

Análise da Bay Area Bike Share

Introdução

Dica: Seções citadas como esta fornecerão instruções úteis sobre como navegar e usar um notebook do iPython.

Bay Area Bike Share (<http://www.bayareabikeshare.com/>) é uma empresa que oferece aluguel de bicicletas on-demand para clientes em San Francisco, Redwood City, Palo Alto, Mountain View e San Jose. Os usuários podem desbloquear bicicletas de uma variedade de estações em cada cidade, e devolvê-las em qualquer estação dentro da mesma cidade. Os usuários pagam o serviço por meio de assinatura anual ou pela compra de passes de 3 dias ou 24 horas. Os usuários podem fazer um número ilimitado de viagens. Viagens com menos de trinta minutos de duração não têm custo adicional; Viagens mais longas incorrem em taxas de horas extras.

Neste projeto, você vai se colocar no lugar de um analista de dados realizando uma análise exploratória sobre os dados. Você vai dar uma olhada em duas das principais partes do processo de análise de dados: limpeza de dados e análise exploratória. Mas antes que você comece a olhar os dados, pense algumas perguntas que você pode querer fazer sobre os dados. Considere, por exemplo, se você estivesse trabalhando para Bay Area Bike Share: que tipo de informação você gostaria de saber a fim de tomar decisões de negócios mais inteligentes? Ou você pode pensar se você fosse um usuário do serviço de compartilhamento de bicicletas. Que fatores podem influenciar a maneira como você gostaria de usar o serviço?

Sobre este projeto

Este é o seu primeiro projeto com a Udacity. Queremos fazer com que você treine os conhecimentos aprendidos durante o curso e que entenda algumas das dificuldades que pode ter quando for aplicar os mesmos.

Os principais pontos que serão verificados neste trabalho:

- Criação de dicionários e mapeamento de variáveis
- Uso de lógica com o `if`
- Manipulação de dados e criação de gráficos simples com o Pandas

Como conseguir ajuda: Sugerimos que tente os seguintes canais, nas seguintes ordens:

Tipo de dúvida\Canais	Google	Fórum	Slack	Email
Programação Python e Pandas	1	2	3	
Requisitos do projeto		1	2	3
Partes específicas do Projeto		1	2	3

Os endereços dos canais são:

- Fórum: <https://discussions.udacity.com/c/ndfdsi-project> (<https://discussions.udacity.com/c/ndfdsi-project>)
- Slack: udacity-br.slack.com (<https://udacity-br.slack.com/messages/C5MT6E3E1>)
- Email: data-suporte@udacity.com

Espera-se que o estudante entregue este relatório com:

- Todos os TODO feitos, pois eles são essenciais para que o código rode corretamente
- Todas as perguntas respondidas. Elas estão identificadas como PERGUNTA em letras grandes.

Para entregar este projeto, vá a [sala de aula](#)

(<https://classroom.udacity.com/nanodegrees/nd110/parts/05e3b1e2-ff23-488f-aaec-caba12ad8ed3/modules/519425b3-ee26-4ecd-a952-f689decef51c/lessons/47133342-a203-4df9-9b9e-69b04408e089/project>) e submeta o seu `.ipynb` e um pdf, zipados.

Pergunta 1

Escreva pelo menos duas perguntas que você acha que poderiam ser respondidas usando os dados.

Resposta:

- Qual o tempo médio de aluguel de bicicletas por estação?
- Qual a rota mais utilizada?

Usando visualizações para comunicar resultados em dados

Como um analista de dados, a capacidade de comunicar eficazmente resultados é uma parte fundamental do trabalho. Afinal, sua melhor análise é tão boa quanto sua capacidade de comunicá-la.

Em 2014, Bay Area Bike Share realizou um Open Data Challenge (<http://www.bayareabikeshare.com/datachallenge-2014>) para incentivar os analistas de dados a criar visualizações com base em seu conjunto de dados aberto. Você criará suas próprias visualizações neste projeto, mas primeiro, dê uma olhada no vencedor de inscrições para Melhor Análise (<http://thfield.github.io/babs/index.html>) de Tyler Field. Leia todo o relatório para responder à seguinte pergunta:

Pergunta 2

Que visualizações você acha que fornecem as idéias mais interessantes?

Selecione duas visualizações de Tyler que forneçam idéias interessantes e que respondam alguma das perguntas da análise de Tyler.

Resposta :

- A identificação do tipo de cliente que mais utiliza o serviço.
- A popularidade das estações em relação os assinantes(subscribers) e os clientes (customers).

Pergunta 2.1

Quais são as perguntas que foram respondidas com suas visualizações? Porque você as escolheu?

Resposta :

- Foi respondido que tipo de cliente que mais utilizado o serviço foi o assinantes(subscribers). Eu escolhi essa visão, pois a maior parte do faturamento vem esses clientes, sendo assim, tem-se que melhorar cada vez mais o serviço para atender esses clientes, por exemplo, criando novas estações, promoções, engajamento, etc.
- Foi respondido qual as estações mais populares por tipo de cliente. Eu escolhi essa visão, pois através dessa informação é possível melhorar o processo de atendimento, disponibilizar mais bicicletas, etc. para que os clientes fiquem cada vez mais satisfeitos com o serviço prestado.

Data Wrangling (Limpeza de Dados)

Agora é a sua vez de explorar os dados. Os [dados abertos](http://www.bayareabikeshare.com/open-data) (<http://www.bayareabikeshare.com/open-data>) do Ano 1 e do Ano 2 da página Bay Area Bike Share já foram fornecidos com os materiais do projeto; você não precisa baixar nada extra. O dado vem em três partes: a primeira metade do Ano 1 (arquivos a partir de 201402), a segunda metade do Ano 1 (arquivos a partir de 201408) e todo o Ano 2 (arquivos a partir de 201508). Existem três arquivos de dados principais associados a cada parte: dados de viagem que mostram informações sobre cada viagem no sistema (*_trip_data.csv), informações sobre as estações no sistema (*_station_data.csv) e dados meteorológicos diários para cada cidade no sistema (*_weather_data.csv).

Ao lidar com muitos dados, pode ser útil começar trabalhando com apenas uma amostra dos dados. Desta forma, será muito mais fácil verificar se nossos passos da limpeza de dados (Data Wrangling) estão funcionando, pois nosso código demorará menos tempo para ser concluído. Uma vez que estamos satisfeitos com a forma como funcionam as coisas, podemos configurar o processo para trabalhar no conjunto de dados como um todo.

Uma vez que a maior parte dos dados está contida na informação de viagem, devemos segmentar a procura de um subconjunto dos dados da viagem para nos ajudar a seguir em frente. Você começará olhando apenas o primeiro mês dos dados da viagem de bicicleta, de 2013-08-29 a 2013-09-30. O código abaixo selecionará os dados da primeira metade do primeiro ano, então escreverá o valor do primeiro mês de dados para um arquivo de saída. Este código explora o fato de que os dados são classificados por data (note que os dois primeiros dias são classificados por tempo de viagem, em vez de serem completamente cronológicos).

Primeiro, carregue todos os pacotes e funções que você usará em sua análise executando a primeira célula de código abaixo. Em seguida, execute a segunda célula de código para ler um subconjunto do primeiro arquivo de dados de viagem e escrever um novo arquivo contendo apenas o subconjunto em que inicialmente estamos interessados.

Dica: Você pode executar uma célula de código ou renderizar um texto em Markdown clicando na célula e usando o atalho do teclado **Shift + Enter** ou **Shift + Return**. Alternativamente, uma célula de código pode ser executada usando o botão **Play** na barra de ferramentas (a cima no IPython Notebook) depois de selecioná-la. Enquanto a célula estiver em execução, você verá um asterisco na mensagem à esquerda da célula, ou seja, In [*]:. O asterisco mudará para um número para mostrar que a execução foi concluída, Ex: In [1]. Se houver saída, ele aparecerá como Out [1]:, com um número apropriado para coincidir com o número de "In".

In [119]:

```
# Importa todas as bibliotecas necessárias
%matplotlib inline
import csv
from datetime import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from babs_datacheck import question_3
from babs_visualizations import usage_stats, usage_plot
from IPython.display import display
```

In [120]:

```
# definição dos arquivos
file_in = '201402_trip_data.csv'
file_out = '201309_trip_data.csv'

with open(file_out, 'wb') as f_out, open(file_in, 'r') as f_in:
    # configura o leitor de csv
    in_reader = csv.reader(f_in)
    out_writer = csv.writer(f_out)

    # escreve os dados no arquivo de saída até que a data limite seja atingida
    while True:
        datarow = next(in_reader)
        # data de início de das viagens na terceira coluna no formato 'm/d/yyyy HH:MM'
        if datarow[2][:9] == '10/1/2013':
            break
        out_writer.writerow(datarow)
```

Condensando os Dados de Viagem

O primeiro passo é analisar a estrutura do conjunto de dados para ver se há alguma limpeza de dados que devemos realizar. A célula abaixo irá ler o arquivo de dados amostrado que você criou na célula anterior. Você deve imprimir as primeiras linhas da tabela.

In [121]:

```
sample_data = pd.read_csv('201309_trip_data.csv')

# TODO: escreva o código para visualizar as primeiras linhas
sample_data.head()
```

Out[121]:

	Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #
0	4576	63	8/29/2013 14:13	South Van Ness at Market	66	8/29/2013 14:14	South Van Ness at Market	66	520
1	4607	70	8/29/2013 14:42	San Jose City Hall	10	8/29/2013 14:43	San Jose City Hall	10	661
2	4130	71	8/29/2013 10:16	Mountain View City Hall	27	8/29/2013 10:17	Mountain View City Hall	27	48
3	4251	77	8/29/2013 11:29	San Jose City Hall	10	8/29/2013 11:30	San Jose City Hall	10	26
4	4299	83	8/29/2013 12:02	South Van Ness at Market	66	8/29/2013 12:04	Market at 10th	67	319

Nesta exploração, vamos nos concentrar nos fatores nos dados da viagem que afetam o número de viagens realizadas. Vamos focar em algumas colunas selecionadas: a duração da viagem (trip duration), hora de início (start time), terminal inicial (start terminal), terminal final (end terminal) e tipo de assinatura. O tempo de início será dividido em componentes de ano, mês e hora. Também adicionaremos uma coluna para o dia da semana e resumiremos o terminal inicial e final para ser a *cidade* de início e fim.

Vamos primeiro abordar a última parte do processo de limpeza. Execute a célula de código abaixo para ver como as informações da estação estão estruturadas e observe como o código criará o mapeamento estação-cidade. Observe que o mapeamento da estação está configurado como uma função, `create_station_mapping()`. Uma vez que é possível que mais estações sejam adicionadas ou removidas ao longo do tempo, esta função nos permitirá combinar as informações da estação em todas as três partes dos nossos dados quando estivermos prontos para explorar tudo.

In [122]:

```
# Mostra as primeiras linhas do arquivo de dados das estações
station_info = pd.read_csv('201402_station_data.csv')
station_info.head()
```

Out[122]:

	station_id	name	lat	long	dockcount	landmark	installation
0	2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose	8/6/2013
1	3	San Jose Civic Center	37.330698	-121.888979	15	San Jose	8/5/2013
2	4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose	8/6/2013
3	5	Adobe on Almaden	37.331415	-121.893200	19	San Jose	8/5/2013
4	6	San Pedro Square	37.336721	-121.894074	15	San Jose	8/7/2013

Preencha a função abaixo de forma que a função retorne um mapeamento entre o id da estação (station_id) e a cidade em que ela se encontra (landmark).

In [123]:

```
# esta função será usada mais tarde para criar o mapeamento entre station e cidade
def create_station_mapping(station_data):
    """
    Cria um mapeamento (também conhecido como de-para) entre a estação
    e a cidade
    """
    # TODO: Inicie esta variável de maneira correta.
    station_map = {}
    for data_file in station_data:
        with open(data_file, 'r') as f_in:
            # configura o objeto csv reader - note que está sendo usado o DictReader,
            # que usa a primeira linha do arquivo como cabeçalho e cria as chaves
            # do dicionário com estes valores.
            weather_reader = csv.DictReader(f_in)
            for row in weather_reader:
                station_map[int(row['station_id'])] = row['landmark']
    return station_map
```

Você pode agora usar o mapeamento para condensar as viagens para as colunas selecionadas acima. Isto acontecerá na função abaixo `summarise_data()`. Nela o módulo `datetime` é usado para fazer o **parse** do tempo (timestamp) em formato de strings no arquivo original para um objeto usando a função `strptime`. Este objeto permitirá a conversão para outros formatos de datas usando a função `strftime`. O objeto possui também outras funções que facilitam a manipulação dos dados. Veja [este tutorial](http://usandopython.com.br/manipulando-data-hora-python-datetime/) (<http://usandopython.com.br/manipulando-data-hora-python-datetime/>) para entender um pouco melhor como trabalhar com a biblioteca.

Você precisa concluir duas tarefas para completar a função `summarise_data()`. Inicialmente, você deverá realizar a operação de converter a duração das viagens de segundos para minutos. Esta é muito fácil, pois existem 60 segundos em um minuto!

Na sequência, você deve criar colunas para o ano, mês, hora e dia da semana. Verifique o tutorial acima ou a [documentação para o objeto de datetime no módulo datetime](https://docs.python.org/2/library/datetime.html#datetime-objects) (<https://docs.python.org/2/library/datetime.html#datetime-objects>).

TODO: Encontre os atributos e métodos necessários para poder completar o código abaixo

Dica: Você pode abrir uma nova caixa para testar um pedaço do código ou verificar uma variável que seja global. Caso ela esteja dentro da função, você também pode usar o comando `print()` para imprimi-la e ajudar no Debug.

In [124]:

```
def summarise_data(trip_in, station_data, trip_out):
    """
    Esta função recebe informações de viagem e estação e produz um novo
    arquivo de dados com um resumo condensado das principais informações de viagem. Os
    argumentos trip_in e station_data serão listas de arquivos de dados para
    as informações da viagem e da estação enquanto trip_out especifica o local
    para o qual os dados sumarizados serão escritos.
    """

    # gera o dicionário de mapeamento entre estações e cidades
    station_map = create_station_mapping(station_data)

    with open(trip_out, 'wb') as f_out:
        # configura o objeto de escrita de csv
        out_colnames = ['duration', 'start_date', 'start_year',
                        'start_month', 'start_hour', 'weekday',
                        'start_city', 'end_city', 'subscription_type']
        trip_writer = csv.DictWriter(f_out, fieldnames = out_colnames)
        trip_writer.writeheader()

        for data_file in trip_in:
            with open(data_file, 'r') as f_in:
                # configura o leitor do csv
                trip_reader = csv.DictReader(f_in)

                # processa cada linha lendo uma a uma
                for row in trip_reader:
                    new_point = {}

                    # converte a duração de segundos para minutos.
                    ### TODO: Pergunta 3a: Adicione uma operação matemática ###
                    ### para converter a duração de segundos para minutos. ###
                    new_point['duration'] = int(row['Duration']) / 60.0
```

```

# reformate strings com datas para múltiplas colunas
### TODO: Pergunta 3b: Preencha os __ abaixo para criar os
##

### campos esperados nas colunas (olhe pelo nome da coluna) ###
trip_date = datetime.strptime(row['Start Date'], '%m/%d/%Y %H:%M')
new_point['start_date'] = trip_date.date()
new_point['start_year'] = trip_date.year
new_point['start_month'] = trip_date.month
new_point['start_hour'] = trip_date.hour
new_point['weekday'] = trip_date.weekday()

# TODO: mapeia o terminal de inicio e fim com o a cidade de inicio
e fim

new_point['start_city'] = station_map[int(row['Start Terminal'])]
new_point['end_city'] = station_map[int(row['End Terminal'])]

# TODO: existem dois nomes diferentes para o mesmo campo. Trate cad
a um deles.

if 'Subscription Type' in row:
    new_point['subscription_type'] = row['Subscription Type']
else:
    new_point['subscription_type'] = row['Subscriber Type']

# escreve a informação processada para o arquivo de saída.
trip_writer.writerow(new_point)

```

Pergunta 3:

Execute o bloco de código abaixo para chamar a função `summarise_data()` que você terminou na célula acima. Ela usará os dados contidos nos arquivos listados nas variáveis `trip_in` e `station_data` e escreverá um novo arquivo no local especificado na variável `trip_out`. Se você executou a limpeza de dados corretamente, o bloco de código abaixo imprimirá as primeiras linhas do DataFrame e uma mensagem que verificando se as contagens de dados estão corretas.

In [125]:

```

# processe os dados usando a função criada acima
station_data = ['201402_station_data.csv']
trip_in = ['201309_trip_data.csv']
trip_out = '201309_trip_summary.csv'
summarise_data(trip_in, station_data, trip_out)

```

In [126]:

```
# Carregue os dados novamente mostrando os dados
## TODO: Complete o código para leitura dos dados no arquivo criado na função acima
sample_data = pd.read_csv('201309_trip_summary.csv')
display(sample_data.head())
```

	duration	start_date	start_year	start_month	start_hour	weekday	start_city	end_
0	1.050000	2013-08-29	2013	8	14	3	San Francisco	San Fran
1	1.166667	2013-08-29	2013	8	14	3	San Jose	San ,
2	1.183333	2013-08-29	2013	8	10	3	Mountain View	Mour View
3	1.283333	2013-08-29	2013	8	11	3	San Jose	San ,
4	1.383333	2013-08-29	2013	8	12	3	San Francisco	San Fran

In [127]:

```
# Verifica o DataFrame contando o número de pontos de dados com as características de
# tempo corretas.
question_3(sample_data)
```

Todas as contagens estão como esperadas.

Dica: se você salvar um notebook do jupyter, a saída dos blocos de código em execução também será salva. No entanto, o estado do seu arquivo será reiniciado uma vez que uma nova sessão será iniciada. Certifique-se de que você execute todos os blocos de código necessários da sessão anterior para restabelecer variáveis e funções antes de continuar de onde você deixou na última vez.

Análise Exploratória de Dados

Agora que você tem alguns dados salvos em um arquivo, vejamos algumas tendências iniciais nos dados. Algum código já foi escrito para você no script [babs_visualizations.py](#) ([babs_visualizations.py](#)) para ajudar a resumir e visualizar os dados; Isso foi importado como as funções `usage_stats()` e `usage_plot()`. Nesta seção, vamos percorrer algumas das coisas que você pode fazer com as funções, e você usará as funções para você mesmo na última parte do projeto. Primeiro, execute a seguinte célula para carregar os dados. Depois preencha a célula abaixo com os comandos para verificar os dados básicos sobre os dados.

In [128]:

```
trip_data = pd.read_csv('201309_trip_summary.csv')
```

In [129]:

```
# TODO: preencha os campos com os dados de acordo com o print
print('Existem {:d} pontos no conjunto de dados'.format(len(trip_data)))
print('A duração média das viagens foi de {:.2f} minutos'.format(trip_data.mean()['duration']))
print('A mediana das durações das viagens foi de {:.2f} minutos'.format(trip_data.median()['duration']))

# TODO: verificando os quartis
duration_qtiles = trip_data['duration'].quantile([.25, .5, .75]).as_matrix()
print('25% das viagens foram mais curtas do que {:.2f} minutos'.format(duration_qtiles[0]))
print('25% das viagens foram mais compridas do que {:.2f} minutos'.format(duration_qtiles[2]))
```

Existem 27345 pontos no conjunto de dados
A duração média das viagens foi de 27.60 minutos
A mediana das durações das viagens foi de 10.72 minutos
25% das viagens foram mais curtas do que 6.82 minutos
25% das viagens foram mais compridas do que 17.28 minutos

In [130]:

```
# execute este campo para verificar os seu processamento acima.
usage_stats(trip_data)
```

Existem 27345 pontos no conjunto de dados
A duração média das viagens foi de 27.60 minutos
A mediana das durações das viagens foi de 10.72 minutos
25% das viagens foram mais curtas do que 6.82 minutos
25% das viagens foram mais compridas do que 17.28 minutos

Out[130]:

```
array([ 6.81666667, 10.71666667, 17.28333333])
```

Você deve ver que há mais de 27.000 viagens no primeiro mês e que a duração média da viagem é maior do que a duração mediana da viagem (o ponto em que 50% das viagens são mais curtas e 50% são mais longas). Na verdade, a média é maior que as durações de 75% das viagens mais curtas. Isso será interessante para ver mais adiante.

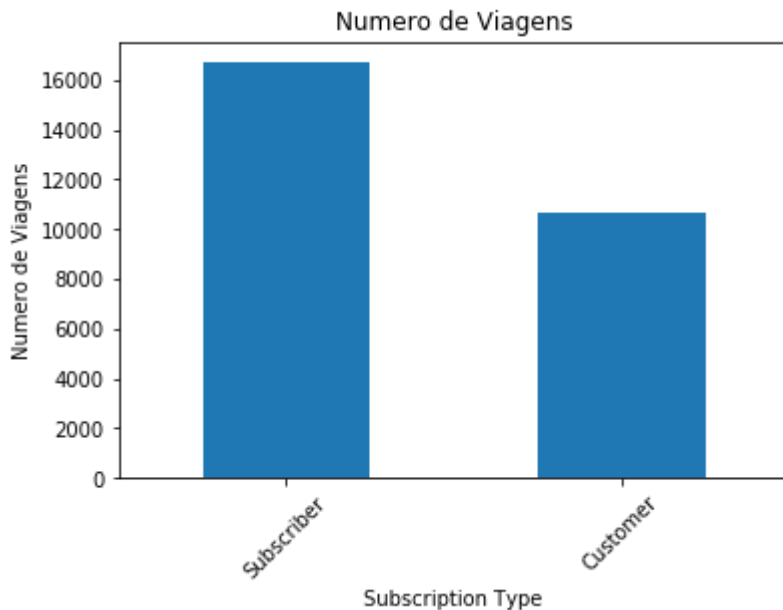
Vamos começar a ver como essas viagens são divididas por tipo de inscrição. Uma maneira fácil de construir uma intuição sobre os dados é traçá-los.

Lembre-se que o Pandas possui maneiras de plotar os gráficos diretamente de um DataFrame. Para cada tipo de dados/análises se pode usar um tipo diferente de gráfico mais apropriado para a análise que se está fazendo.

Na caixa abaixo, faça um gráfico de viagens x tipo de subscrição do tipo barras.

In [182]:

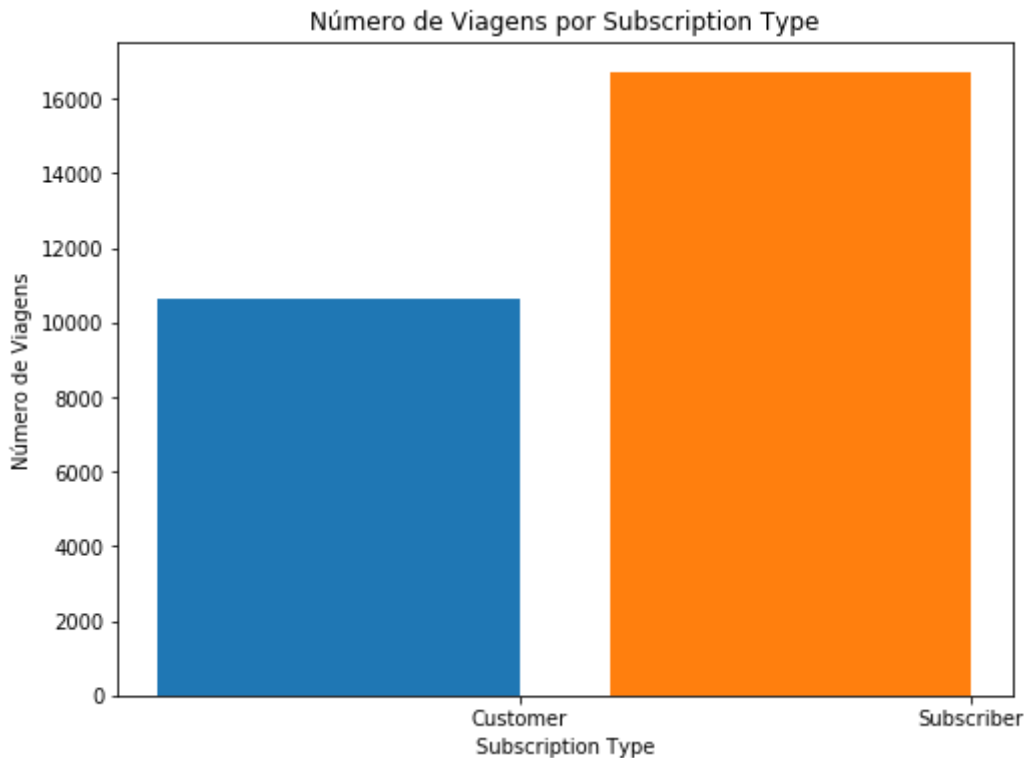
```
# TODO: plote um gráfico de barras que mostre quantidade de viagens por subscription_type  
# Lembrando que quando o comando .plot é usado, se pode escolher o tipo de gráfico usando  
# o parâmetro kind. Ex: plot(kind='bar')  
trip_data_by_subscription_type = trip_data['subscription_type'].value_counts()  
trip_data_by_subscription_type.plot(kind='bar',rot=45)  
plt.title('Numero de Viagens')  
plt.ylabel('Numero de Viagens')  
plt.xlabel('Subscription Type')  
plt.show()
```



Para que você possa conferir se os seus gráficos estão corretos, usaremos a função `use_plot()`. O segundo argumento da função nos permite contar as viagens em uma variável selecionada, exibindo as informações em um gráfico. A expressão abaixo mostrará como deve ter ficado o seu gráfico acima.

In [132]:

```
# como o seu gráfico deve ficar. Descomente a linha abaixo caso queira rodar este comando  
usage_plot(trip_data, 'subscription_type')
```



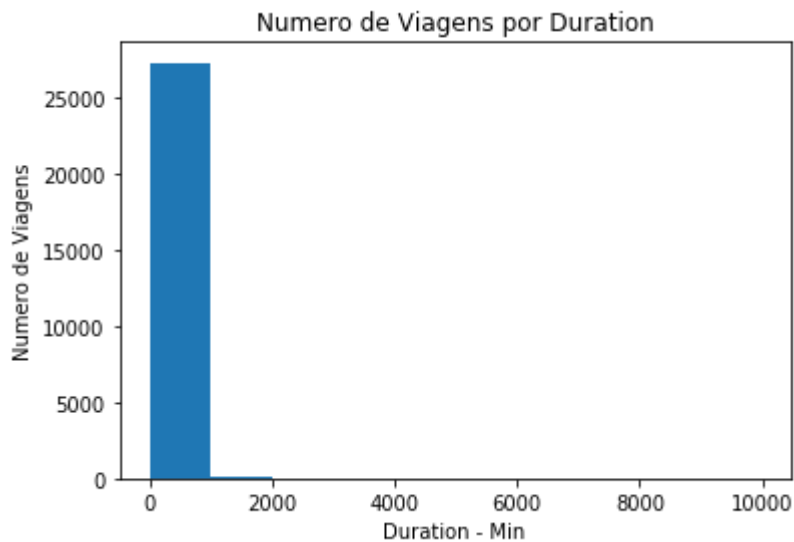
Nota: Perceba que provavelmente o seu gráfico não ficou exatamente igual, principalmente pelo título e pelo nome dos eixos. Lembre-se, estes são detalhes mas fazem toda a diferença quando você for apresentar os gráficos que você analisou. Neste Nanodegree não focaremos nestas questões, mas tenha em mente que ter os gráficos acertados é de extrema importância.

Parece que existe 50% mais viagens feitas por assinantes (subscribers) no primeiro mês do que outro tipos de consumidores. Vamos tentar uma outra variável. Como é a distribuição da duração das viagens (trip duration)?

In [134]:

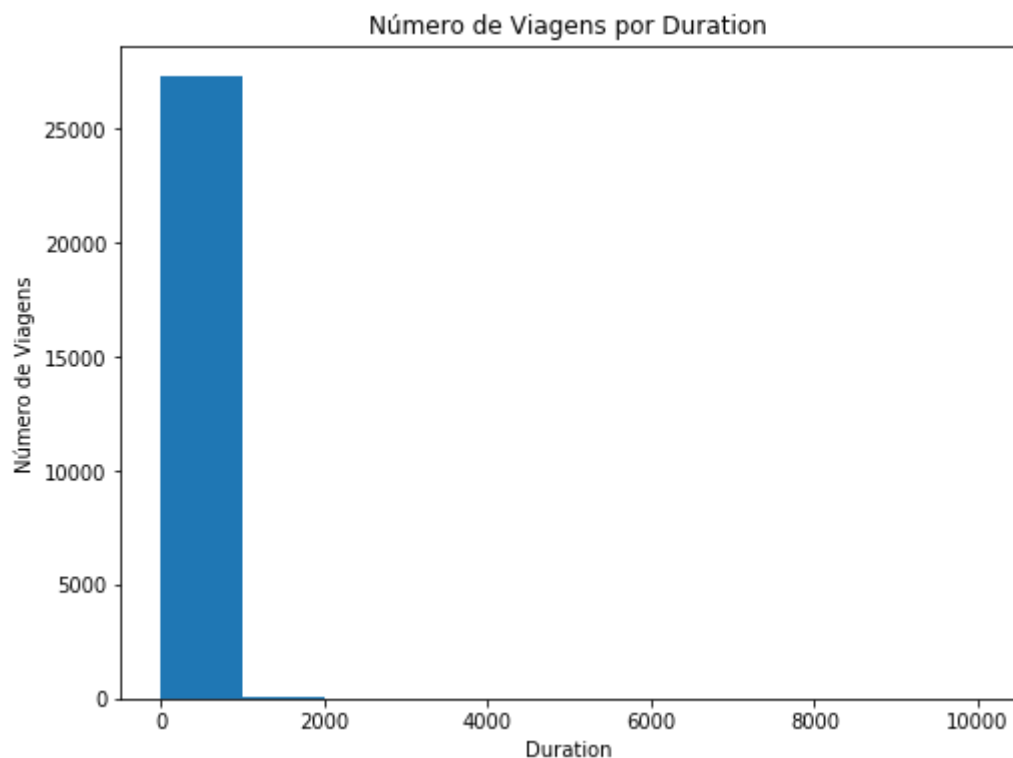
```
# TODO: Faça um gráfico baseado nas durações
durations = trip_data['duration']

plt.hist(durations)
plt.title('Numero de Viagens por Duration')
plt.ylabel('Numero de Viagens')
plt.xlabel('Duration - Min')
plt.show()
```



In [135]:

```
# rode este comando abaixo caso esteja em dúvida quanto ao resultado esperado
usage_plot(trip_data, 'duration')
```



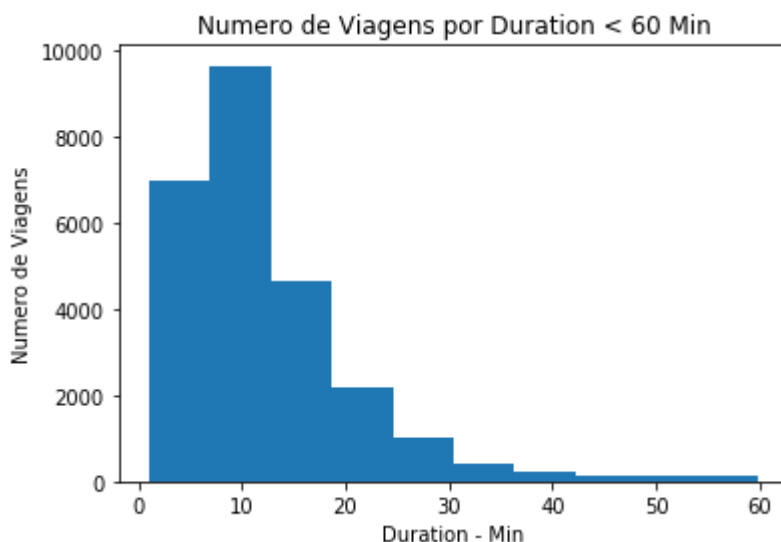
Parece muito estranho, não é? Dê uma olhada nos valores de duração no eixo x. A maioria dos passeios deve ser de 30 minutos ou menos, uma vez que há taxas de excesso de tempo extra em uma única viagem. A primeira barra abrange durações de até 1000 minutos, ou mais de 16 horas. Com base nas estatísticas que obtivemos do `use_stats()`, deveríamos ter esperado algumas viagens com durações muito longas que levem a média a ser muito superior à mediana: o gráfico mostra isso de forma dramática, mas inútil.

Ao explorar os dados, muitas vezes você precisará trabalhar com os parâmetros da função de visualização para facilitar a compreensão dos dados. É aqui que os filtros vão ajudar você. Começemos por limitar as viagens de menos de 60 minutos.

In [136]:

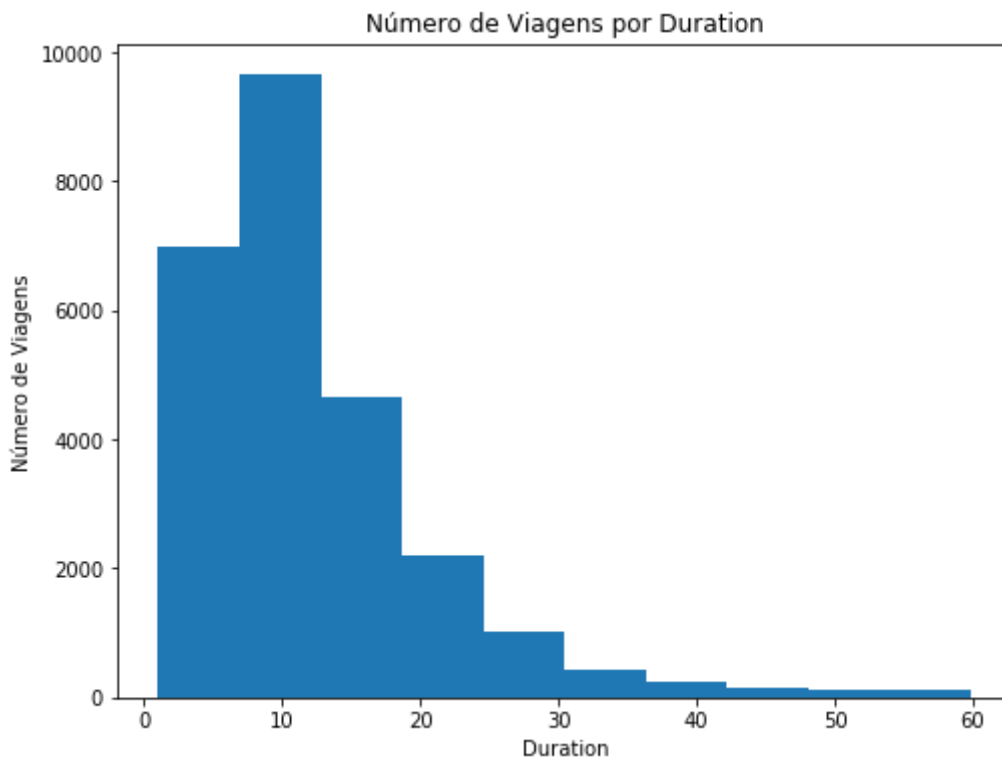
```
# TODO: faça um gráfico de barras para os dados com duração inferior a 60 minutos
```

```
durations_60 = trip_data[trip_data['duration'] < 60]
durations_60['duration'].plot(kind='hist')
plt.title('Numero de Viagens por Duration < 60 Min')
plt.ylabel('Numero de Viagens')
plt.xlabel('Duration - Min')
plt.show()
```



In [137]:

```
# descomente a linha abaixo para verificar o gráfico esperado.  
usage_plot(trip_data, 'duration', ['duration < 60'])
```



Isso está bem melhor! Você pode ver que a maioria das viagens têm menos de 30 minutos de duração, mas que você pode fazer mais para melhorar a apresentação. Uma vez que a duração mínima não é 0, a barra da esquerda está ligeiramente acima de 0. Nós queremos saber onde existe um limite perto dos 30 minutos, então ficará mais agradável se tivermos tamanhos de intervalos (bin sizes) e limites dos intervalos que correspondam a alguns minutos.

Felizmente, o Pandas e o Matplotlib te dão a opção de resolver estes problemas. Uma das maneiras de fazê-lo é definindo qual o intervalo no eixo x (parâmetro range) e quantos intervalos desejamos (bins).

No campo abaixo, faça o ajuste do gráfico para que os limites das barras se encontrem nas extremidades e que as barras tenham tamanho 5 (0, 5, 10, 15, etc). Se precisar, use a [documentação](http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.hist.html#matplotlib.axes.Axes.hist) (http://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.hist.html#matplotlib.axes.Axes.hist).

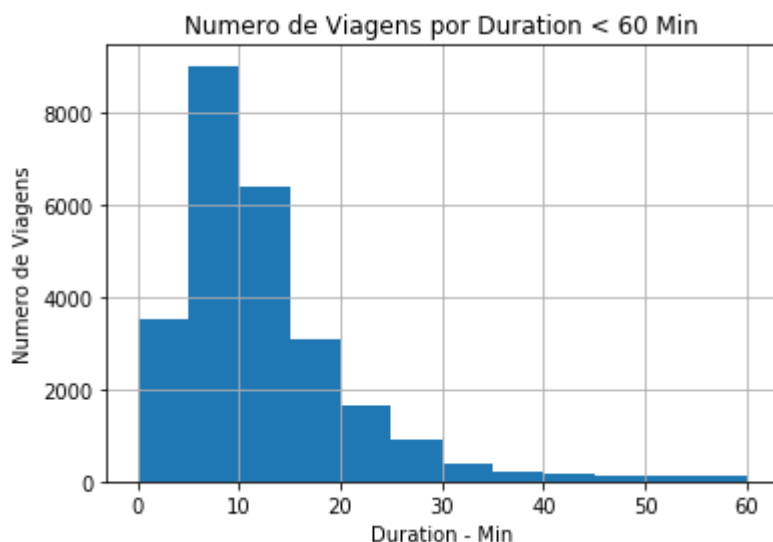
In [138]:

```
# faça o gráfico ajustado que começará no 0 e terá o bin size de 5
```

```

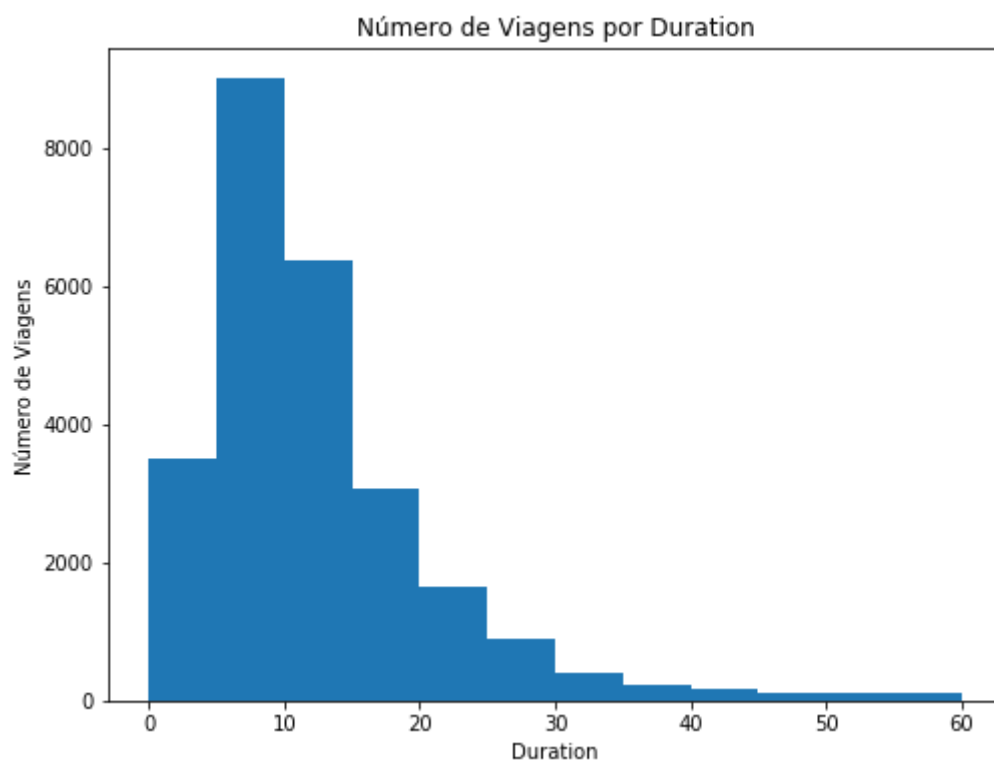
durations_60 = trip_data[trip_data['duration'] < 60]
durations_60['duration'].hist(bins=int(60/5), range=(0,60))
plt.title('Numero de Viagens por Duration < 60 Min')
plt.ylabel('Numero de Viagens')
plt.xlabel('Duration - Min')
plt.show()

```



In [139]:

```
# rode esta linha para verificar como deve ficar o seu gráfico
usage_plot(trip_data, 'duration', ['duration < 60'], boundary = 0, bin_width = 5)
```



Pequenos ajustes como este podem ser pequenos mas fazem toda a diferença na entrega de um trabalho de qualidade e com atenção aos detalhes.

Pergunta 4

Analise o histograma do exercício anterior e responda:

Qual o intervalo de duração com maior quantidade de viagens?

Resposta: Entre o intervalo 5 e 10

Pergunta 4.1

Com base na sua resposta da pergunta anterior, conte quantas viagens existem para cada origem e destino desse intervalo de tempo. Monte uma tabela que demonstre isso.

Qual origem e destino tem a maior quantidade de viagens? Aproximadamente quantas viagens foram feitas nesta faixa de tempo?

Resposta: Substitua este texto pela sua resposta!

In [183]:

```
# origem e destino para intervalo entre 5 e 10
durations_5_10 = trip_data[(trip_data.duration >= 5) & (trip_data.duration <= 10)]
durations_5_10.head()

total_viagens_5_10 = len(durations_5_10)
```

In [184]:

```
# qual origem e destino tem a maior quantifade de viagens
origem_qtde_viagens = pd.value_counts(durations_5_10['start_city'])
origem_max_viagens = origem_qtde_viagens.argmax()

destino_qtde_viagens = pd.value_counts(durations_5_10['end_city'])
destino_max_viagens = destino_qtde_viagens.argmax()
```

In [185]:

```
#Resposta - Pergunta 4.1
print 'Total de viagens feitas no intervalo de 5 a 10 min: {0}'.format(total_viagens_5_10)
print 'A origem com o maior número de viagens ({0}) foi : {1}'.format(origem_qtde_viagens.max(),origem_max_viagens)
print 'O destino com o maior número de viagens ({0}) foi : {1}'.format(destino_qtde_viagens.max(),destino_max_viagens)
```

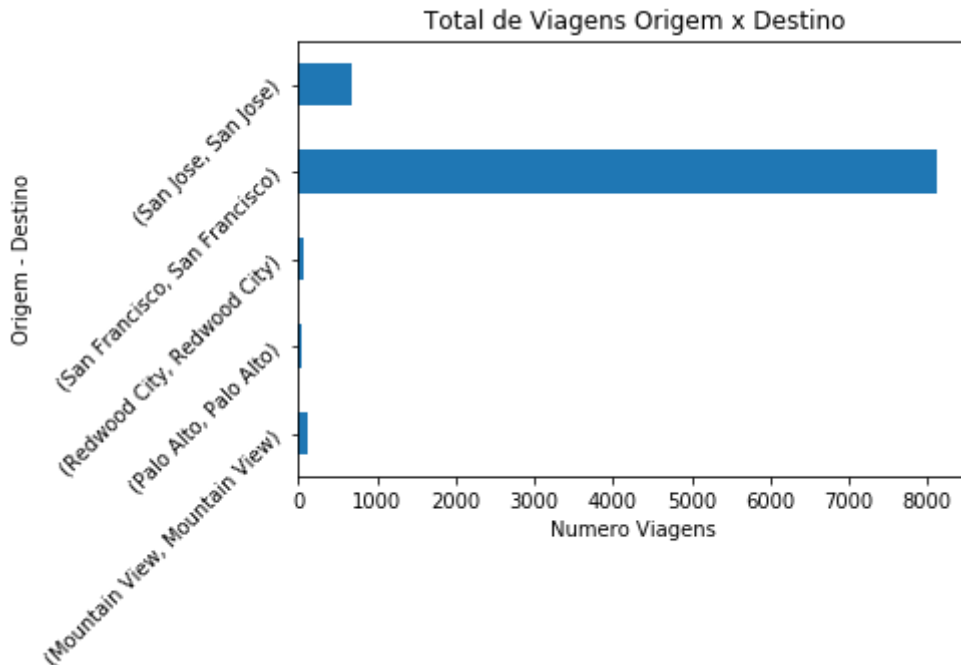
Total de viagens feitas no intervalo de 5 a 10 min: 9025

A origem com o maior número de viagens (8129) foi : San Francisco

O destino com o maior número de viagens (8129) foi : San Francisco

In [186]:

```
# grafico de quantidade de viagens por origem e destino
graph_orig_dest = durations_5_10.groupby(['start_city', 'end_city']).size()
graph_orig_dest.plot.barh(rot=45)
plt.title('Total de Viagens Origem x Destino')
plt.ylabel('Origem - Destino')
plt.xlabel('Numero Viagens')
plt.show()
```



Fazendo suas Próprias Análises

Agora que você fez alguma exploração em uma pequena amostra do conjunto de dados, é hora de avançar e reunir todos os dados em um único arquivo e ver quais tendências você pode encontrar. O código abaixo usará a mesma função `summarise_data()` para processar dados. Depois de executar a célula abaixo, você terá processado todos os dados em um único arquivo de dados. Observe que a função não exibirá qualquer saída enquanto ele é executado, e isso pode demorar um pouco para ser concluído, pois você tem muito mais dados do que a amostra com a qual você trabalhou.

In [187]:

```
station_data = ['201402_station_data.csv',
                '201408_station_data.csv',
                '201508_station_data.csv' ]
trip_in = ['201402_trip_data.csv',
           '201408_trip_data.csv',
           '201508_trip_data.csv' ]
trip_out = 'babs_y1_y2_summary.csv'

# Esta função irá ler as informações das estações e das viagens
# e escreverá um arquivo processado com o nome trip_out
summarise_data(trip_in, station_data, trip_out)
```

Já que a função `summarise_data()` escreveu um arquivo de saída, a célula acima não precisa ser rodada novamente mesmo que este notebook seja fechado e uma nova sessão seja criada. Você pode simplesmente ler os dados novamente e fazer a exploração deste ponto (não esqueça de executar a parte das funções no começo do notebook caso esteja em uma nova sessão)

In [188]:

```
trip_data = pd.read_csv('babs_y1_y2_summary.csv')
display(trip_data.head())
```

	duration	start_date	start_year	start_month	start_hour	weekday	start_city	end_
0	1.050000	2013-08-29	2013	8	14	3	San Francisco	San Francisco
1	1.166667	2013-08-29	2013	8	14	3	San Jose	San Jose
2	1.183333	2013-08-29	2013	8	10	3	Mountain View	Mountain View
3	1.283333	2013-08-29	2013	8	11	3	San Jose	San Jose
4	1.383333	2013-08-29	2013	8	12	3	San Francisco	San Francisco

Agora é a SUA vez de fazer a exploração do dataset (do conjunto de dados) completo.

Aproveite para fazer filtros nos dados e tentar encontrar padrões nos dados.

Explore algumas variáveis diferentes usando o mesmo racional acima e tome nota de algumas tendências que você encontra. Sinta-se livre para criar células adicionais se quiser explorar o conjunto de dados de outras maneiras ou de várias maneiras.

Dica: para adicionar células adicionais a um notebook, você pode usar as opções "Inserir célula acima" (Insert Cell Above) e "Inserir célula abaixo" (Insert Cell Below) na barra de menu acima. Há também um ícone na barra de ferramentas para adicionar novas células, com ícones adicionais para mover as células para cima e para baixo do documento. Por padrão, as novas células são do tipo de código; Você também pode especificar o tipo de célula (por exemplo, Código ou Markdown) das células selecionadas no menu Cell ou no menu dropdown na barra de ferramentas.

Um feito com suas explorações, copie as duas visualizações que você achou mais interessantes nas células abaixo e responda as seguintes perguntas com algumas frases descrevendo o que você encontrou e por que você selecionou os números. Certifique-se de que você ajusta o número de caixas ou os limites da bandeja para que efetivamente transmitam os resultados dos dados. Sinta-se livre para complementar isso com quaisquer números adicionais gerados a partir de `use_stats()` ou coloque visualizações múltiplas para suportar suas observações.

Para ver alguns outros tipos de gráficos que o matplotlib (padrão do Pandas) possui, leia [este artigo](https://www.labri.fr/perso/nrougier/teaching/matplotlib/#other-types-of-plots) (<https://www.labri.fr/perso/nrougier/teaching/matplotlib/#other-types-of-plots>).

Para entender um pouco mais como e quais gráficos podem ser úteis, leia [este documento](https://www.tableau.com/sites/default/files/media/Whitepapers/which_chart_v6_ptb.pdf) (https://www.tableau.com/sites/default/files/media/Whitepapers/which_chart_v6_ptb.pdf). Ele lhe dará um pouco de idéia de como mostrar os dados de forma mais acertada

Pergunta 5a

Explore os dados e faça um gráfico que demonstre alguma particularidade dos dados:

In [189]:

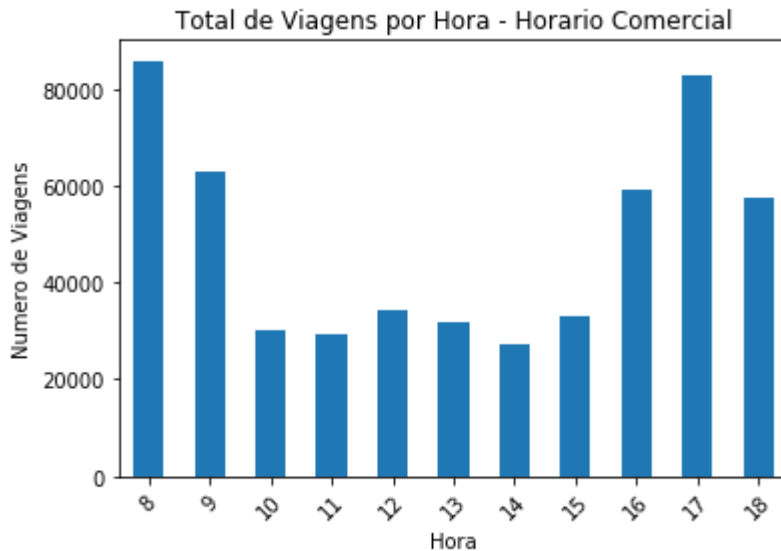
```
# Analisando dados
trip_data.head()
```

Out[189]:

	duration	start_date	start_year	start_month	start_hour	weekday	start_city	end
0	1.050000	2013-08-29	2013	8	14	3	San Francisco	San Francisco
1	1.166667	2013-08-29	2013	8	14	3	San Jose	San Jose
2	1.183333	2013-08-29	2013	8	10	3	Mountain View	Mountain View
3	1.283333	2013-08-29	2013	8	11	3	San Jose	San Jose
4	1.383333	2013-08-29	2013	8	12	3	San Francisco	San Francisco

In [192]:

```
# Gráfico quantidade de viagens por hora dentro do horário comercial 8 até 18
trip_data_filter = trip_data[(trip_data.start_hour >= 8) & (trip_data.start_hour <= 18)]
trip_data_filter.groupby(['start_hour']).size()
trip_data_filter.plot.bar(rot=45)
plt.title('Total de Viagens por Hora - Horário Comercial')
plt.ylabel('Numero de Viagens')
plt.xlabel('Hora')
plt.show()
```



O que é interessante na visualização acima? Por que você a selecionou?

Answer: Podemos identificar os horários de pico que os clientes utilizam o serviço, com esses dados é possível estabelecer os turnos de trabalhos dos funcionários para melhorar o atendimento dos clientes.

Pergunta 5b

Faça um gráfico que demonstre alguma particularidade dos dados: