Before you turn in the homework, make sure everything runs as expected. To do so, select **Kernel→ Restart & Run All** in the toolbar above. Remember to submit both on **DataHub** and **Gradescope**.

Please fill in your name and include a list of your collaborators below.

In [1]:
```
NAME = "Connor McCormick"
COLLABORATORS = "Hannah Grossman"
```

Project 2: NYC Taxi Rides

Part 2: EDA, Visualization, Feature Engineering

In this part, we will conduct EDA on the NYC Taxi dataset that we cleaned and train/validation split in part 1. We will also guide you through the engineering of some features that hopefully will help our model to accurately understand the data.

Imports

Let us start by loading the Python libraries and custom tools we will use in this part.

In [2]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
import os
from pathlib import Path

plt.rcParams['figure.figsize'] = (12, 9)
plt.rcParams['font.size'] = 12

sns.set(style="whitegrid", palette="muted")
%matplotlib inline
```

Loading & Formatting data

The following code loads the data into a pandas DataFrame.

In [3]:
```
# Run this cell to load the data.
data_file = Path("data/part1", "cleaned_data.hdf")
train_df = pd.read_hdf(data_file, "train")
```

In [4]:
```
train_df.head()
```

Out[4]:

	record_id	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_co
16434	8614300	2	2016-01-21 17:37:12	2016-01-21 18:37:56	2
21929	7230200	2	2016-01-29 23:22:26	2016-01-29 23:31:23	2
3370	9830300	2	2016-01-05 18:50:16	2016-01-05 18:56:00	2
21975	7251500	2	2016-01-30 00:14:34	2016-01-30 00:47:13	1
13758	6168000	1	2016-01-18 13:25:24	2016-01-18 13:38:51	1

5 rows × 21 columns

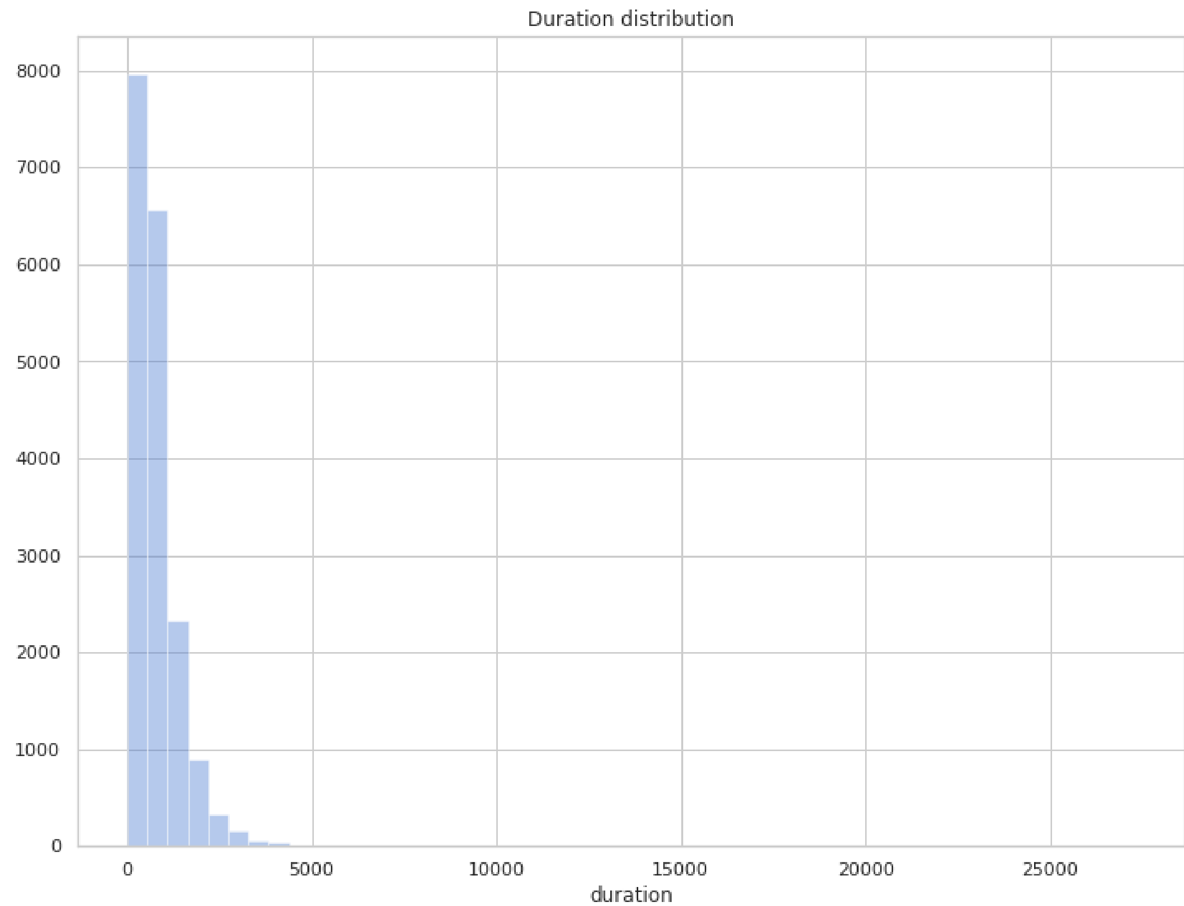
1: Data Overview

As a reminder, the raw taxi data contains the following columns:

- `recordID`: primary key of this database
- `VendorID`: a code indicating the provider associated with the trip record
- `passenger_count`: the number of passengers in the vehicle (driver entered value)
- `trip_distance`: trip distance
- `tpep_dropoff_datetime`: date and time when the meter was engaged
- `tpep_pickup_datetime`: date and time when the meter was disengaged
- `pickup_longitude`: the longitude where the meter was engaged
- `pickup_latitude`: the latitude where the meter was engaged
- `dropoff_longitude`: the longitude where the meter was disengaged
- `dropoff_latitude`: the latitude where the meter was disengaged
- `duration`: duration of the trip in seconds
- `payment_type`: the payment type
- `fare_amount`: the time-and-distance fare calculated by the meter
- `extra`: miscellaneous extras and surcharges
- `mta_tax`: MTA tax that is automatically triggered based on the metered rate in use
- `tip_amount`: the amount of credit card tips, cash tips are not included
- `tolls_amount`: amount paid for tolls
- `improvement_surcharge`: fixed fee
- `total_amount`: total amount paid by passengers, cash tips are not included

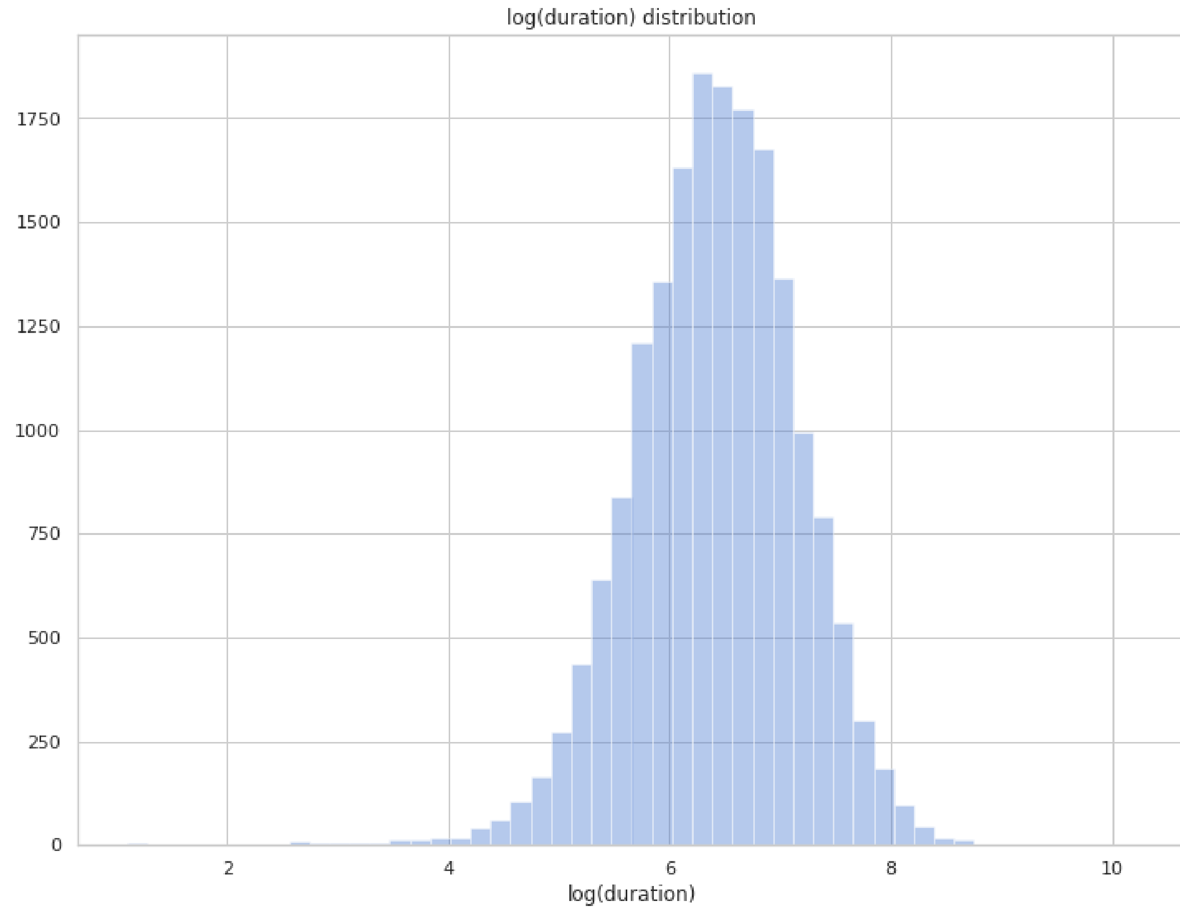
Let us take a closer look at the target `duration` variable. In the cell below, we plot its distribution using `sns.distplot`. This should give us an idea about whether we have some outliers in our data.

In [5]:
```
fig, ax = plt.subplots(figsize=(12, 9))

# Plot the distribution of duration using sns.distplot
# You can fill `ax=ax` to sns.distplot to plot in the ax object created
 above

sns.distplot(train_df['duration'], ax=ax, kde=False)

plt.title('Duration distribution');
```



As expected for a positive valued variable, we observe a skewed distribution. Note that we seem to have a handful of very long trips within our data. Use an appropriate data transformation to squeeze this highly-skewed distribution. Plot a `sns.distplot` of the transformed duration data for `train_df`.

In [6]:
```
fig, ax = plt.subplots(figsize=(12, 9))

# Use a log transformation to squeeze the distribution
# You can add + 1 to all values before taking the log to handle possible
 0 values for distribution

sns.distplot(np.log(train_df['duration'] + 1),
             ax=ax,
             axlabel='log(duration)',
             kde=False)

plt.title('log(duration) distribution');
```



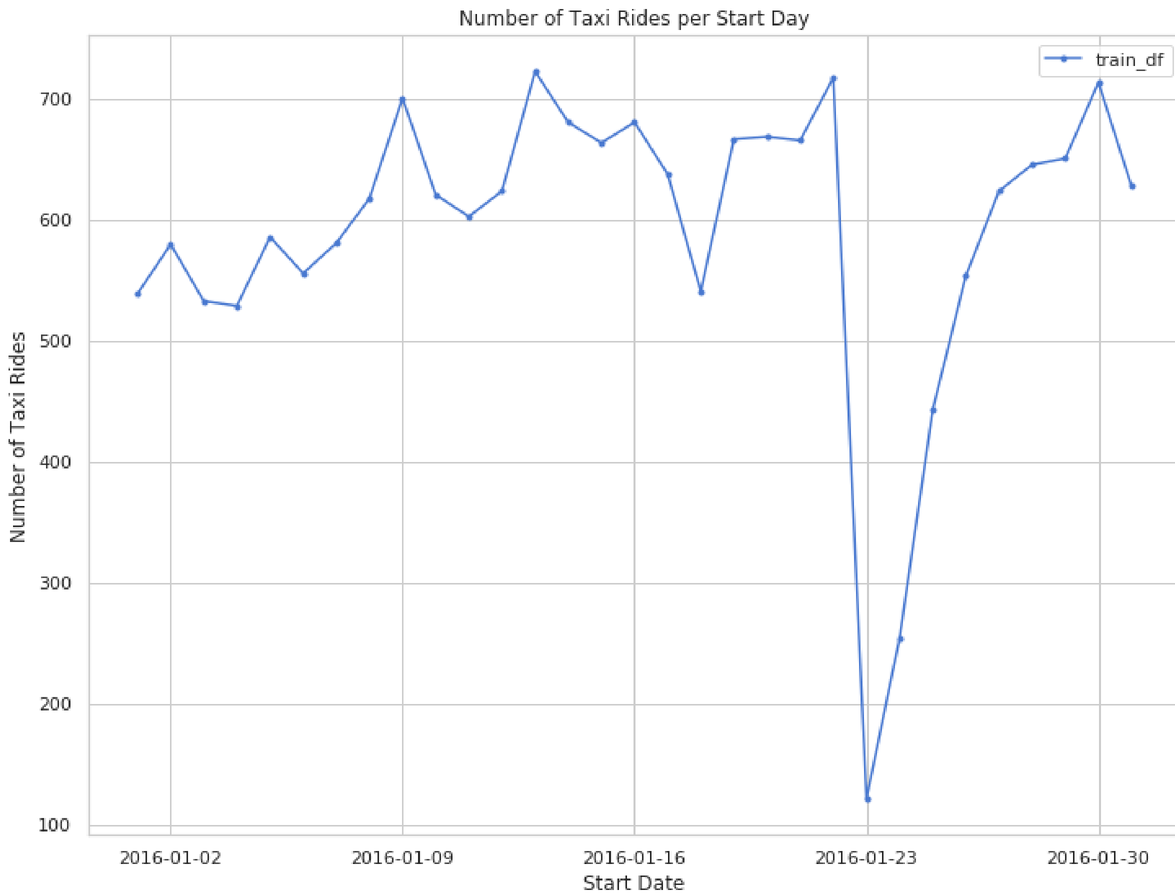
After transforming our data, we should immediately observe that we are dealing with what seems to be log-normal distribution for the target variable `duration`. We can see the behavior of shorter rides better, whereas before they were lumped in a bar near 0. This is a nice result, since it can facilitate modeling later.

Note: Keep in mind that we want to avoid peeking at our validation data because it may introduce bias. Therefore, we will be focusing on analyzing the training data for the remainder of this notebook.

2: Date Analysis

In order to understand the general pattern/trends of our taxi ride data, we will plot the number of taxi rides requested over time. Please run the following cell.

In [7]:
```
plt.figure(figsize=(12, 9))

# Make a temporary copy of our datasets
tmp_train = train_df.copy()
tmp_train['date'] = tmp_train['tpep_pickup_datetime'].dt.date
tmp_train = tmp_train.groupby('date').count()['pickup_longitude']

# Plot the temporal overlap
plt.plot(tmp_train, '.-', label='train_df')

plt.title('Number of Taxi Rides per Start Day')
plt.xlabel("Start Date")
plt.legend()
plt.ylabel('Number of Taxi Rides');
```



Question 2a

Taking a closer look at the plot above, we notice a drastic drop in taxi rides towards the end of Janurary. What is the date corresponding to the lowest number of taxi rides? Enter your answer as a string in the format MM-DD-YYYY.

In [8]:
```
lowest_rides_date = "01-23-2016"

print(lowest_rides_date)
```

```
01-23-2016
```

In [9]:
```
# Hidden test!
```

Question 2b

What event could have caused this drop in taxi rides? Feel free to use Google.

In [10]:
```
q2b_answer = r"""

Category 5 blizzard delivers record 3ft of snow to the Mid-Atlantic and
 Northeast US, killing 55 and incurring
estimated $500m - $3bn economic losses

"""

print(q2b_answer)
```

```
Category 5 blizzard delivers record 3ft of snow to the Mid-Atlantic and
Northeast US, killing 55 and incurring
estimated $500m - $3bn economic losses
```

3. Spatial/Locational Analysis

We are curious about the distribution of taxi pickup/dropoff coordinates. We also may be interested in observing whether this distribution changes as we condition of longer/shorter taxi rides. In the cells below, we will categorize our data into long and short rides based on duration. Then we will plot the latitude and longitude coordinates of rides conditioned on these categories.

First you may want to familiarize yourself with a map of Manhattan (https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m1!4b 73.9712488).

Here we split `train_df` into two data frames, one called `short_rides` and one called `long_rides`. `short_rides` should contain all rides less than or equal to 15 minutes and `long_rides` should contain rides more than 15 minutes.

Note: We chose 15 minutes because the mean duration of a ride is roughly 700 seconds ~ 12 minutes. We then round up to the nearest nice multiple of 5. Note that you should adjust how you determine short/long rides and outliers when feature engineering.

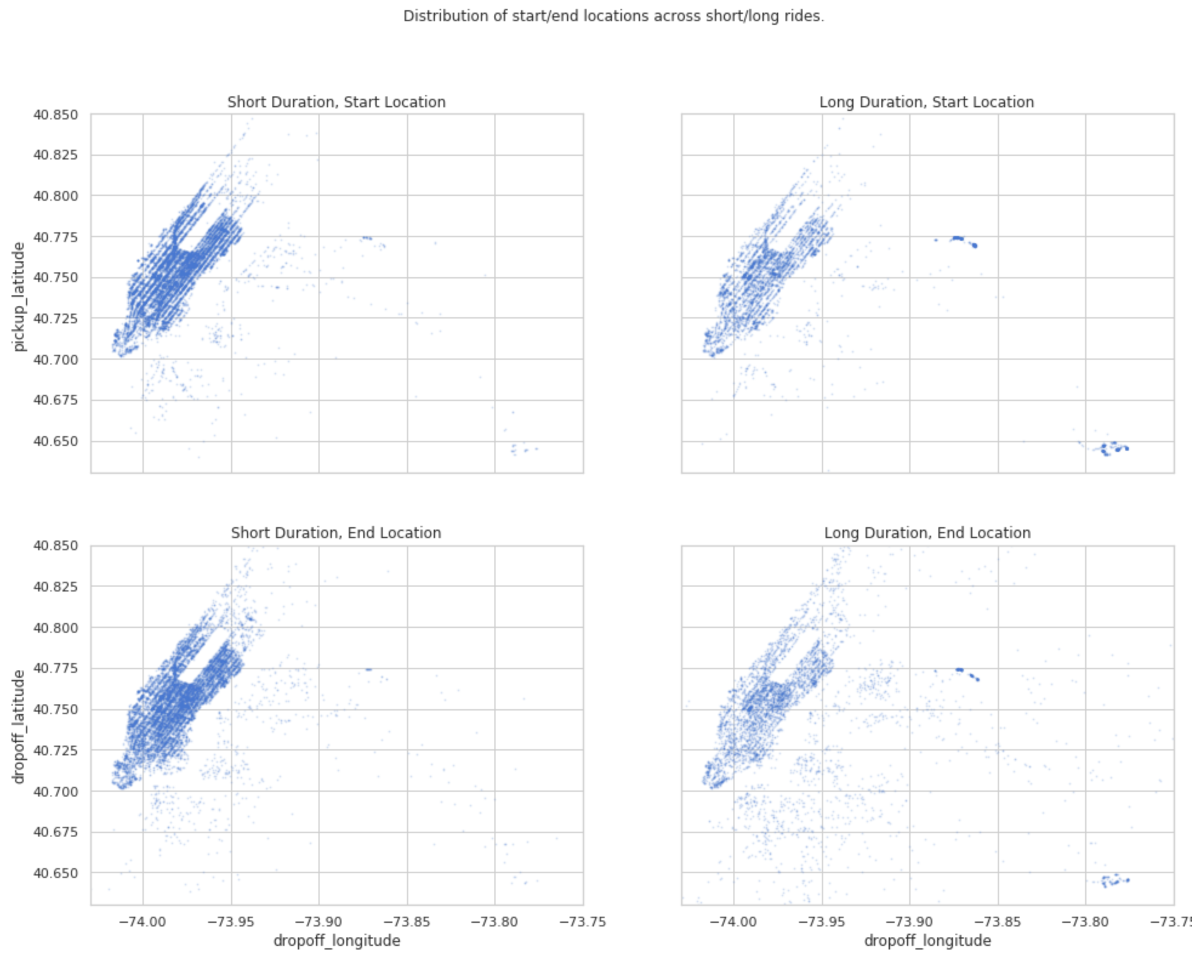
In [11]:
```
short_rides = train_df[train_df["duration"] <= 900] # rides less than or
 equal to 15 mins
long_rides = train_df[train_df["duration"] > 900] # rides more than 15 m
inutes
```

In [12]:
```
assert len(short_rides) == 12830
assert len(long_rides) == 5524
```

Below we generate 4 scatter plots. The scatter plots are ordered as follows:

- ax1: plot the **start** location of short duration rides
- ax2: plot the **start** location of long duration rides
- ax3: plot the **end** location of short duration rides
- ax4: plot the **end** location of long duration rides

In [13]:
```
# Set random seed of reproducibility
random.seed(42)

# City boundaries
city_long_border = (-74.03, -73.75)
city_lat_border = (40.63, 40.85)

# Define figure
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(ncols=2, nrows = 2, figsize
=(16, 12), sharex=True, sharey=True)
alpha = 0.15 # make sure to include these as an argument
s = 1 # make sure to include this as an argument

short_rides.plot(kind = "scatter", x = "pickup_longitude", y = "pickup_l
atitude",
                     ax = ax1, alpha = alpha, s = s, title='Short Durati
on, Start Location')
long_rides.plot(kind = "scatter", x = "pickup_longitude", y = "pickup_la
titude",
                    ax = ax2, alpha = alpha, s = s, title='Long Duratio
n, Start Location')
short_rides.plot(kind = "scatter", x = "dropoff_longitude", y = "dropoff
_latitude",
                     ax = ax3, alpha = alpha, s = s , title='Short Durat
ion, End Location')
long_rides.plot(kind = "scatter", x = "dropoff_longitude", y = "dropoff_
latitude",
                    ax = ax4, alpha = alpha, s = s, title='Long Duratio
n, End Location')


fig.suptitle('Distribution of start/end locations across short/long ride
s.')


plt.ylim(city_lat_border)
plt.xlim(city_long_border);
```



Question 3a

What do the plots above look like?

In particular:

- Find what the following circled regions correspond to:

Hint: Here is a [page](https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m) [(https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m 73.9712488)] that may be useful.

In [14]:
```
q3a_answer = r"""

The big circled region on the left corresponds to Manhattan.  You can ev
en see how it very closely resembles the
shape of Manhattan.  The region circled within the big circle on the lef
t refers to central park.  Finally the
small circled region on the right corresponds to LaGuardia Airport.

"""

print(q3a_answer)
```

```
The big circled region on the left corresponds to Manhattan.  You can e
ven see how it very closely resembles the
shape of Manhattan.  The region circled within the big circle on the le
ft refers to central park.  Finally the
small circled region on the right corresponds to LaGuardia Airport.
```

Question 3b

In each scatter plot above, why are there no points contained within the small rectangular region (towards the top left between the blue points)? Could this be an error/mistake in our data?

In [15]:
```
q3b_answer = r"""

This is not an error/mistake in the data. There are no points within the
 small rectangular region because that
region corresponds to central park.  Although it is theoretically possib
le to drive a taxi in central park, it
is frowned upon, plus there are no roads in CP, so consequently, no data
 points for taxi pickups and dropoffs.

"""


print(q3b_answer)
```

```
This is not an error/mistake in the data. There are no points within th
e small rectangular region because that
region corresponds to central park.  Although it is theoretically possi
ble to drive a taxi in central park, it
is frowned upon, plus there are no roads in CP, so consequently, no dat
a points for taxi pickups and dropoffs.
```

Question 3c

What observations/conclusions do you make based on the scatter plots above? In particular, how are trip duration and pickup/dropoff location related?

In [16]:
```
q3c_answer = r"""

For the scatter plots above there are a couple conclusions that I have d
rawn.  The first is that the short rides are
more concentrated within Manhattan.  This is seen as the deeper color of
 blue in the Manhattan region corresponding
to more pickups and dropoffs within Manhattan. In contrast, the long rid
es are slightly more spread out than the
short rides. You can see this by the way the long rides scatter plots ha
ve their point distribution scattered more
about the entire plot and how the Manhattan part of the scatter plot is
 a lighter blue. This makes sense because
people commonly take taxis throughout Manhattan, but outside of the cit
y, there is a better chance that they will
use other modes of transport. The second conclusion that I drew was that
 long rides have more trips too and from
airports.  This is seen by the higher concentration of data points at La
Guardia airport (the circled region to the
right in Q3a) and the small grouping of points in the bottom right corne
r (You can only see it in long rides scatter
plots).

"""

print(q3c_answer)
```


```
For the scatter plots above there are a couple conclusions that I have
drawn.  The first is that the short rides are
more concentrated within Manhattan.  This is seen as the deeper color o
f blue in the Manhattan region corresponding
to more pickups and dropoffs within Manhattan. In contrast, the long ri
des are slightly more spread out than the
short rides. You can see this by the way the long rides scatter plots h
ave their point distribution scattered more
about the entire plot and how the Manhattan part of the scatter plot is
a lighter blue. This makes sense because
people commonly take taxis throughout Manhattan, but outside of the cit
y, there is a better chance that they will
use other modes of transport. The second conclusion that I drew was tha
t long rides have more trips too and from
airports.  This is seen by the higher concentration of data points at L
aGuardia airport (the circled region to the
right in Q3a) and the small grouping of points in the bottom right corn
er (You can only see it in long rides scatter
plots).
```

This confirms that the trips are localized in NYC, with a very strong concentration in Manhattan **and** on the way to LaGuardia Airport. This might give you ideas of relevant features for feature engineering.

Another way to visualize ride coordinates is using a **heat map** (this also helps us avoid overplotting). The following plots count the number of trips for NYC neighborhoods and areas, plotting with the `geopandas` package and theses shapefiles (https://geo.nyu.edu/catalog/nyu_2451_36743) (do not mind the values on the colorbar). If you are curious about how to create the figures below, feel free to check out `geopandas` (http://geopandas.org/).

neighborhoods

neighborhoods

4: Temporal features

We can utilize the `start_timestamp` column to design a lot of interesting features.

We implement the following temporal (related to time) features using the `add_time_columns` function below.

- `month` derived from `start_timestamp`.
- `week_of_year` derived from `start_timestamp`.
- `day_of_month` derived from `start_timestamp`.
- `day_of_week` derived from `start_timestamp`.
- `hour` derived from `start_timestamp`.
- `week_hour` derived from `start_timestamp`.

Note 1: You can use the `dt` attribute of the `start_timestamp` column to convert the entry into a `DateTime` object.

Note 2: We set `df.is_copy = False` to explicitly write back to the original dataframe, `df`, that is being passed into the `add_time_columns` function. Otherwise `pandas` will complain.

In [17]:
```
def add_time_columns(df):
    """
    Add temporal features to df
    """
    df.is_copy = False
    df.loc[:, 'month'] = df['tpep_pickup_datetime'].dt.month
    df.loc[:, 'week_of_year'] = df['tpep_pickup_datetime'].dt.weekofyear
    df.loc[:, 'day_of_month'] = df['tpep_pickup_datetime'].dt.day
    df.loc[:, 'day_of_week'] = df['tpep_pickup_datetime'].dt.dayofweek
    df.loc[:, 'hour'] = df['tpep_pickup_datetime'].dt.hour
    df.loc[:, 'week_hour'] = df['tpep_pickup_datetime'].dt.weekday * 24
+ df['hour']

    # No real need to return here, but we harmonize with remove_outliers
 for later pipelinezation
    return df
```

In [18]:
```
# Note that we are applying this transformation to train_df, short_rides
 and long_rides
train_df = add_time_columns(train_df)
short_rides = add_time_columns(short_rides)
long_rides = add_time_columns(long_rides)
```

```
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generi
c.py:4388: FutureWarning: Attribute 'is_copy' is deprecated and will be
removed in a future version.
  object.__getattribute__(self, name)
/srv/conda/envs/data100/lib/python3.6/site-packages/pandas/core/generi
c.py:4389: FutureWarning: Attribute 'is_copy' is deprecated and will be
removed in a future version.
  return object.__setattr__(self, name, value)
```

In [19]:
```
train_df[['month', 'week_of_year', 'day_of_month', 'day_of_week', 'hour'
, 'week_hour']].head()
```

Out[19]:

	month	week_of_year	day_of_month	day_of_week	hour	week_hour
16434	1	3	21	3	17	89
21929	1	4	29	4	23	119
3370	1	1	5	1	18	42
21975	1	4	30	5	0	120
13758	1	3	18	0	13	13

Your `train_df.head()` should look like this, although the ordering of the data in `id` might be different:

time_columns

In [20]:
```
time_columns = ['month',
                'week_of_year',
                'day_of_month',
                'day_of_week',
                'hour',
                'week_hour']


# Check columns were created
assert all(column in train_df.columns for column in time_columns)

# Check type
assert train_df[time_columns].dtypes.nunique() == 1

assert train_df[time_columns].dtypes.nunique() == 1
```

Visualizing Temporal Features

Question 4a

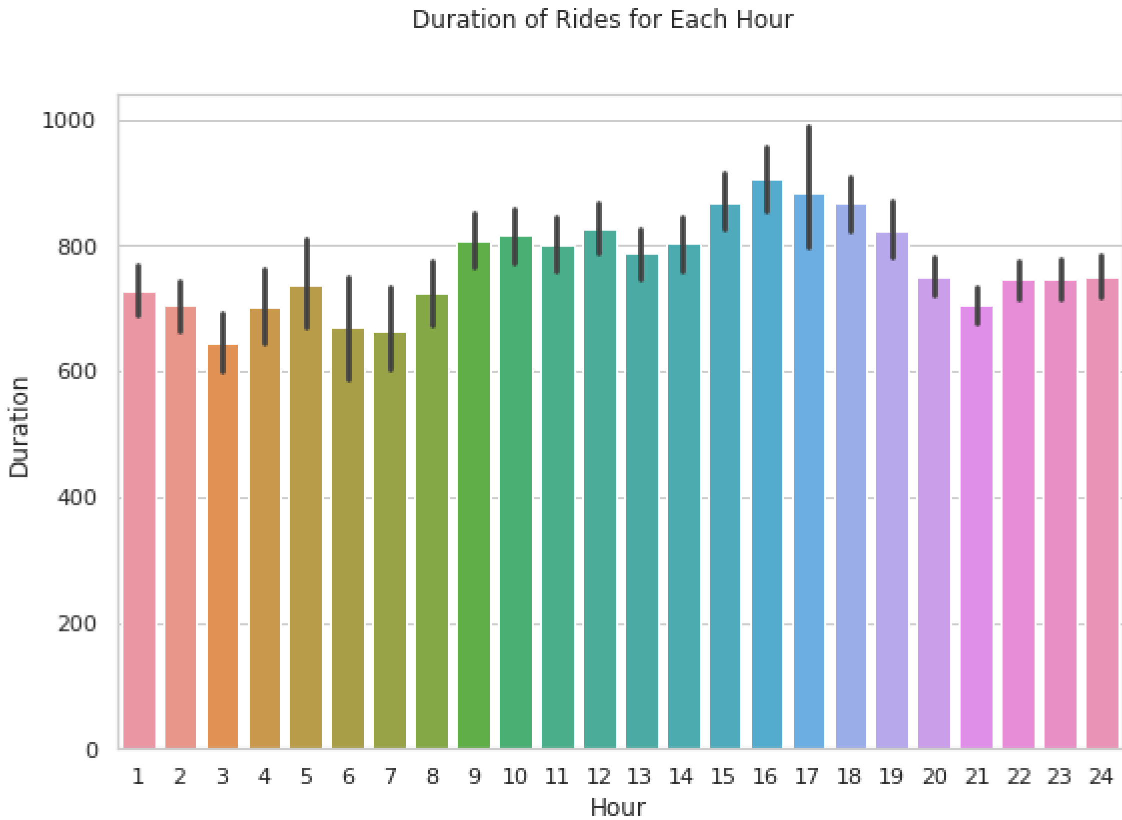
Let us now use the features we created to plot some histograms and visualize patterns in our dataset. We will analyze the distribution of the number of taxi rides across months and days of the week. This can help us visualize and understand patterns and trends within our data.

This is a open ended question. Create 2 plots that visualize temporal information from our dataset. At least one of them must visualize the hour of each day. Aside from that you can use any column from `time_columns`.

You can use the same column multiple times, but if the plots are redundant you will not receive full credit. This will be graded based on how informative each plot is and how "good" the visualization is (remember what good/bad visualizations look like for different kinds of data!).

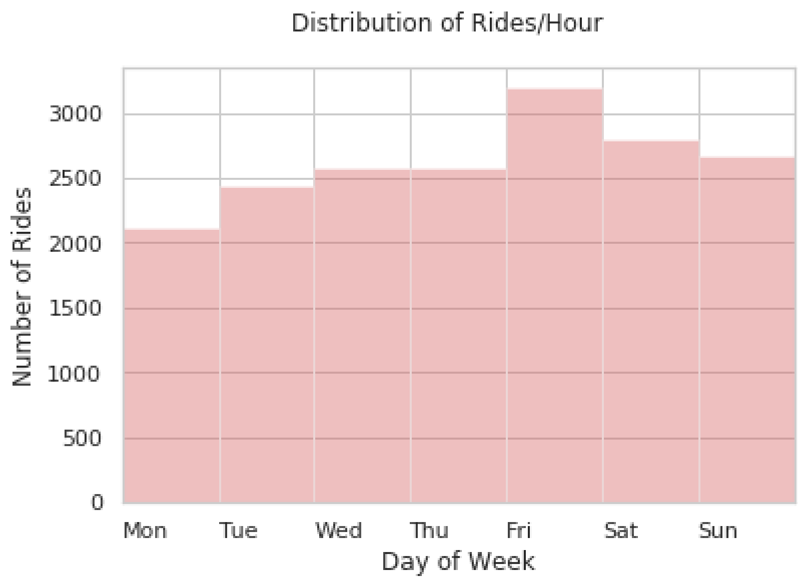
Visualization 1

In [21]:
```
# Visualization 1
bins = list(range(0, 25))
fig, ax = plt.subplots(figsize=(9,6))
sns.barplot(train_df['hour'], train_df['duration'])
ax.set_xticklabels(list(range(1, 25)))
ax.set_xlabel('Hour')
ax.set_ylabel('Duration')
fig.suptitle('Duration of Rides for Each Hour');
```



Visualization 2

In [22]:
```
# Visualization 2
fig, ax = plt.subplots()
sns.distplot(train_df['day_of_week'], bins=list(range(0, 8)), kde=False,
 color='r');
ax.set_xlim(0, 7)
ax.set_xticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'], ho
rizontalalignment='left')
ax.set_xlabel('Day of Week')
ax.set_ylabel('Number of Rides')
fig.suptitle('Distribution of Rides/Hour');
```



Question 4b

Briefly explain for each plot

1. What feature you're visualization
2. Why you chose this feature
3. Why you chose this visualization method

In [23]:
```
q4b_answer = r"""

For Visualization 1, I chose to visualize the duration of rides for each
 hour in the day. The reason I chose this
feature was to compare the duration of taxi rides for different times th
roughout the day.  The reason I chose the
seaborn barplot was because it allows you to plot an x variable against
 a y variable while keeping the plot visually
appealing.  I initially tried to plot hour against duration using a scat
ter plot, however, I realized that using a
barplot achieves the same effect and is easier to read.


For Visualization 2, I chose to visualize the distribution of rides over
 the 7 days of the week. The reason I chose
this feature was to see which days had a higher rate of taxi rides compa
red to others.  The reason I chose this
visualization method is similar to the reason in visualization 1: the ab
ility to view relative frequencies of rides
next to each other --> being able to compare the relative frequencies of
 rides throughout the days of the week.

"""


print(q4b_answer)
```

```
For Visualization 1, I chose to visualize the duration of rides for eac
h hour in the day. The reason I chose this
feature was to compare the duration of taxi rides for different times t
hroughout the day.  The reason I chose the
seaborn barplot was because it allows you to plot an x variable against
a y variable while keeping the plot visually
appealing.  I initially tried to plot hour against duration using a sca
tter plot, however, I realized that using a
barplot achieves the same effect and is easier to read.


For Visualization 2, I chose to visualize the distribution of rides ove
r the 7 days of the week. The reason I chose
this feature was to see which days had a higher rate of taxi rides comp
ared to others.  The reason I chose this
visualization method is similar to the reason in visualization 1: the a
bility to view relative frequencies of rides
next to each other --> being able to compare the relative frequencies o
f rides throughout the days of the week.
```

Question 4c

From the various plots above, what conclusions can you draw about the temporal aspects of our data? How does this relate to duration?

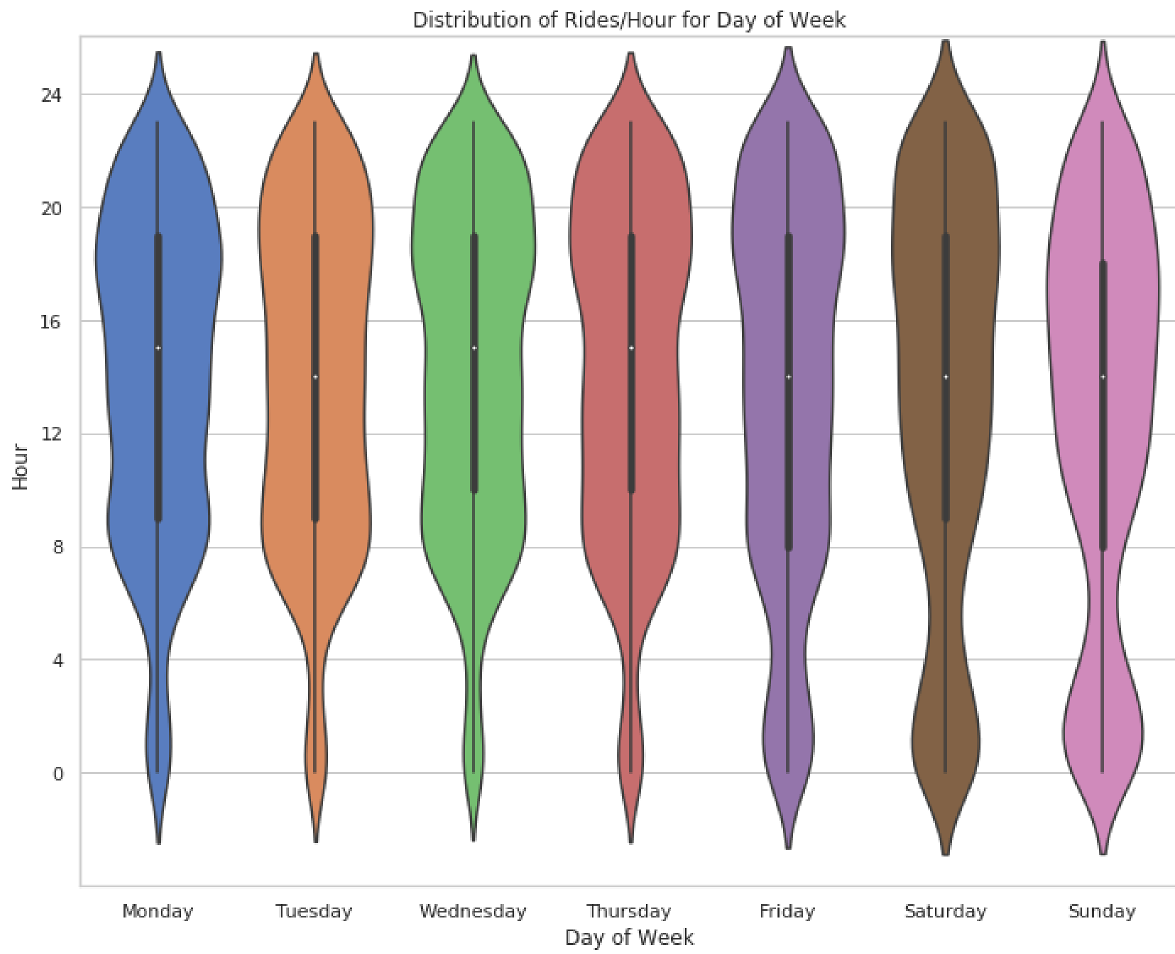
In [24]:
```
q4c_answer = r"""

From plot 1, we can see that ride durations are a bit lower during the m
orning hours, and a bit higher in the
afternoon around 2pm to 8pm. I'm inferring that the reason for this is t
he gradual increase of people coming into
Manhattan throughout the day, which in turn causes more traffic and long
er rides. This relates to duration because
the graph specifically shows how ride duration fluctuates throughout the
 day.

From plot 2, we can see that the days of the week have different distrib
utions of ride counts. From the plot we note
that ride counts are highest on the weekends, specifically on Friday.
 I'm inferring that the cause of this phenomena
is the increase of people traveling into the city on weekends to do fun
 weekend activities. Additionally, the
increase in the population's propensity to consume alcohol increase on w
eekends as well which also probably
influences the increase of ride counts on weekends.  This relates to dur
ation in the sense that the more rides there
are, the more traffic there is within the city, which most likely causes
 ride duration to lower.

"""

print(q4c_answer)
```

```
From plot 1, we can see that ride durations are a bit lower during the
morning hours, and a bit higher in the
afternoon around 2pm to 8pm. I'm inferring that the reason for this is
the gradual increase of people coming into
Manhattan throughout the day, which in turn causes more traffic and lon
ger rides. This relates to duration because
the graph specifically shows how ride duration fluctuates throughout th
e day.

From plot 2, we can see that the days of the week have different distri
butions of ride counts. From the plot we note
that ride counts are highest on the weekends, specifically on Friday.
I'm inferring that the cause of this phenomena
is the increase of people traveling into the city on weekends to do fun
weekend activities. Additionally, the
increase in the population's propensity to consume alcohol increase on
weekends as well which also probably
influences the increase of ride counts on weekends.  This relates to du
ration in the sense that the more rides there
are, the more traffic there is within the city, which most likely cause
s ride duration to lower.
```

Question 4d

Previously, we have analyzed the temporal features `hour` and `day_of_week` independently, but these features may in fact have a relationship between each other. Determining the extent to their relationship may be useful in helping us create new features in our model. Create a violin plot that displays distribution of rides over each hour per day of the week.

In [25]:
```
fig, axes = plt.subplots(1, 1, figsize=(10, 8))
days_of_week = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"]


sns.violinplot(x="day_of_week", y="hour", data=train_df).set_title('Dist
ribution of Rides/Hour for Day of Week')
axes.set_xticklabels(days_of_week)
axes.set_ylim(-4, 26)
axes.set_yticks(list(range(0, 25, 4)))
axes.set_ylabel('Hour')
axes.set_xlabel('Day of Week')

plt.tight_layout();
```



Question 4e

Do you notice anything interesting about your visualization? How would you explain this plot to a lay person? What are the features/patterns of interest?

In [26]:
```
q4e_answer = r"""

One interesting thing to note is that the distribution of rides for the
 late night/early morning hours, specifically
the hours from 11pm to 4am, is higher on the weekends.  This is probably
 due to the fact that people like to go out
in the city on the weekends.  We can also notice that around 8am and 5pm
 on the weekdays, there are slightly larger
bulges around the hours of 8am - 10am and 5pm - 8pm. This is most likely
 due to rush hour traffic.
"""

print(q4e_answer)
```

```
One interesting thing to note is that the distribution of rides for the
late night/early morning hours, specifically
the hours from 11pm to 4am, is higher on the weekends.  This is probabl
y due to the fact that people like to go out
in the city on the weekends.  We can also notice that around 8am and 5p
m on the weekdays, there are slightly larger
bulges around the hours of 8am - 10am and 5pm - 8pm. This is most likel
y due to rush hour traffic.
```

5: Vendors

Recall that in Part 1, we found that there are only two unique vendors represented in the dataset. We may wonder if the vendor feature can be useful when trying to understand taxi ride duration.

Question 5a

Visualize the VendorID feature. Create at least one plot that gives insight as to whether this feature would be useful or not in our model.

In [27]:
```
train_df.head()
```

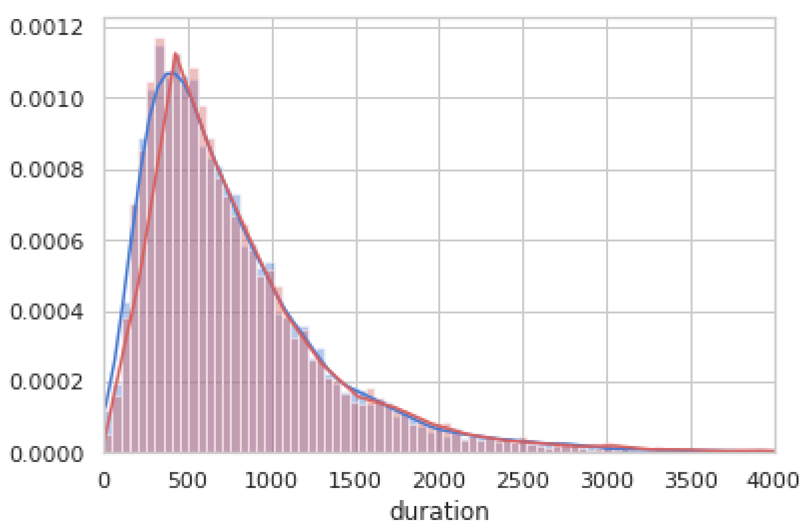
Out[27]:

	record_id	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_co
16434	8614300	2	2016-01-21 17:37:12	2016-01-21 18:37:56	2
21929	7230200	2	2016-01-29 23:22:26	2016-01-29 23:31:23	2
3370	9830300	2	2016-01-05 18:50:16	2016-01-05 18:56:00	2
21975	7251500	2	2016-01-30 00:14:34	2016-01-30 00:47:13	1
13758	6168000	1	2016-01-18 13:25:24	2016-01-18 13:38:51	1

5 rows × 27 columns

In [28]:
```
# Visualization
dataV1 = train_df[train_df['VendorID'] == 1]
dataV2 = train_df[train_df['VendorID'] == 2]
bins = np.linspace(0, 5000, 100)
fig, ax = plt.subplots()
#sns.scatterplot(train_df['trip_distance'], train_df['duration'], hue=tr
ain_df['VendorID'])
sns.distplot(dataV1['duration'], bins = bins)
sns.distplot(dataV2['duration'], bins = bins, color='r')
#ax.set_ylim(0, 10000)'
ax.set_xlim(0, 4000)
```

Out[28]: (0, 4000)



Question 5b

Justify why you chose this visualization method and how it helps determine whether `vendor_id` is useful in our model or not.

In [29]:
```
q5b_answer = r"""

I worked through a few different visualizations, but the one plot that I
 decided was the best was a distribution
plot where the durations were plotted for both VendorID's alondside each
 other.  The reason why I think this is
the best visualization method is that it clearly compares the distributi
ons of ride durations for the two Vendors
right next to each other. With this information we can clearly see that
 there is not much of a difference between
the vendors in ride duration, ergo, this would not be a useful feature i
n our model.

"""

print(q5b_answer)
```

```
I worked through a few different visualizations, but the one plot that
I decided was the best was a distribution
plot where the durations were plotted for both VendorID's alondside eac
h other.  The reason why I think this is
the best visualization method is that it clearly compares the distribut
ions of ride durations for the two Vendors
right next to each other. With this information we can clearly see that
there is not much of a difference between
the vendors in ride duration, ergo, this would not be a useful feature
in our model.
```

Question 5c

From the plot above, do you think vendor_id will help us understand duration? Why or why not?

In [30]:
```
q5c_answer = r"""

No I do not think that vendor_id will help us understand duration. The r
eason I think that it won't help is because
the distribution of ride durations for both vendors is virtually identic
al, which implies that using it as a feature
won't help us at all. In other words, both vendors have very similar rid
e durations across the board so if you were
to try and predict a specific ride duration, having the vendor_id would
n't offer much help.

"""

print(q5c_answer)
```

```
No I do not think that vendor_id will help us understand duration. The
reason I think that it won't help is because
the distribution of ride durations for both vendors is virtually identi
cal, which implies that using it as a feature
won't help us at all. In other words, both vendors have very similar ri
de durations across the board so if you were
to try and predict a specific ride duration, having the vendor_id would
n't offer much help.
```

6: Distance features

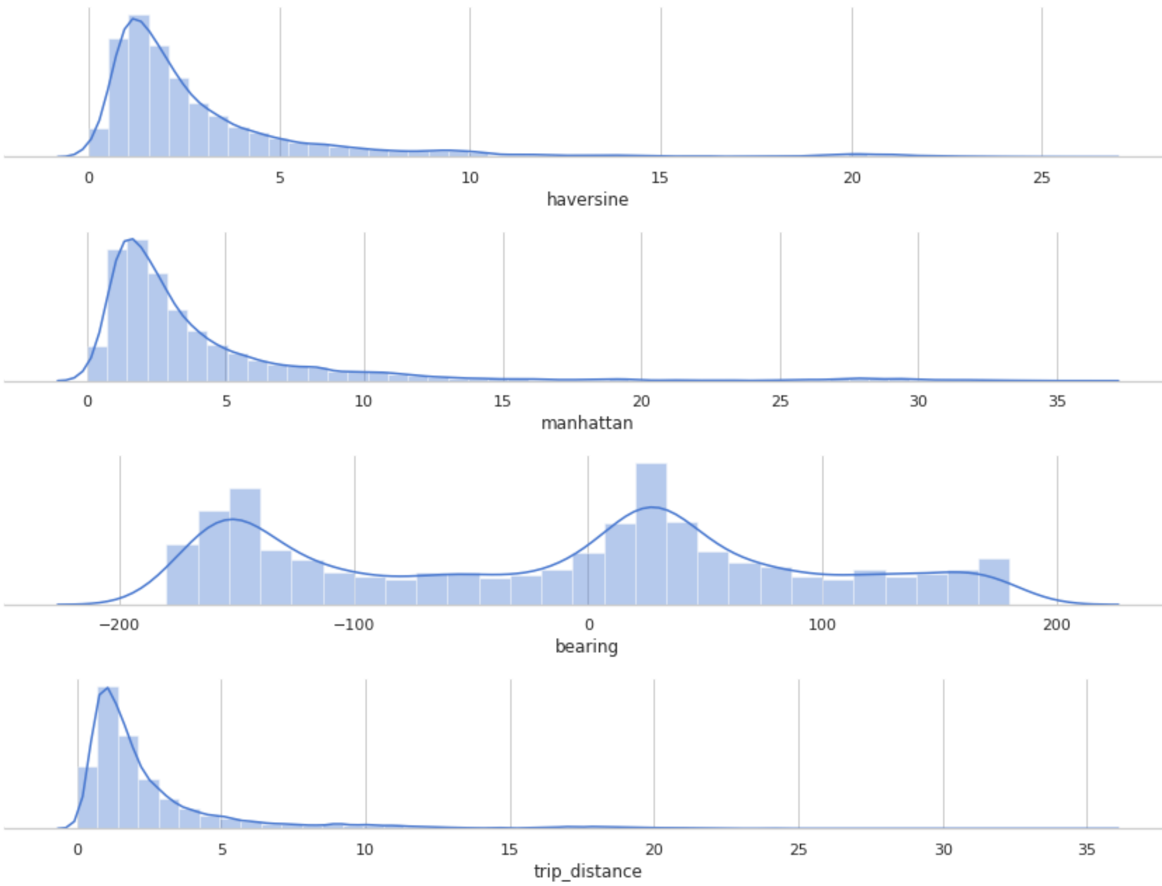
We can also use the coordinates information to compute distance features. This will allow us to compute speed related features.
We will compute the haversine (https://en.wikipedia.org/wiki/Haversine_formula) distance, the manhattan (https://en.wikipedia.org/wiki/Taxicab_geometry) distance and the bearing (http://www.mathsteacher.com.au/year7/ch08_angles/07_bear/bearing.htm) angle.

In [31]:
```
# These functions are implemented for you
def haversine(lat1, lng1, lat2, lng2):
    """
    Compute haversine distance

    The haversine formula determines the great-circle distance between t
wo points
    on a sphere given their longitudes and latitudes. Important in navig
ation, it
    is a special case of a more general formula in spherical trigonometr
y,
    the law of haversines, that relates the sides and angles of spherica
l triangles.
    """
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    average_earth_radius = 6371
    lat = lat2 - lat1
    lng = lng2 - lng1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(ln
g * 0.5) ** 2
    h = 2 * average_earth_radius * np.arcsin(np.sqrt(d))
    return h

def manhattan_distance(lat1, lng1, lat2, lng2):
    """
    Computes Manhattan distance

    The name alludes to the grid layout of most streets on the island of
 Manhattan,
    which causes the shortest path a car could take between two intersec
tions in the borough
    to have length equal to the intersections' distance in taxicab geome
try.
    """
    a = haversine(lat1, lng1, lat1, lng2)
    b = haversine(lat1, lng1, lat2, lng1)
    return a + b

def bearing(lat1, lng1, lat2, lng2):
    """
    Compute the bearing, or angle, from (lat1, lng1) to (lat2, lng2).
    A bearing of 0 refers to a NORTH orientation.
    """
    lng_delta_rad = np.radians(lng2 - lng1)
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    y = np.sin(lng_delta_rad) * np.cos(lat2)
    x = np.cos(lat1) * np.sin(lat2) - np.sin(lat1) * np.cos(lat2) * np.c
os(lng_delta_rad)
    return np.degrees(np.arctan2(y, x))
```

In [32]:
```
def add_distance_columns(df):
    df.loc[:, 'manhattan'] = manhattan_distance(lat1=df['pickup_latitud
e'],
                                                lng1=df['pickup_longitud
e'],
                                                lat2=df['dropoff_latitud
e'],
                                                lng2=df['dropoff_longitu
de'])

    df.loc[:, 'bearing'] = bearing(lat1=df['pickup_latitude'],
                                   lng1=df['pickup_longitude'],
                                   lat2=df['dropoff_latitude'],
                                   lng2=df['dropoff_longitude'])
    df.loc[:, 'haversine'] = haversine(lat1=df['pickup_latitude'],
                                   lng1=df['pickup_longitude'],
                                   lat2=df['dropoff_latitude'],
                                   lng2=df['dropoff_longitude'])

    return df
```

In [33]:
```
train_df = add_distance_columns(train_df)
short_rides = add_distance_columns(short_rides)
long_rides = add_distance_columns(long_rides)
```

In [34]:
```
fig, axes = plt.subplots(4, 1, figsize=(12, 9))
sns.distplot(train_df['haversine'], ax=axes[0], axlabel='haversine');
sns.distplot(train_df['manhattan'], ax=axes[1], axlabel='manhattan');
sns.distplot(train_df['bearing'], ax=axes[2], axlabel='bearing');
sns.distplot(train_df['trip_distance'], ax=axes[3], axlabel='trip_distan
ce');

sns.despine(left=True);
plt.setp(axes, yticks=[]);
plt.tight_layout();
```



Question 6a

The `bearing` direction is angle, the initial direction of the trip.
The bearing direction has two prominent peaks around 30 and -150 degrees.
Can you relate these peaks to the orientation of Manhattan? What do you notice about these angles?

Hint: This wikipedia article (https://en.wikipedia.org/wiki/Commissioners%27_Plan_of_1811) has the answer, although it may take some digging. Alternatively, try to look at a map of Manhattan.

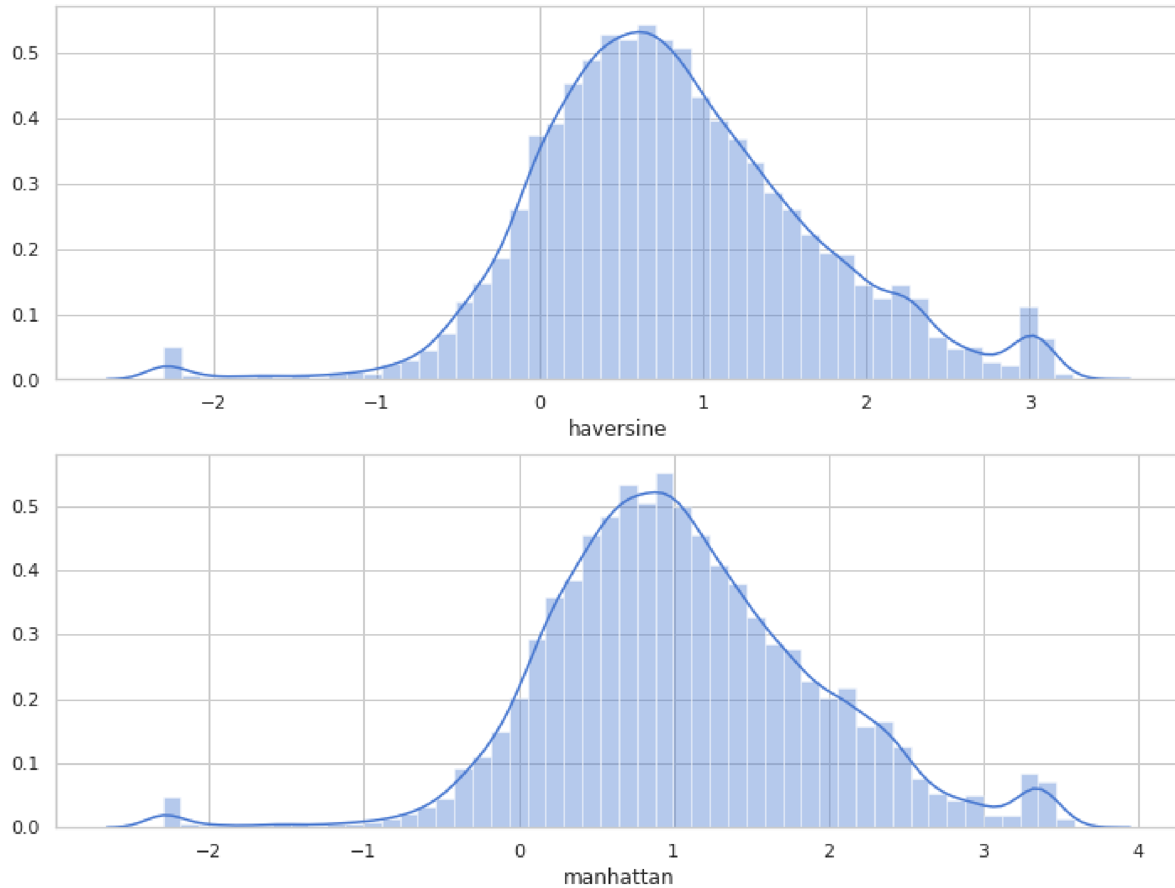
In [35]:
```
q6a_answer = r"""

When Manhattan was built, it was intended to be built in a true north-so
uth/east-west fashion, however, since
Manhattan is tilted at a 29°, the had to shift the north-south/east-west
 layout of streets 29° as well. Consequently,
we note that 30 and -150 degrees has a difference of 180 degrees. So the
 way that these peaks in bearing direction
relate to the orientation of Manhattan is that they are the long directi
ons of Manhattan, or in other words, the
directions moving up and down the length of Manhattan hot dog style, not
 hamburger style.

"""

print(q6a_answer)
```

```
When Manhattan was built, it was intended to be built in a true north-s
outh/east-west fashion, however, since
Manhattan is tilted at a 29°, the had to shift the north-south/east-wes
t layout of streets 29° as well. Consequently,
we note that 30 and -150 degrees has a difference of 180 degrees. So th
e way that these peaks in bearing direction
relate to the orientation of Manhattan is that they are the long direct
ions of Manhattan, or in other words, the
directions moving up and down the length of Manhattan hot dog style, no
t hamburger style.
```

Question 6b

For haversine and manhattan distances, it is probably more helpful to look at the log distribution. We are also curious about whether these distance features can help us understand duration. Create at least one plot that compares haversine and manhattan distances and gives insight as to whether this would be a useful feature in our model.

In [36]:
```
# Visualization
fig, axes = plt.subplots(2, 1, figsize=(12, 9))

logHaversine = np.log(train_df['haversine'] + 0.1)
logManhattan = np.log(train_df['manhattan'] + 0.1)
sns.distplot(logHaversine, ax=axes[0], axlabel='haversine')
sns.distplot(logManhattan, ax=axes[1], axlabel='manhattan');
```



Question 6c

Justify why you chose this visualization method and how it helps inform you about using manhattan/haversine distance as a feature for predicting trip duration.

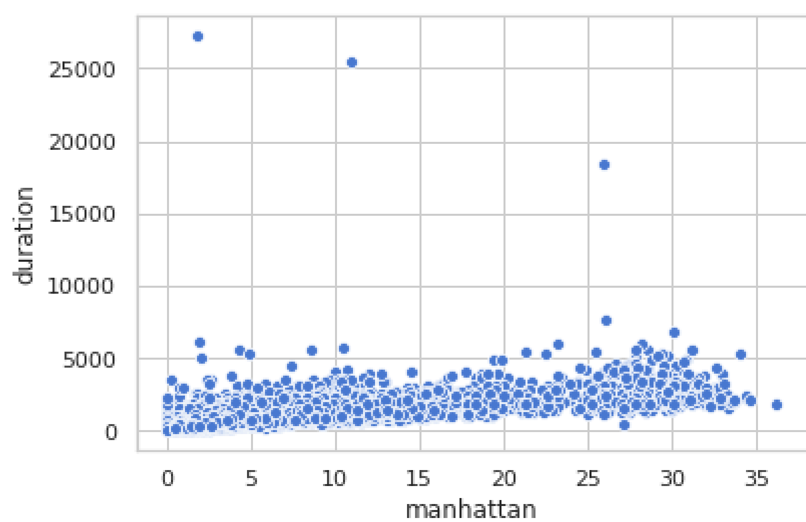
In [37]:
```
q6c_answer = r"""

The reason that I chose this visualization method is that you can easily
 compare the log distribution of manhattan
distances and haversine distances side by side.  As we notice though, th
ere is not much difference between the two
distributions, so we must conclude that using manhattan/haversine distan
ce as a feature will not be very useful.

"""

print(q6c_answer)
```

```
The reason that I chose this visualization method is that you can easil
y compare the log distribution of manhattan
distances and haversine distances side by side.  As we notice though, t
here is not much difference between the two
distributions, so we must conclude that using manhattan/haversine dista
nce as a feature will not be very useful.
```

Question 6d

Fill in the code below to plot a scatter plot of manhattan distance vs duration.

In [38]:
```
fig, ax = plt.subplots()
sns.scatterplot(train_df['manhattan'], train_df['duration']);
```



Question 6e

According to the plot above, there are a few outliers in both duration and manhattan distance. **Which type of outliers is most likely to be a mistake in our data?**

In [39]:
```
q6e_answer = r"""

Based on the three outliers we can see in the scatterplot, we can conclu
de that the type of outliers where there are
mistakes is duration. This is because the outliers are far more extreme
 than the other outliers.

"""

print(q6e_answer)
```

```
Based on the three outliers we can see in the scatterplot, we can concl
ude that the type of outliers where there are
mistakes is duration. This is because the outliers are far more extreme
than the other outliers.
```

7: Advanced features

You do not need to incorporate these features into your model, although it may help lower your error. You are required to read through this portion and respond to the questions. All of the code is provided, please skim through it and try to understand what each cell is doing.

Clustering

Clustering (https://en.wikipedia.org/wiki/Cluster_analysis) is the task of grouping objects such that members within each group are more similar to each other than members of other groups. Clustering is a powerful tool used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. Recall cluster sampling, which we learned earlier in the semester. We will use a simple clustering method (clustering by spatial locality) to reveal some more advanced features.

Speed features

For `train_df`, we have the `duration` and now some distance information.
This is enough for us to compute average speed and try to better understand our data.

For `test_df`, we cannot use `duration` as a feature because it is what we are trying to predict. One clever way to include speed information for modeling would be as follows:

1. Cluster the observations in `train_df` by rounding the latitude and longitudes.
2. Compute the average speed per pickup cluster and dropoff cluster.
3. Match each observation in `test_df` to its pickup cluster and dropoff cluster based off the latitude and longitude, thus assigning the average speed for the pickup and dropoff cluster.
4. We have added speed information as features for `test_df`.

Therefore, we have propagated information computed in the `train_df` into the `test_df` via clustering. This is not something we will do in this notebook, although you can try it for yourself!

Other information that could be added based on clustering (both pickup cluster and dropoff cluster):

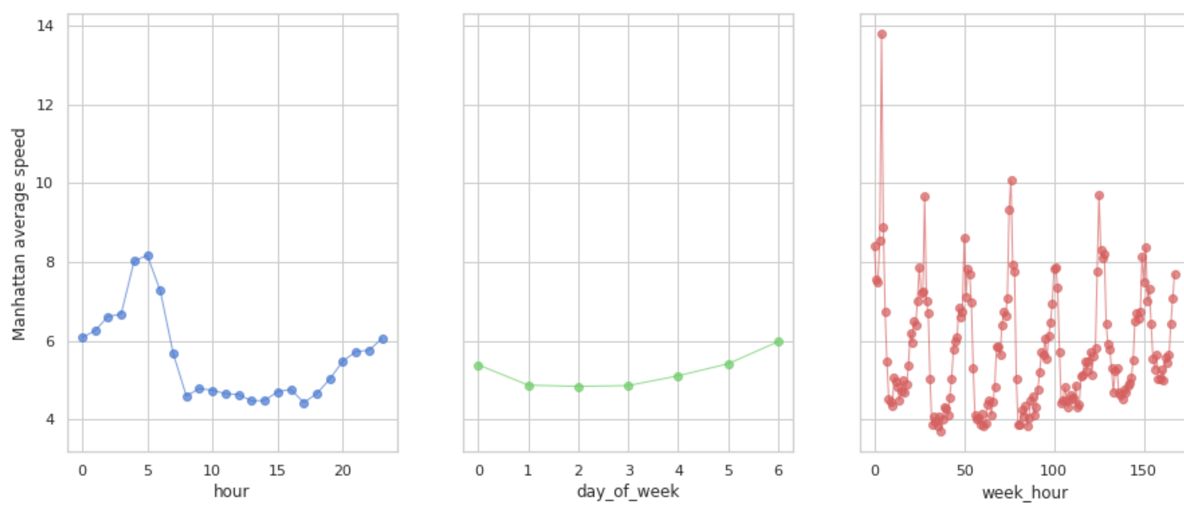
- Average of `avg_speed_h` per cluster.
- Average of `duration` per cluster.
- Average of `avg_speed_h` per cluster and hour.
- Average of `duration` per cluster and hour.
- In-cluster flow of trips for 60 min period.
- Out-cluster flow of trips for 60 min period.

In [40]:
```
# Calculate average manhattan speed
train_df['avg_speed_m'] = 1000 * train_df['manhattan'] / train_df['durat
ion']
train_df['avg_speed_m'] = train_df['avg_speed_m'][train_df['avg_speed_m'
] < 100]
train_df['avg_speed_m'].fillna(train_df['avg_speed_m'].median(), inplace
=True)
```

In [41]:
```
train_df['avg_speed_m'].describe()
```

Out[41]:
```
count    18354.000000
mean         5.210825
std          2.883174
min          0.000000
25%          3.287328
50%          4.617264
75%          6.413992
max         59.225577
Name: avg_speed_m, dtype: float64
```

In [42]:
```
# Visualize average manhattan speed by hour, day of week and week hour
fig, axes = plt.subplots(ncols=3, figsize=(15, 6), sharey=True)

axes[0].plot(train_df.groupby('hour').mean()['avg_speed_m'], 'bo-', lw=1
, alpha=0.7)
axes[1].plot(train_df.groupby('day_of_week').mean()['avg_speed_m'], 'go-
', lw=1, alpha=0.7)
axes[2].plot(train_df.groupby('week_hour').mean()['avg_speed_m'], 'ro-',
 lw=1, alpha=0.7)

axes[0].set_xlabel('hour')
axes[1].set_xlabel('day_of_week')
axes[2].set_xlabel('week_hour')
axes[0].set_ylabel('Manhattan average speed');
```



Question 7a

Based off of these visualizations, provide 2-3 insights on the average speed.

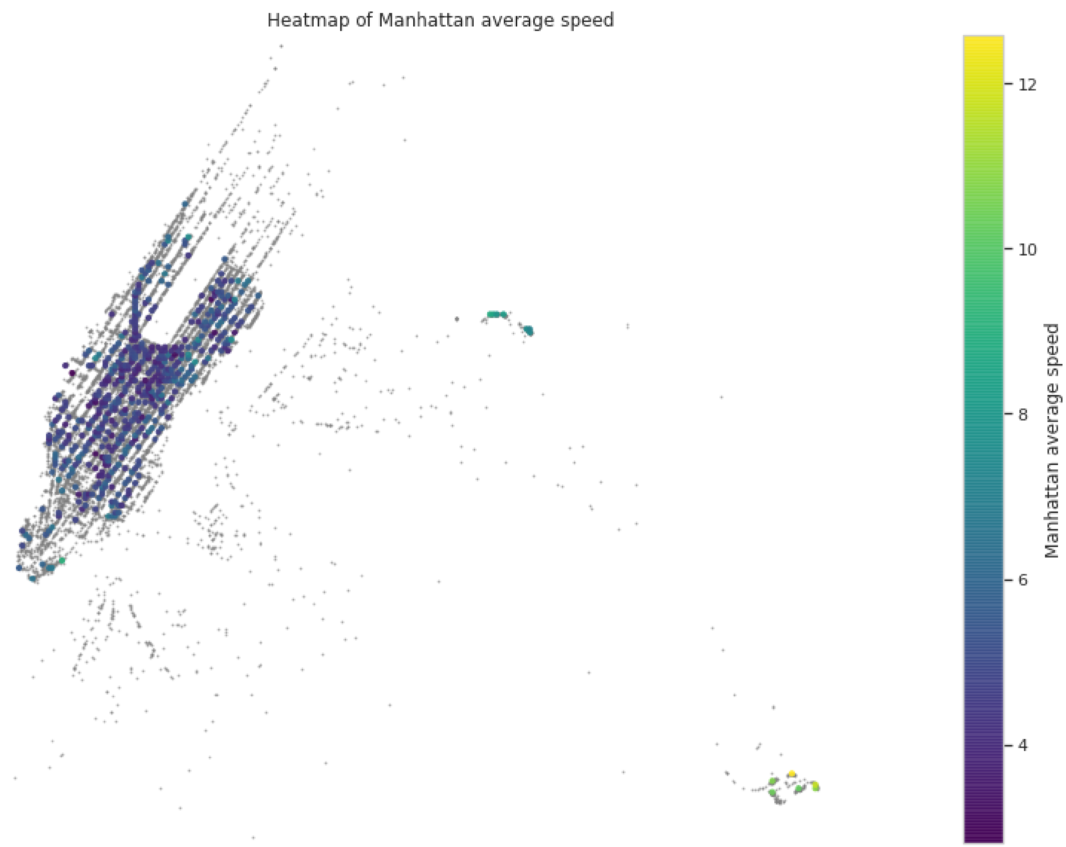
In [43]:
```
q7a_answer = r"""

Based on these visualizations, I can come up with a few valuable insight
s.  First is the fact that avg Manhattan
speed starts climbing at around 4 o'clock pm, climbs steeper at 3 am, pe
aks at 5 am, then steeply drops after that.
This corresponds to daily traffic levels - there is no traffic at night,
 and consequently, the night hours have the
highest avg manhattan speed.

The second insight is based on the day_of_week graph: the average manhat
tatan speed climbs around the weekends, and
lowers during the week days.  This is most likely due to the lower level
 of commuter traffic on the weekends. So less
cars = higher speeds.

The third and final insight is based on the week_hour graph, and helps s
upport the first insight of speeds being
higher at night.  As you can see in the week_hour graph, there are very
 distinct peaks during the nights of the week
and equally distinct valleys during the days of the week.  This further
 supports the insight of higher speeds at
night due to the lower level of traffic at night.

"""


print(q7a_answer)
```


```
Based on these visualizations, I can come up with a few valuable insigh
ts.  First is the fact that avg Manhattan
speed starts climbing at around 4 o'clock pm, climbs steeper at 3 am, p
eaks at 5 am, then steeply drops after that.
This corresponds to daily traffic levels - there is no traffic at nigh
t, and consequently, the night hours have the
highest avg manhattan speed.

The second insight is based on the day_of_week graph: the average manha
ttatan speed climbs around the weekends, and
lowers during the week days.  This is most likely due to the lower leve
l of commuter traffic on the weekends. So less
cars = higher speeds.

The third and final insight is based on the week_hour graph, and helps
support the first insight of speeds being
higher at night.  As you can see in the week_hour graph, there are very
distinct peaks during the nights of the week
and equally distinct valleys during the days of the week.  This further
supports the insight of higher speeds at
night due to the lower level of traffic at night.
```

We are now going to visualize the average speed per region. Here we define regions as a very basic classical clustering based on rounding of spatial coordinates.

In [44]:
```
# Round / bin the latitude and longitudes
train_df['start_lat_bin'] = np.round(train_df['pickup_latitude'], 3)
train_df['start_lng_bin'] = np.round(train_df['pickup_longitude'], 3)

# Average speed for regions
gby_cols = ['start_lat_bin', 'start_lng_bin']

coord_stats = (train_df.groupby(gby_cols)
               .agg({'avg_speed_m': 'mean', 'manhattan': 'count'})
               .reset_index())

coord_stats = coord_stats[coord_stats['manhattan'] > 10]
```

In [45]:
```
# Visualize the average speed per region
city_long_border = (-74.03, -73.75)
city_lat_border = (40.63, 40.85)
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(14, 10))

scatter_trips = ax.scatter(train_df['pickup_longitude'].values,
                           train_df['pickup_latitude'].values,
                           color='grey', s=1, alpha=0.5)

scatter_cmap = ax.scatter(coord_stats['start_lng_bin'].values,
                          coord_stats['start_lat_bin'].values,
                          c=coord_stats['avg_speed_m'].values,
                          cmap='viridis', s=10, alpha=0.9)

cbar = fig.colorbar(scatter_cmap)
cbar.set_label("Manhattan average speed")
ax.set_xlim(city_long_border)
ax.set_ylim(city_lat_border)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.title('Heatmap of Manhattan average speed')
plt.axis('off');
```



Question 7b

In 2-3 sentences, describe how we can use the clustering visualization above to gain insight on the speed. Do you think spatial clustering would be useful in reducing the error of our model?

In [46]:
```
q7b_answer = r"""

We can use clustering visulization to group the data points based on lat
itude and longitude into their respective
speed clusters which will allow us to predict ride duration based partia
lly on location.  This will help us reduce
error because there are areas where speed is clearly lower (middle of ma
nhattan/around central park) and other areas
where speed is clearly higher (LaGuardia Airport and JFK Airport). So if
 we factor this into our model, it will
theoretically increase accuracy, and decrease error.


"""


print(q7b_answer)
```

```
We can use clustering visulization to group the data points based on la
titude and longitude into their respective
speed clusters which will allow us to predict ride duration based parti
ally on location.  This will help us reduce
error because there are areas where speed is clearly lower (middle of m
anhattan/around central park) and other areas
where speed is clearly higher (LaGuardia Airport and JFK Airport). So i
f we factor this into our model, it will
theoretically increase accuracy, and decrease error.
```

Part 2 Exports

We are not requiring you to export anything from this notebook, but you may find it useful to do so. There is a space below for you to export anything you wish.

In [47]:
```
Path("data/part2").mkdir(parents=True, exist_ok=True)
data_file = Path("data/part2", "data_part2.hdf") # Path of hdf file
...
```

Out[47]: Ellipsis

Part 2 Conclusions

We now have a good understanding of the taxi data we are working with. Visualizing large amounts of data can be a difficult task. One helpful tool is datashader (https://github.com/bokeh/datashader), a data rasterization pipeline for automating the process of creating meaningful representations of large amounts of data. Using the geopandas (http://geopandas.org/) package also makes working with geospatial data easier. We encourage you to explore these tools if you are interested in learning more about visualization!

Within our taxi data set, we have explored different features and their relationship with ride duration. Now, we are ready to incorporate more data in order to add to our set of features.

Please proceed to part 3 where we will be engineering more features and building our models using a processing pipeline.

Submission

You're almost done!

Before submitting this assignment, ensure that you have:

1. Restarted the Kernel (in the menubar, select Kernel→Restart & Run All)
2. Validated the notebook by clicking the "Validate" button.

Then,

1. **Submit** the assignment via the Assignments tab in **Datahub**
2. **Upload and tag** the manually reviewed portions of the assignment on **Gradescope**