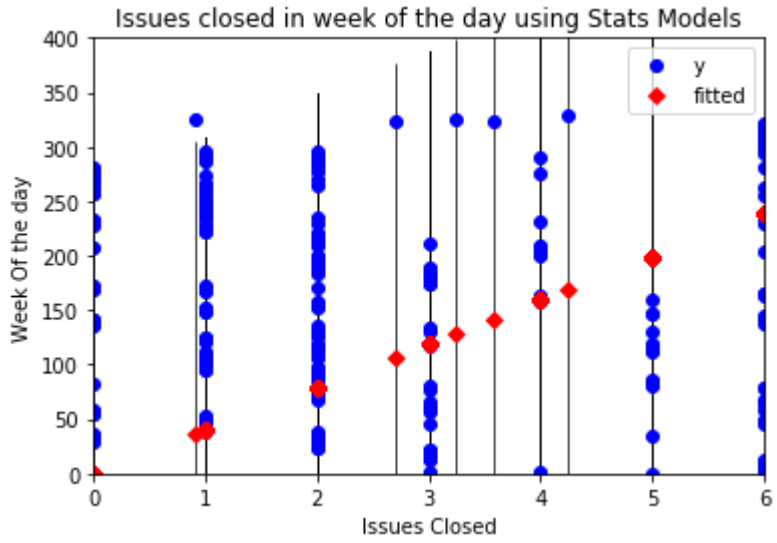
3. Plot the created issues forecast by calling the Prophet.plot method and passing in your forecast dataframe.

In [37]:
```
model = sm.OLS(all_times, all_observations)
results = model.fit()
fig, ax = plt.subplots()
fig = sm.graphics.plot_fit(results, 0, ax=ax)
ax.set_ylabel("Week Of the day")
ax.set_xlabel("Issues Created")
ax.set_title("Issues created in week of the day using Stats Models")
matplotlib.pyplot.ylim(0, 40)
matplotlib.pyplot.xlim(0, 6)
plt.show()
```



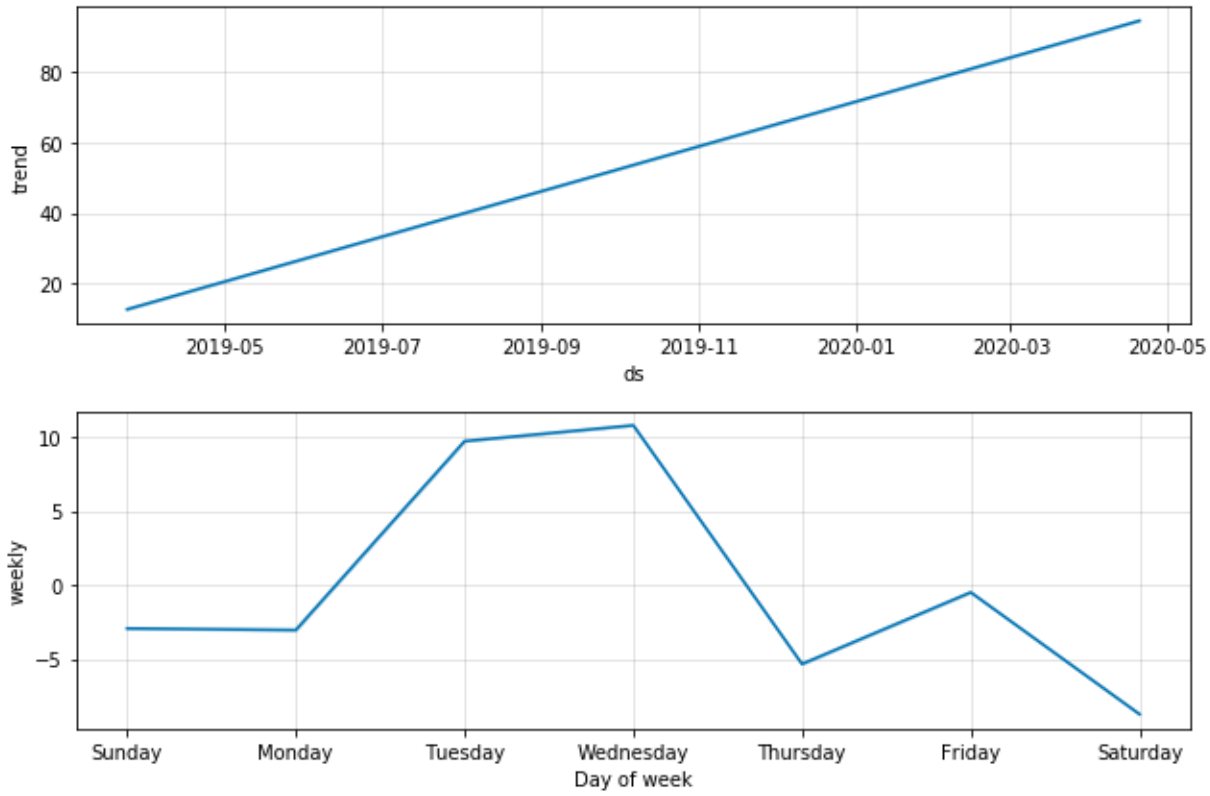
4. Plot the closed issues forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

In [38]:
```
model = sm.OLS(all_times, all_observations)
results = model.fit()
fig, ax = plt.subplots()
fig = sm.graphics.plot_fit(results, 0, ax=ax)
ax.set_ylabel("Week Of the day")
ax.set_xlabel("Issues Closed")
ax.set_title("Issues closed in week of the day using Stats Models")
matplotlib.pyplot.ylim(0, 400)
matplotlib.pyplot.xlim(0, 6)
plt.show()
```



5. Plot the pulls forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

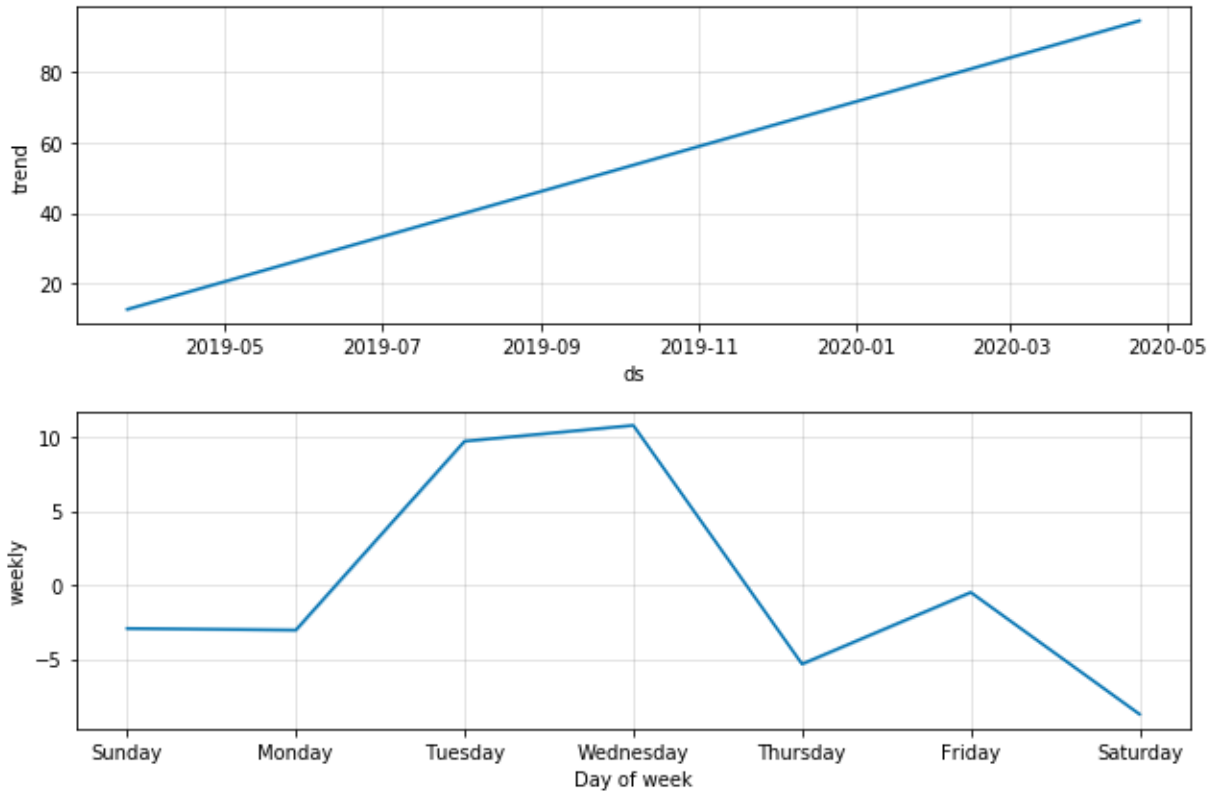
In [39]:
```
pulls_forecast = m.plot_components(forecast)
```



6. Plot the commits forecast; use the Prophet.plot_components method. By default you'll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you'll see those here, too.

In [40]:
```
df_created = issues_df['created_at'].value_counts().rename_axis('ds').reset_index
m = Prophet()
m.fit(df_created)
future = m.make_future_dataframe(periods=365)
future.tail()
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
commits_forecast = m.plot_components(forecast)
```

```
C:\Users\17739\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureW
arning: Series.nonzero() is deprecated and will be removed in a future version.
Use Series.to_numpy().nonzero() instead
  min_dt = dt.iloc[dt.nonzero()[0]].min()
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonalit
y=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=
True to override this.
INFO:fbprophet:n_changepoints greater than number of observations.Using 17.0.
```



In [ ]:
```
```