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 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 Pyhon 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

- Slack: udacity-br.slack.com
- 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 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: Após a análise de quais informações são fornecidas nos dados, consegui pensar nas seguintes perguntas:

Questão 1: Qual é a estação com o menor numero médio de vagas? Se existir momentos onde não há vagas, é possível determinar os horários onde essa lotação ocorre? A causa dessa lotação é devido a não locação das bicicletas da estação ou por muitas devoluções nessa estação? Os clientes não devem encontrar uma estação sem diponibilidade de vaga. Se isso ocorrer, não há dados sobre quantos estão ficando sem vagas, visto que não há como saber o trajeto realizado (apenas ponto de partido e de devolução) através dos dados fornecidos.

Questão 2: Quais estações recebem mais devoluções de bicicletas do que retiradas? Por outro lado, quais estações recebem mais retiradas do que devoluções? Quantos utilizam o seviço para se deslocar entre as estações e quantos realizam a devolução na mesma estações onde retirou a bicicleta? Existindo esses défictis, é possível fazer um planejamento e controlar essee problemas.

Questão 3: Qual o tempo médio de utilização? A maioria dos usuários utlizam o serviço para um deslocamento curto ou longo? Com qual frequencia? Essa informação poderia ser comparada com a porcentagem de clientes casual e anual. Dependendo do resultado, é possível identificar potenciais clientes anuais que atualmente são do tipo casual.

Questão 4: Qual o número médio de locações realizadas em dias de chuva compardo a média dos dias sem chuva? Qaul a redução em porcentagem para dias chuvosos? Qual a porcentagem de dias chuvosos? Analisando essas duas informações, é possivél prever quanto está sendo perdido, em média, por conta dos dias chuvosos e assim planejar uma solução caso esse parâmetro mostre-se significancia.

> **Dica**: Se você clicar duas vezes nesta célula, você verá o texto se alterar removendo toda a formatação. Isso permite editar este bloco de texto. Este bloco de texto é escrito usando Markdown, que é uma forma de formatar texto usando cabeçalhos, links, itálico e muitas outras opções. Pressione **Shift** + **Enter** ou **Shift** + **Retorno** para voltar a mostrar o texto formatado.

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 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 de Tyler Field. Leia todo o relatório para responder à seguinte pergunta:

Pergunta 2

Que visualizações você acha que fornecem as idéias mais interessantes? Você é capaz de responder a uma das perguntas identificadas acima com base na análise de Tyler? Por que ou por que não?

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

Resposta :

As visualizações que fornecem as ideia mais interessantes são:

- Número de trajetos, separados em tipos de usuários, para os dias da semana e para as horas do dia. (Rides by Weekday e Rides by Hour)
- Número total de trajeto ao longo de todo o período dos dados, separados por tipo de usuário e com possibilidade de destacar período de tempos específicos como fins de semana, dias de semana, jogos esportivos importantes, condições climáticas, acontecimentos políticos, greves e feriados nacionais.

Considerando as perguntas "identificadas a cima" como sendo as perguntas formuladas por mim, sim, é possível responder a **Questão 2**, **Questão 3**, **Questão 4**, pois a base dos dados necessários para essas análises foram plotados pela análise de Taylor.

Visualização 1: When is Bay Area Bike Share used?

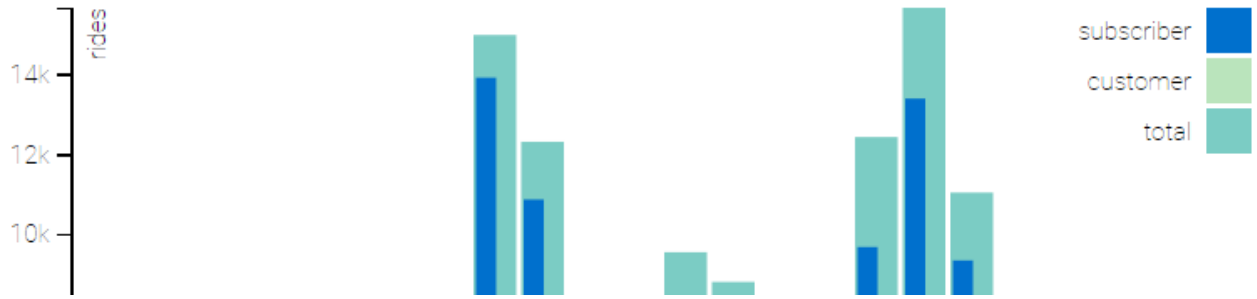
In [1]:

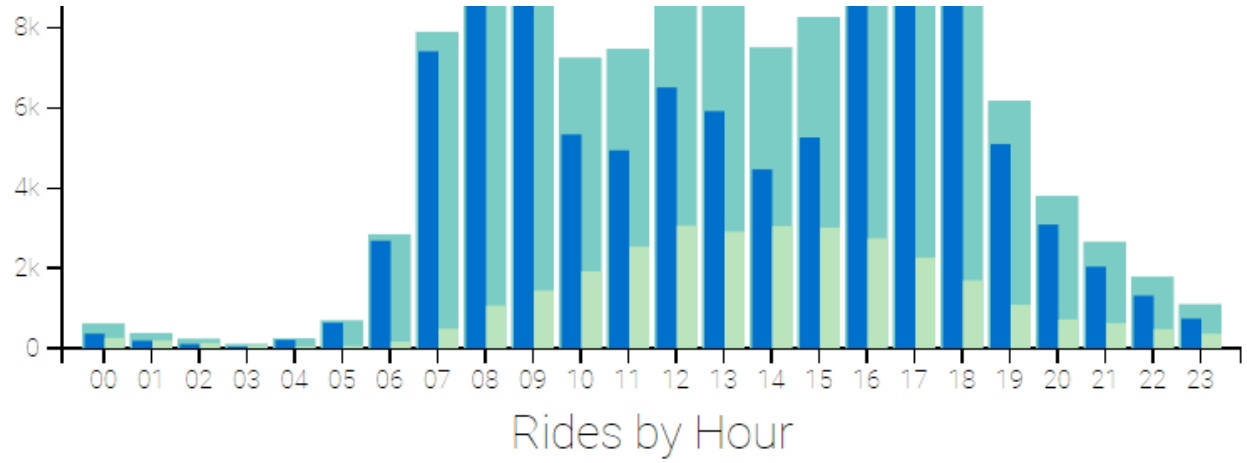
```
from IPython.display import Image
```

In [2]:

```
Image(filename = "C:/Users/User/Desktop/Fund. of Data Analysis I -  Udacity
/WEEK 4 - Final Project/Rides by hour - Taylor.png", width=700, height=700)
```

Out[2]:



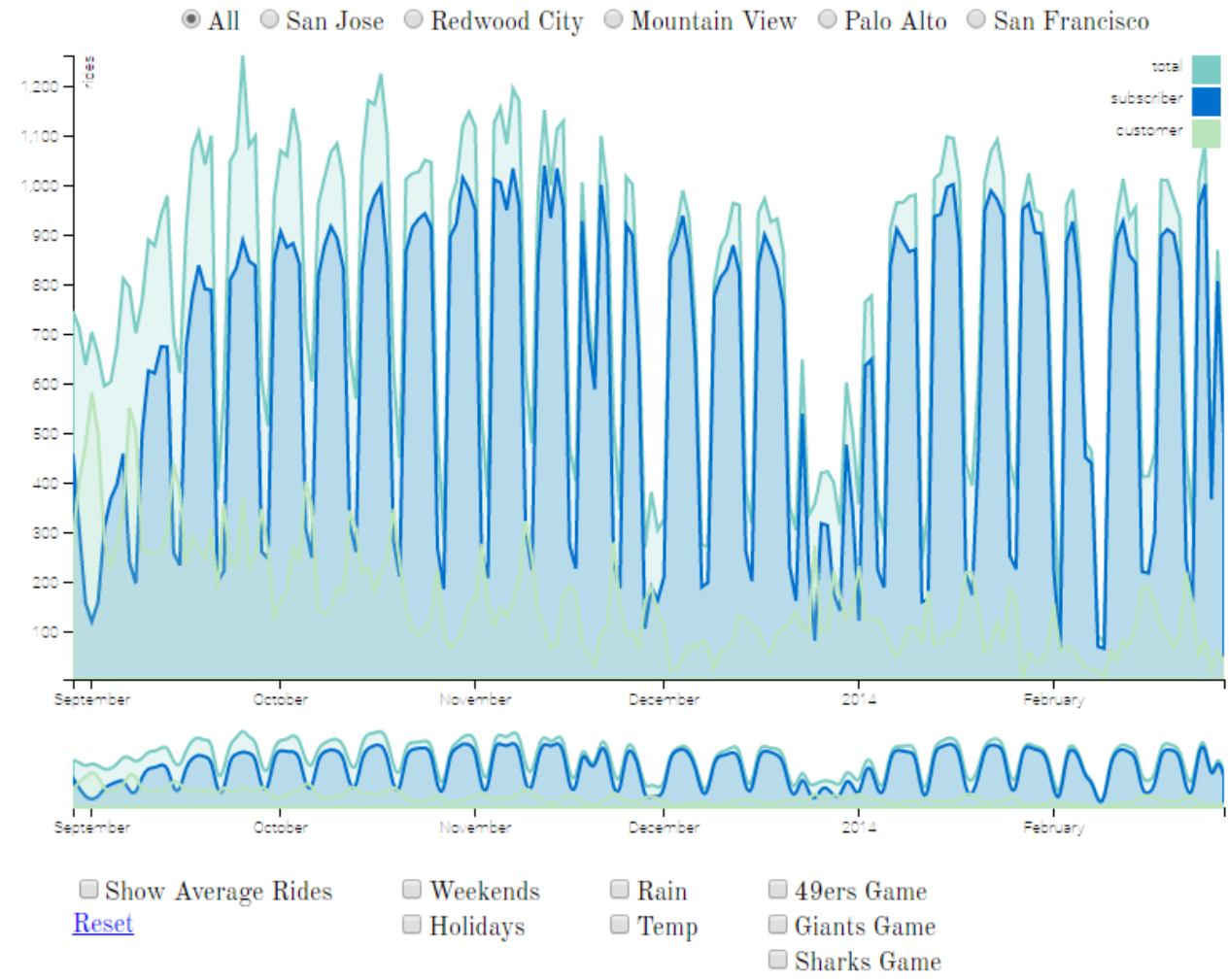


Visualização 2: Where do people ride Bike Share?

In [3]:

```
Image("C:/Users/User/Desktop/Fund. of Data Analysis I -  Udacity/WEEK 4 - Final Project/Interactive Chart - Taylor.png")
```

Out[3]:



Pergunta 2.1

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

Resposta : As minhas visualizacões respondem quando a Bay Area Bike Share é usada e quais os

Resposta: As minhas visualizações respondem quando a Bay Area Bike Share é usada e quais os locais que apresentam maior número de usuário. A escolha dessas perguntas é motivada pela importância de compreender o padrão dos usuários, podendo classifica-los, assim como Taylor nos mostrou, e localizar a estações de maior fluxo, visando aprofundar a análise nesses usuários para entender o que os motiva a utilizar o serviço, bem como entender o motivo pelo qual algumas estações quase não são utilizadas.

Data Wrangling (Limpeza de Dados)

Agora é a sua vez de explorar os dados. Os dados abertos 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 [4]:

```
# Importa todas as bibliotecas necessárias
%matplotlib inline
import csv
```

```
import csv
from datetime import datetime
import numpy as np
import pandas as pd

from babs_datacheck import question_3
from babs_visualizations import usage_stats, usage_plot
from IPython.display import display
```

OBS: Segue abaixo dois link que me ajudaram na análise

http://thfield.github.io/babs/index.html

https://review.udacity.com/#!/rubrics/1071/view

In [5]:

```
# definição dos arquivos

file_in  = '201402_trip_data.csv'
file_out = '201309_trip_data.csv' #Inicialmente é um arquivo em branco

#Abre o arquivo file_out para edição e o arquivo file_in para leitura
with open(file_out, 'w') 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 ating
ida
    while True:

        #For acess the next row of data
        datarow = next(in_reader)

        # data de início de das viagens na terceira coluna no formato 'm/d/
yyyy HH:MM' = sting
        #acessando a coluna da data[2] e coletando do inicio até a posição
8 (inclusa), ou seja, [:9]
        if datarow[2][:9] == '10/1/2013':
            break
        out_writer.writerow(datarow)
```

O arquivo '201309_trip_data.csv', antes vazio, agora contém os dados do primeiro mes da parte trip_data do primeiro ano.

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 [6]:

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

# TODO: escreva o código para visualizar as primeiras linhas
```

```
sample_data.head()
```

Out[6]:

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

Nesta exploração, vamos nos concentrar nos fatores dos 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 (subscription type). O start time **(NÃO SERIA START DATE??)** será dividido em componentes de ano, mês e hora **(DIVIDIDO EM ANO, MES, DIA 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 [7]:

```
# Mostra as primeiras linhas do arquivo de dados das estações
station_info = pd.read_csv('201402_station_data.csv')
station_info.head(10)
```

Out[7]:

	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

	station_id	name	lat	long	dockcount	landmark	installation
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
5	7	Paseo de San Antonio	37.333798	-121.886943	15	San Jose	8/7/2013
6	8	San Salvador at 1st	37.330165	-121.885831	15	San Jose	8/5/2013
7	9	Japantown	37.348742	-121.894715	15	San Jose	8/5/2013
8	10	San Jose City Hall	37.337391	-121.886995	15	San Jose	8/6/2013
9	11	MLK Library	37.335885	-121.885660	19	San Jose	8/6/2013

NOTA SOBRE OS DADOS DAS ESTAÇÕES: Embora as estações tenham sido instaladas antes de 8/29/13 (lançamento do sistema), nenhuma estação estava ativa até a data de lançamento. Portanto, para capturar com precisão a popularidade da estação, recomendamos ajustar todas as datas de instalação pré-lançamento em 29/8/13. *(Ainda não foi realizado)*

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 [8]:

```
# esta função será usada mais tarde para criar o mapeamento entre station e
cidade
def create_station_mapping(station_data):
    """
    Cria um mapeamento (tambémm 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 D
ictReader,
            # 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: #row irá percorrer todas as keys do
disct weather_reader
                station_id = row['station_id']
```

```
                city = row['landmark']
                station_map[station_id] = city
    return station_map
```

In [9]:

```
### JUST FOR TEST

#test = ['201402_station_data.csv']
#print create_station_mapping(test)
```

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 **p**arse do tempo (timestamp) em formato de strings no arquivo original para um objeto usando a função `strptime`. Este objeto permitirá a conversão para outros **f**ormatos de datas usando a função `strftime`. O objeto possui também outras funções que facilitam a manipulação dos dados. Veja este tutorial 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 .

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 [10]:

```
#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 (subscription type).
#O start time **(NÃO SERIA START DATE??)** será dividido em componentes de
ano, mês e hora **(DIVIDIDO EM ANO, MES, DIA 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.

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 par
a
    as informações da viagem e da estação enquanto trip_out especifica o lo
cal
    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, 'w') 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'] = float(row['Duration'])/float(60)

                    # reformate strings com datas para múltiplas colunas
                    ### TODO: Pergunta 3b: Preencha os __ abaixo para criar
os        ###
                    ### campos experados nas colunas (olhe pelo nome da col
na) ###
                    trip_date = datetime.strptime(row['Start Date'], '%m/%d
%Y %H:%M')
                    new_point['start_date']  = trip_date.strftime('%d/%m/%Y
')
                    #print '1) ', new_point['start_date']
                    new_point['start_year']  = trip_date.year
                    #print '2) ', new_point['start_year']
                    new_point['start_month'] = trip_date.month
                    #print '3) ', new_point['start_month']
                    new_point['start_hour']  = trip_date.strftime('%H')
                    #print '4) ', new_point['start_hour']
                    new_point['weekday']     = trip_date.weekday()
                    #print '5) ', new_point['weekday']

                    # TODO: mapeia o terminal de inicio e fim com o a
cidade de inicio e fim
                    new_point['start_city'] =  station_map[row['Start Termi
al']]
                    #print '6) ', new_point['start_city']
                    new_point['end_city'] =  station_map[row['End Terminal']
]
                    #print '7) ', new_point['end_city']
                    # TODO: existem dois nomes diferentes para o mesmo camp
. Trate cada um deles.
                    #o arquivo csv 201402_trip_data tem 'Subscription_type'
e o 201408 tem 'Subscriber type'
                    if 'Subscription Type' in row:
                        new_point['subscription_type'] = row['Subscription T
ype']
                    else:
```

```
                        new_point['subscription_type'] = row['Subscriber Typ
e']

                    # escreve a informação processada para o arquivo de saí
a.
                    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 [11]:

```
# 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 [12]:

```
# Carregue os dados novamente mostrando os dados
## TODO: Complete o código para leitura dos dados no arquivo criado na funç
ão acima
sample_data = pd.read_csv(trip_out)
display(sample_data.head())
```

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

In [13]:

```
# 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 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 [14]:

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

In [15]:

```
type(trip_data)
```

Out[15]:

```
pandas.core.frame.DataFrame
```

In [16]:

```
trip_data.head()
```

Out[16]:

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

Para ajudar nessa formatação: Python String Format Cookbook

In [17]:

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

# TODO: verificando os quartis
duration_qtiles = trip_data['duration'].quantile([.25, .5, .75]).as_matrix(
)
#print duration_qtiles
#print type(duration_qtiles)
print '25% das viagens foram mais curtas do que {:.2f} minutos'.format(dura
tion_qtiles[0])
print '25% das viagens foram mais compridas do que {:.2f} minutos'.format(d
uration_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 [18]:

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

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.

OBS: Para conseguir realizar esse plot utilizei: [Counting Values & Basic Plotting in Python]

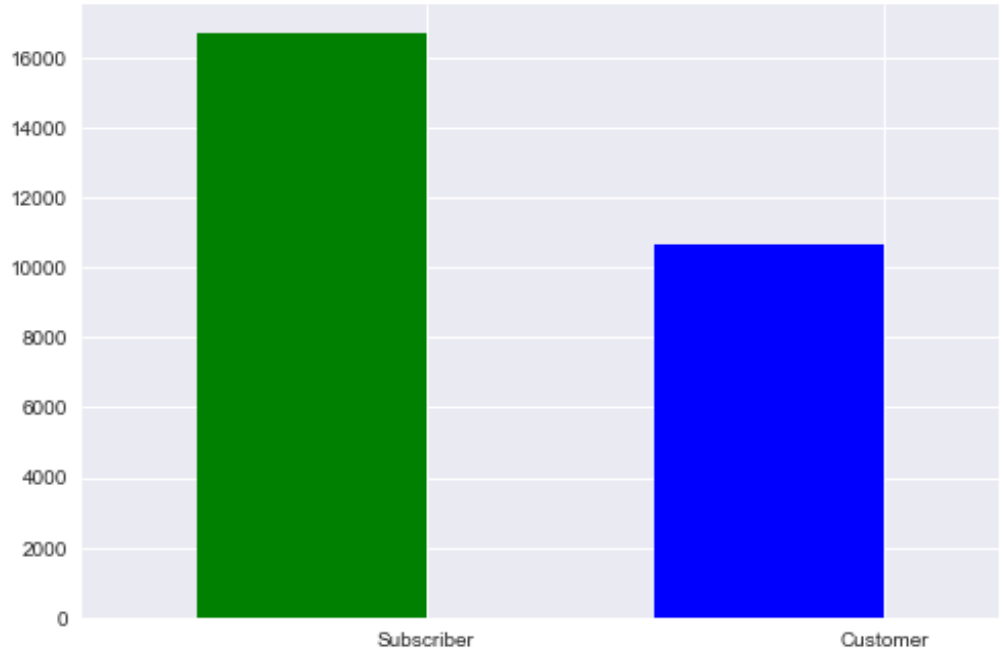
Tentativa 1

In [19]:

```
# TODO: plote um gráfico de barras que mostre quantidade de viagens por sub
scription_type
# lembrando que quando o comando .plot é usado, se pode escolher o tipo de
gráfico usando
# o parâmetro kind. Ex: plot(kind='bar')
print trip_data['subscription_type'].value_counts().plot(kind='bar', color=
'g''b', rot=0, position =1)
```

```
#trip_data['subscription_type'].plot(data= kind='bar')
```

```
Axes(0.125,0.125;0.775x0.755)
```



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 [20]:

```
# como o seu gráfico deve ficar. Descomente a linha abaixo caso queira roda
r 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)?

Tentativa 1

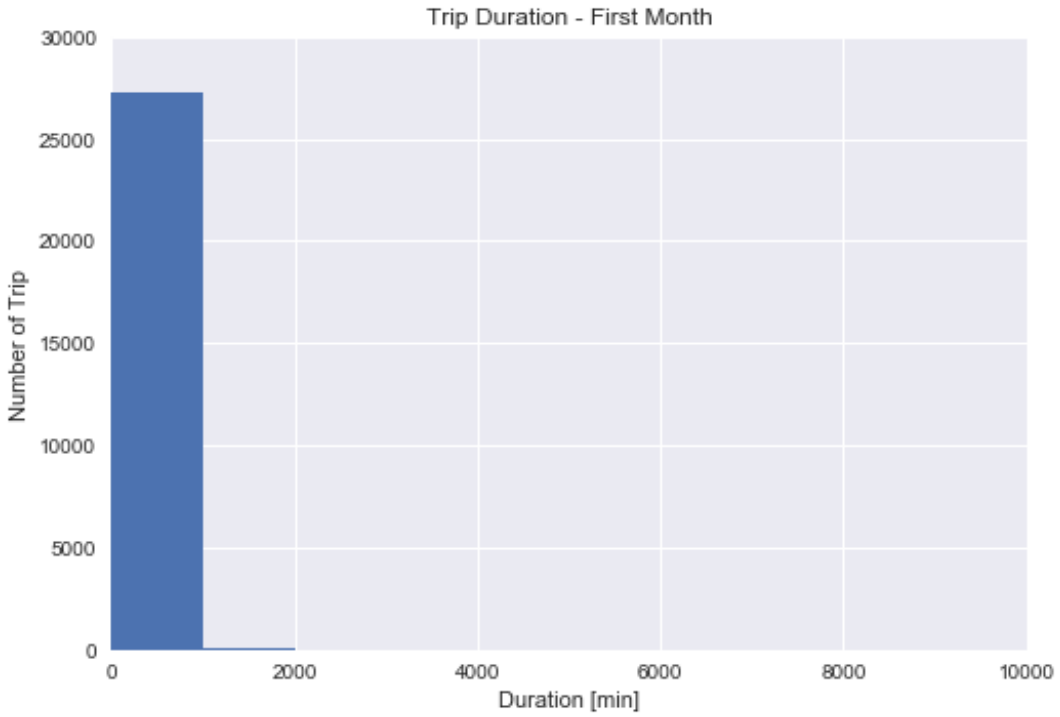
In [21]:

```
# TODO: Faça um gráfico baseado nas durações
trip_duration = trip_data['duration'].plot(kind='hist', bins=10, range=(0,1
0000), xlim=[0,10000], ylim=[0,30000], title='Trip Duration - First Month')
trip_duration.set_xlabel('Duration [min]')
trip_duration.set_ylabel('Number of Trip')
```

```
trip_duration.set_ylabel('Number of Trip')
```

Out[21]:

```
<matplotlib.text.Text at 0xbf9b160>
```



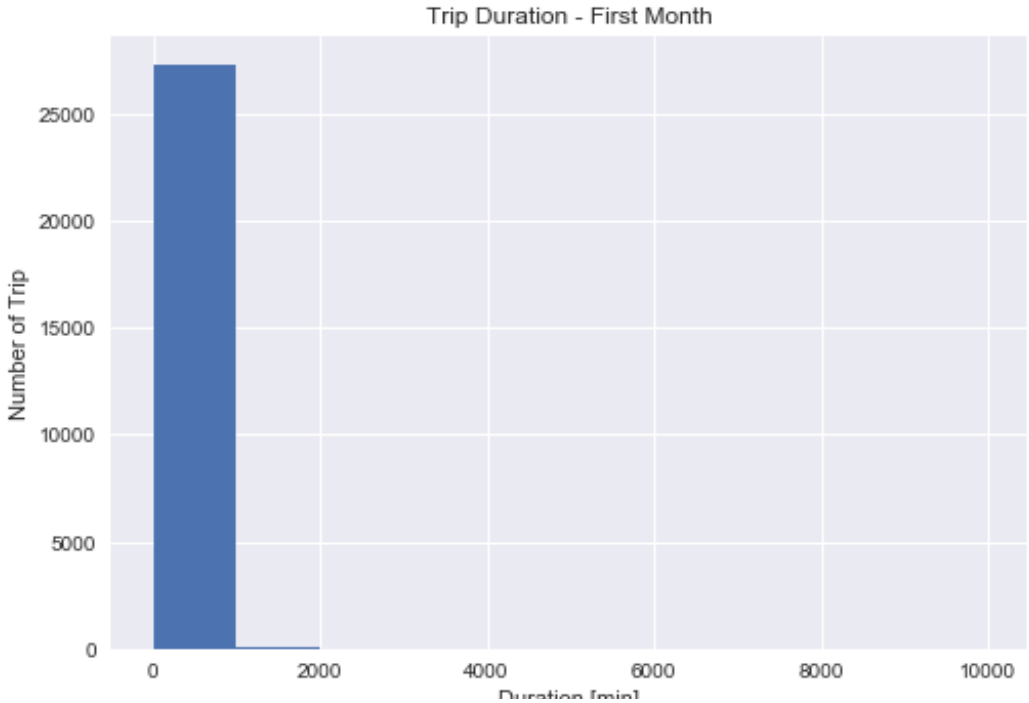
Tentativa 2 - Sem defnir limite de eixos e 'bins'

In [22]:

```
# TODO: Faça um gráfico baseado nas durações
trip_duration = trip_data['duration'].plot(kind='hist', title='Trip Duratio
n - First Month')
trip_duration.set_xlabel('Duration [min]')
trip_duration.set_ylabel('Number of Trip')
```

Out[22]:

```
<matplotlib.text.Text at 0xc08ad68>
```



Duration [min]

In [23]:

```
# rode este comando abaixo caso esteja em dúvida quanto ao resultado espera
do
#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.

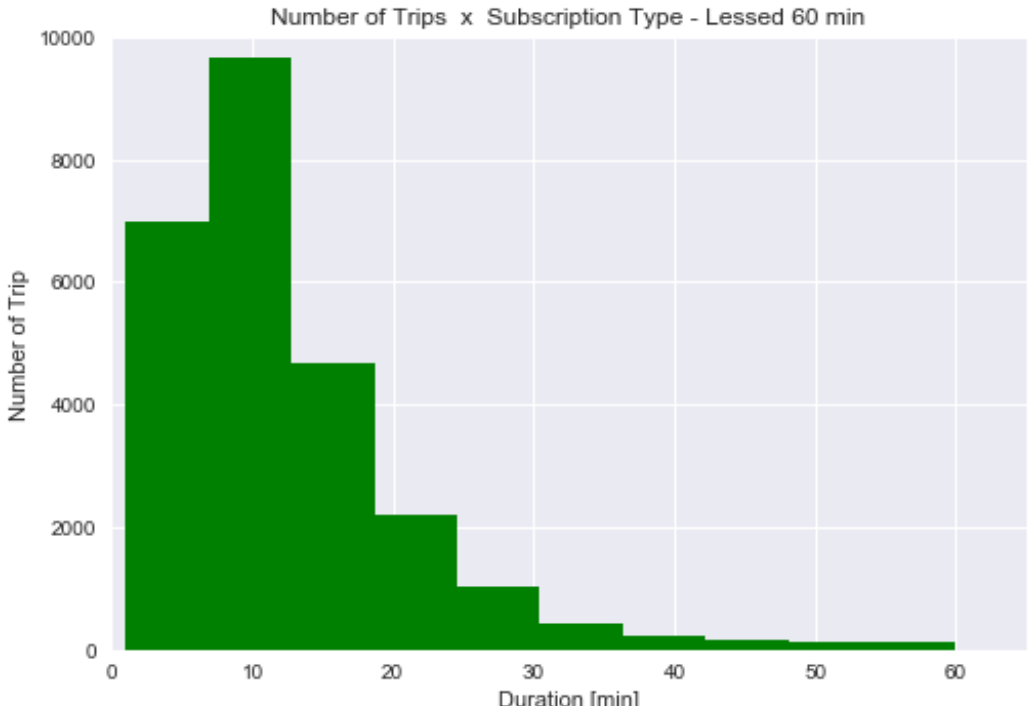
Ao explorar os dados, muitas vezes você precisará trabalhar com os parâmetros da função de visualização para facilitar a compreensão dos dados. É aqui que os filtros vão ajudar você. Comecemos por limitar as viagens de menos de 60 minutos.

In [24]:

```
# TODO: faça um gráfico de barras para os dados com duração inferior a 60 m
inutos.
duration_60_min = trip_data['duration'][trip_data['duration'] < 60]

#print type(duration_60_min)
#print duration_60_min.sort_values().tail()

duration_60_min_plot = duration_60_min.plot(kind='hist', color='g', title='
Number of Trips  x  Subscription Type - Lessed 60 min', xlim=[0,65], ylim=[
0,10000])

duration_60_min_plot.set_xlabel('Duration [min]')
duration_60_min_plot.set_ylabel('Number of Trip')
```

Out[24]:

```
<matplotlib.text.Text at 0xc467860>
```



In [25]:

```
# 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 ester 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.

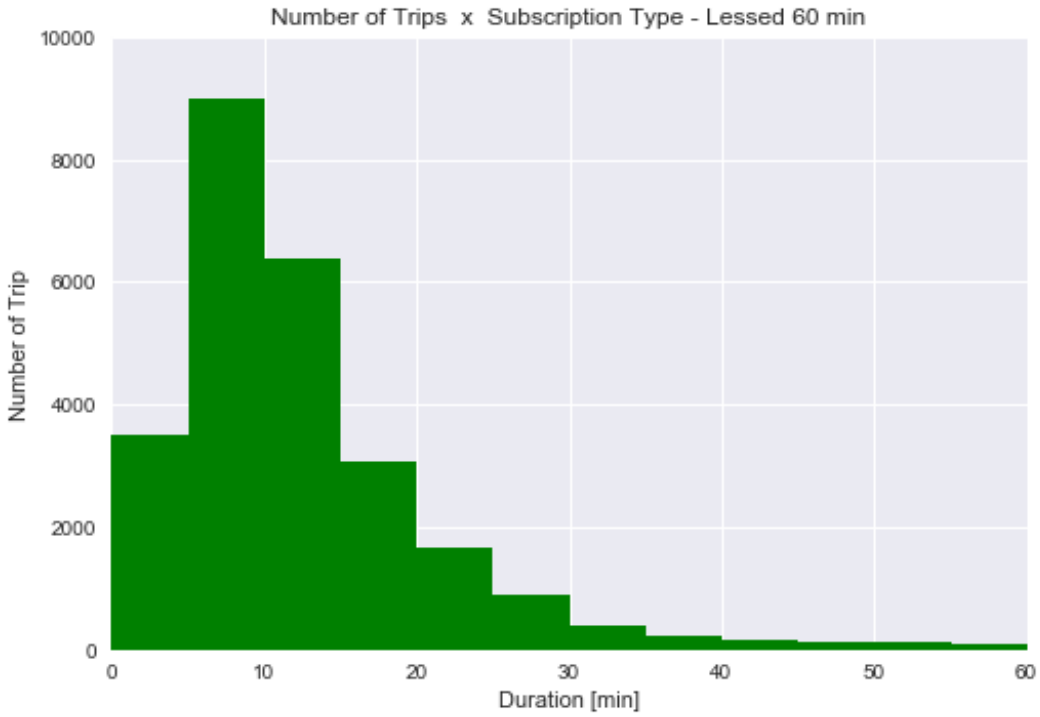
Tentativa 1

In [26]:

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

duration_60_min_plot = duration_60_min.plot(kind='hist', color='g', title='
Number of Trips  x  Subscription Type - Lessed 60 min', xlim=[0,60], ylim=[
0,10000], range=(0,60), bins=12, rwidth=5)

duration_60_min_plot.set_xlabel('Duration [min]')
duration_60_min_plot.set_ylabel('Number of Trip')
```

Out[26]:

```
<matplotlib.text.Text at 0xcc7ada0>
```



In [27]:

```
# rode esta linha para verificar como deve ficar o seu gráfico
#usage_plot(trip_data, 'duration', ['duration < 60'], boundary = 0, bin_wid
th = 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 exercicio anterior e responda:

Qual o intervalo de duração com maior quantidade de viagens?

Reposta: O intervalo de duração com a maior quantidade de viagens é o de 5 a 10 minutos, com mais de 8000 viagens.

Pergunta 4.1

Qual viagem de 5 minutos de duração tem a maior quantidade de viagens? Aproximadamente quantas viagens foram feitas nesta faixa de tempo?

Dica: Identifique a viagens pela origem e destino, calcule quantas viagem de 5 minutos de duração foram realizadas para cada origem e destino. Após isso calcule o total de viagens com 5 minutos de duração.

Reposta:

- *RESPONDENDO A PRIMEIRA PERGUNTA:*

A viagem de 5 minutos com a maior quantidade de viagens foi De: San Francisco Para: San Francisco

- *RESPONDENDO A SEGUNDA PERGUNTA:*

O total de viagens feitos dentro do período de 5 minutos foi de 11326.

Determinado a resposta da Pergunta 4.1

In [28]:

```
duration_5_min = trip_data[['duration', 'start_city', 'end_city']][trip_dat
a['duration'] <= 5]
#print type(duration_5_min)
#print duration_5_min.head(20)
#print len(duration_5_min.index)
trip_5_min ={}
for (start_city, end_city, duration) in zip(duration_5_min['start_city'], d
uration_5_min['end_city'], duration_5_min['duration']):
    if str('De: ' + start_city + ' Para: ' + end_city) not in trip_5_min:
        trip_5_min[str('De: ' + start_city + ' Para: ' + end_city)] = int(du
ration)
```

```
    else:
        trip_5_min[str('De: ' + start_city + ' Para: ' + end_city)] += int(d
uration)

#print trip_5_min
#print '\n'
print 'RESPONDENDO A PRIMEIRA PERGUNTA:'
print 'A viagem de 5 minutos com a maior quantidade de viagens foi ' + sort
ed(trip_5_min, key=trip_5_min.__getitem__)[-1]
print '\n'
print 'RESPONDENDO A SEGUNDA PERGUNTA:'
print 'O total de viagens feitos dentro de um período de 5 minutos foi de '
, sum((trip_5_min.values())),'.'
```

```
RESPONDENDO A PRIMEIRA PERGUNTA:
A viagem de 5 minutos com a maior quantidade de viagens foi De: San Francis
co Para: San Francisco


RESPONDENDO A SEGUNDA PERGUNTA:
O total de viagens feitos dentro de um período de 5 minutos foi de  11326 .
```

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 [29]:

```
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 abaixo ou no começo do notebook caso esteja em uma nova sessão)

In [30]:

```
# Importa todas as bibliotecas necessárias
%matplotlib inline
```

```
import csv
from datetime import datetime
import numpy as np
import pandas as pd

from babs_datacheck import question_3
from babs_visualizations import usage_stats, usage_plot
from IPython.display import display
```

In [31]:

```
trip_data = pd.read_csv('babs_y1_y2_summary.csv')
display(trip_data.tail(20))
```

	duration	start_date	start_year	start_month	start_hour	weekday	start_city	end_
669939	25.600000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669940	25.750000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669941	21.500000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669942	10.500000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669943	5.550000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669944	115.616667	01/09/2014	2014	9	8	0	San Francisco	San Franc
669945	7.500000	01/09/2014	2014	9	8	0	San Francisco	San Franc
669946	2.683333	01/09/2014	2014	9	8	0	San Francisco	San Franc
669947	289.933333	01/09/2014	2014	9	7	0	Mountain View	Mour View
669948	288.283333	01/09/2014	2014	9	7	0	Mountain View	Mour View
669949	2.816667	01/09/2014	2014	9	7	0	San Francisco	San Franc
669950	94.450000	01/09/2014	2014	9	7	0	San Jose	San J
669951	7.350000	01/09/2014	2014	9	6	0	San Francisco	San Franc
669952	6.633333	01/09/2014	2014	9	5	0	San Francisco	San Franc
669953	4.000000	01/09/2014	2014	9	4	0	San Francisco	San Franc
669954	10.316667	01/09/2014	2014	9	4	0	San	San

	duration	start_date	start_year	start_month	start_hour	weekday	start_city	end_
669955	111.866667	01/09/2014	2014	9	3	0	San Francisco	San Franc
669956	8.966667	01/09/2014	2014	9	0	0	San Francisco	San Franc
669957	9.466667	01/09/2014	2014	9	0	0	San Francisco	San Franc
669958	9.483333	01/09/2014	2014	9	0	0	San Francisco	San Franc

In [32]:

```
print 'O total de viagens realizadas nesse novo arquivo é de ', len(trip_data.index)

print "A 'start_hour' máxima é ", max(trip_data['start_hour'].values), " e a mínima é de ", min(trip_data['start_hour'].values)
```

```
O total de viagens realizadas nesse novo arquivo é de  669959
A 'start_hour' máxima é  23  e a mínima é de  0
```

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 "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](#).

Para entender um pouco mais como e quais gráficos podem ser úteis, leia [este documento](#). Ele lhe dará um pouco de idéia de como mostrar os dados de forma mais acertada

OBJEIVO DA 'ANÁLISE PRÓPRIA': O objetivo dessa primeira análise é entender quais são os horários de maior movimento comparando os dois tipo de usuarios (Customer vs Subscriber). Com esses horários determinados, é possivel analisar dados estatisticos sobre a 'duration' nesses horários e ainda analisar a diferença desses dados para os dias da semana e finais de semana, onde monday=0, ..., sunday=6.

In [33]:

```
#Separando por subscription_type


customer = trip_data[['duration', 'start_hour', 'weekday',
'subscription_type']][trip_data['subscription_type'] == 'Customer']
subscriber = trip_data[['duration', 'start_hour', 'weekday',
'subscription_type']][trip_data['subscription_type'] == 'Subscriber']

print "The type of variable 'customer' is ", type(customer)
print "The type of variable 'subscriber' is ", type(subscriber)
print '\n'

print customer.head()
print subscriber.head()
print '\n'

#print customer.tail()
#print subscriber.tail()
#print '\n'


#print len(list(subscriber.values))
#print len(list(customer.values))
#print '\n'
print 'O total de viagens realizadas é de ', len(subscriber.index) + len(cu
stomer.index), '. Condizente com o valor anteriormente encontrado.'
```

```
The type of variable 'customer' is  <class 'pandas.core.frame.DataFrame'>
The type of variable 'subscriber' is  <class 'pandas.core.frame.DataFrame'>


    duration  start_hour  weekday subscription_type
24  2.683333          10        3          Customer
30  2.883333          11        3          Customer
41  3.133333          19        3          Customer
48  3.483333          17        3          Customer
51  3.566667          17        3          Customer
   duration  start_hour  weekday subscription_type
0  1.050000          14        3        Subscriber
1  1.166667          14        3        Subscriber
2  1.183333          10        3        Subscriber
3  1.283333          11        3        Subscriber
4  1.383333          12        3        Subscriber


O total de viagens realizadas é de  669959 . Condizente com o valor anterio
rmente encontrado.
```

In [34]:

```
#Definindo a função que retorna o start_hour de maior movimento
```

```
def busy_hour(data_file):
    #empty dict with all possibles start_hour
    start_hours = {0:int(), 1:int(), 2:int(), 3:int(), 4:int(), 5:int(), 6:i
nt(), 7:int(), 8:int(), 9:int(), 10:int(), 11:int(), 12:int(),
                   13:int(), 14:int(), 15:int(), 16:int(), 17:int(), 18:int(
), 19:int(), 20:int(), 21:int(), 22:int(), 23:int()}
    #print type(start_hours.keys()[0])
    for hour in data_file['start_hour']:
        start_hours[hour] += 1
    return start_hours
```

In [35]:

```
#aplicando para os dois tipos de usuario
hours_subscriber = busy_hour(subscriber)
hours_customer = busy_hour(customer)
```

In [36]:

```
#Usando matplot.pylab para plotar o comportamento e identificar os horários
de maior movimento
import matplotlib.pyplot as plt
```

Maior movimento para 'subscriber':

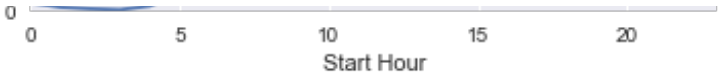
In [37]:

```
subscriber_x = hours_subscriber.keys()
subscriber_y = hours_subscriber.values()

print "O total de viagens ralizadas por usuários do tipo 'subscriber' é de
", sum(subscriber_y)

plt.plot(subscriber_x, subscriber_y)
plt.xlabel('Start Hour')
plt.ylabel('Frequency')
plt.title('Busy Hours for Subscriber')
plt.axis([0, 23, 0, 83000])
```

```
O total de viagens ralizadas por usuários do tipo 'subscriber' é de  566746
```

Out[37]:

```
[0, 23, 0, 83000]
```



Ánalise do Gráfico 'Busy Hours for Subscriber':

Podemos perceber que existem dois horários de grandes picos de movim ento no caso do usuario 'subscriber'. O primeiro é no período entre as 7 e 9h e o segundo das entre as 16 e as 18h. Também há um pico me nor aproximadamente as 12h.

Maior movimento para 'customer':

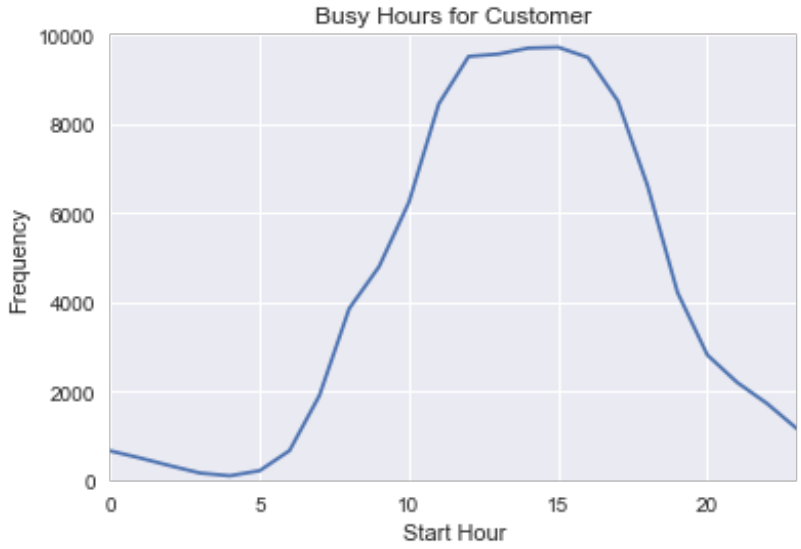
In [38]:

```
customer_x = hours_customer.keys()
customer_y = hours_customer.values()

print  "O total de viagens ralizadas por usuários do tipo 'customer' é de "
, sum(customer_y)

plt.plot(customer_x, customer_y)
plt.xlabel('Start Hour')
plt.ylabel('Frequency')
plt.title('Busy Hours for Customer')
plt.axis([0, 23, 0, 10000])
```

```
O total de viagens ralizadas por usuários do tipo 'customer' é de  103213
```

Out[38]:

```
[0, 23, 0, 10000]
```



Ánalise do Gráfico 'Busy Hours for Subscriber':

Podemos perceber que existem um único horário de grande movimento en tre 12 e 17h.

In [39]:

```
print "Viagens dos usuários do tipo 'subscriber' representam ", (float(sum(
```

```
subscriber_y))/float(len(trip_data.index)))*100,"% do total de viagens."

print "Já os usuários do tipo 'customer' representam ", (float(sum(customer
_y))/float(len(trip_data.index)))*100,"%."
```

```
Viagens dos usuários do tipo 'subscriber' representam  84.5941318797 % do t
otal de viagens.
Já os usuários do tipo 'customer' representam  15.4058681203 %.
```

É possível observar que os diferentes tipos de usuários apresentam diferentes comportamento no que diz a respeito dos horários de urlização. Agora, a análise seguirá para análisar informações estatísticas sobre a duração de seus viagens e dias de utlização, comparando a utilização em dias de semana e fianis de semana.

Análise estatística da duração das viagens nos horários de maior movmento

Para usuários do tipo 'subscriber':

A análise será feita no intervalo de maior movimento, ou seja, entre as 6 e as 9h. Intervalo definido com base no gráfico 'Busy Hours for Subscriber'.

In [40]:

```
subscriber_busy = subscriber[['duration', 'weekday', 'start_hour']][(subscr
iber['start_hour']>=7) & (subscriber['start_hour']<=9)]

#testar funcionamento
print 'The maximum start_hour is ', max(subscriber_busy['start_hour'].value
s)
print 'The minimum start_hour is ', min(subscriber_busy['start_hour'].value
s)
print type(subscriber_busy)
```

```
The maximum start_hour is  9
The minimum start_hour is  7
<class 'pandas.core.frame.DataFrame'>
```

In [41]:

```
#Estatistica com describe() function
print subscriber_busy['duration'].describe()
```

```
count    182131.000000
mean          9.517797
std          36.072745
min           1.000000
25%           5.616667
50%           8.083333
75%          11.116667
max       10322.033333
Name: duration, dtype: float64
```

Análise sobre os dados estatísticos:

Os dados mostram um comportamento de trajetos com tempos de durações mais curtos, onde não

há taxamento extra. Os valores a cima do terceiro quartil não é representativo, visto que 75% ocorre em aproximadamente 11 minutos e o desvio padrão é alto, bem como o há valores muito altos que devem estar mascarando o padrão desse tipo de usuário. Portanto, podemos resolver isso retirando os dados que estão a cima dos 75% dos dados.

In [42]:

```
#Removendo dados a cima de 75% dos dados
subscriber_busy_clean = subscriber_busy[['duration']][subscriber_busy['dura
tion'] <= subscriber_busy['duration'].quantile(.75)]
#Testa para analisar se a limpeza funcionou
subscriber_busy_clean.sort_values('duration').tail()
```

Out[42]:

	duration
71925	11.116667
416194	11.116667
638018	11.116667
574574	11.116667
397300	11.116667

In [43]:

```
#Reexecutando funcao describe() para os dados limpos
subscriber_busy_clean.describe()
```

Out[43]:

	duration
count	136734.000000
mean	6.837652
std	2.339572
min	1.000000
25%	4.983333
50%	6.833333
75%	8.733333
max	11.116667

Agora o desvio padrão é um valor menor e os dados representam de forma mais adequada os usuários do tipo 'subscriber'.

Para usuários do tipo 'customer':

A análise será feita no intervalo de 7 a 22h,visto que existe apenas um intervalo de maior movimento e já tão acentuado como no caso do 'subscriber'. Intervalo definido com base no gráfico 'Busy Hours

for Customer'.

In [44]:

```
customer_busy = customer[['duration', 'weekday', 'start_hour']][(customer['
start_hour']>=7) & (customer['start_hour']<=22)]

#testar funcionamento
print 'The maximum start_hour is ', max(customer_busy['start_hour'].values)
print 'The minimum start_hour is ', min(customer_busy['start_hour'].values)
print type(customer_busy)
```

```
The maximum start_hour is  22
The minimum start_hour is  7
<class 'pandas.core.frame.DataFrame'>
```

In [45]:

```
#Estatistica com describe() function
customer_busy['duration'].describe()
```

Out[45]:

```
count     99353.000000
mean         64.061153
std         946.387841
min           1.000000
25%          10.983333
50%          18.600000
75%          38.600000
max      287840.000000
Name: duration, dtype: float64
```

Análise sobre os dados estatísticos:

Os dados mostram um comportamento de trajetos mais longos, onde a média é bem superior ao tempo sem taxamento extra (30min). Nesse caso, o desvio padrão é é ainda mais alto, porém o terceiro quartil está mais próximo da média, demonstrando que valores mais baixos estão causando um empecilho para caracteriazação do comportamento do usuário do tipo 'customer'. Portanto, podemos limpar esses dados analisando apenas os dados a partir do primeiro quartil (25%) e analisar esse valor de máximo que é de quase 200 dias.

In [46]:

```
#Removendo dados a cima de 75% dos dados
customer_busy_clean = customer_busy[['duration']][customer_busy['duration']
>= customer_busy['duration'].quantile(.25)]
customer_busy_clean2 = customer_busy_clean[['duration']]
[customer_busy_clean['duration'] <= 100000]


#customer_busy_clean2 =
customer_busy_clean.sort_values('duration').drop([len(customer_busy_clean.s
_values('duration'))-1])
#Testa para analisar se a limpeza funcionou
print customer_busy_clean2.sort_values('duration').head()
print customer_busy_clean2.sort_values('duration').tail()
```

```
          duration
```

```
          duration
340      10.983333
11485    10.983333
202113   10.983333
10064    10.983333
215348   10.983333
             duration
421839  11481.650000
606063  12007.566667
80510   12037.266667
371066  18892.333333
382718  35616.666667
```

In [47]:

```
#Reexecutando funcao describe() para os dados limpos
customer_busy_clean2.describe()
```

Out[47]:

	duration
count	74540.000000
mean	79.010043
std	285.451038
min	10.983333
25%	16.316667
50%	24.550000
75%	60.666667
max	35616.666667

Mesmo execuando essa limpeza, o desvio padrão apresentou um valor mais alto, portanto, a média não é representativa para esses dados. Portanto, sria necessário uma analise mais profunda sobre a representatividade desses dados em relação a caracteriazação do comportamento dos usuários do tipo 'customer'.

Análise dos dias da semana de maior utilização em relação aos dois tipos de usuários

Para essa análise o objetivo é plotar os dias da semana e número de viagens realizadas para cada dia, comparando os dias de maior movimento para os dois tipos de usuários.

Para usuários do tipo 'subscriber':

In [48]:

```
#Filtrando para um Pandas.Series com apenas os weeksdays
subscriber_weekday = subscriber['weekday']

print type(subscriber_weekday)
```

```
print subscriber_weekday.head()
print subscriber_weekday.tail()
```

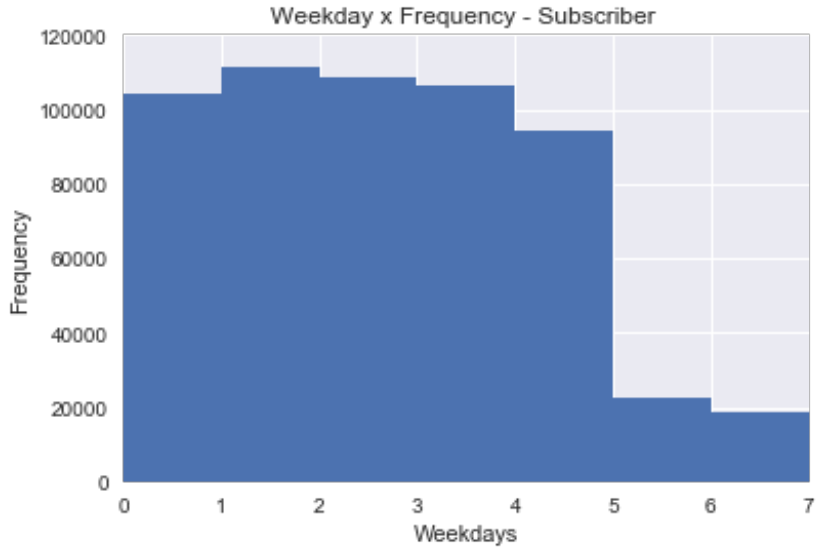
```
<class 'pandas.core.series.Series'>
0    3
1    3
2    3
3    3
4    3
Name: weekday, dtype: int64
669949    0
669951    0
669952    0
669953    0
669954    0
Name: weekday, dtype: int64
```

In [49]:

```
#Criando um histograma que mostre a frequencia de utilização para cada dia da semana
subscriber_weekday.hist(range=(0,7), bins=7)
plt.xlim(0,7)
plt.ylim(0,120000)
plt.title('Weekday x Frequency - Subscriber ')
plt.xlabel('Weekdays')
plt.ylabel('Frequency')
```

Out[49]:

```
<matplotlib.text.Text at 0xb5f9588>
```



Análise do Gráfico 'Weekday x Frequency - Subscriber':

Queda abrupta de utilização dos serviços aos finais de semana.

Para usuários do tipo 'customer':

In [50]:

```
#Filtrando para um Pandas.Series com apenas os weeksdays
customer_weekday = customer['weekday']
```

```
print type(customer_weekday)
print customer_weekday.head()
print customer_weekday.tail()
print max(customer_weekday.values)
```

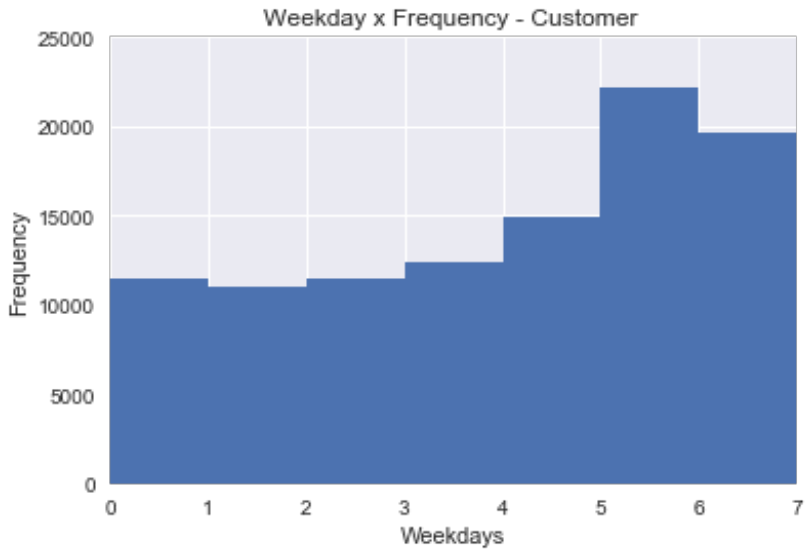
```
<class 'pandas.core.series.Series'>
24    3
30    3
41    3
48    3
51    3
Name: weekday, dtype: int64
669950    0
669955    0
669956    0
669957    0
669958    0
Name: weekday, dtype: int64
6
```

In [51]:

```
#Criando um histograma que mostre a frequencia de utilização para cada dia da semana
customer_weekday.hist(range=(0,7), bins=7)
plt.xlim(0,7)
plt.ylim(0,25000)
plt.title('Weekday x Frequency - Customer ')
plt.xlabel('Weekdays')
plt.ylabel('Frequency')
```

Out[51]:

```
<matplotlib.text.Text at 0xb5e7dd8>
```



Análise do Gráfico 'Weekday x Frequency - Customer':

Aumento de utilização do serviço aos finais de semana.

Pergunta 5a - Para subscriber

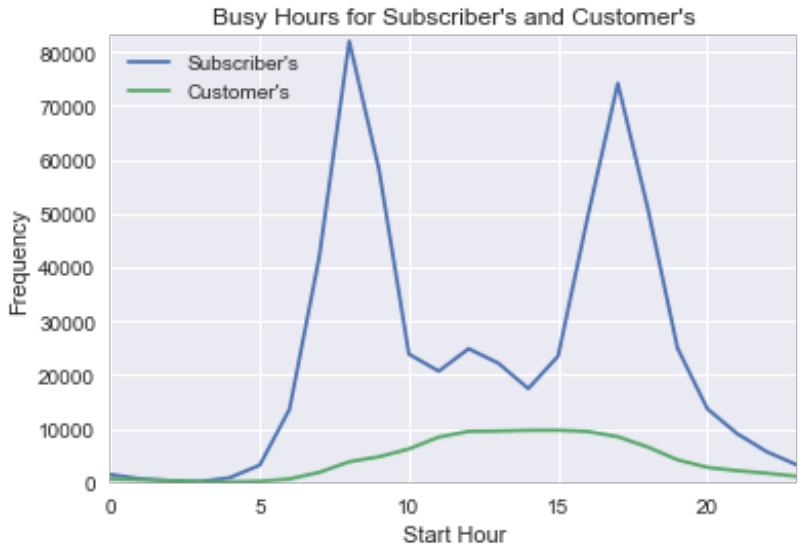
Explore os dados e faça um gráfico que demonstre alguma particularidade dos dados:

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

Gráfico mostrando o número de usuários relacionados a cada hora do dia.

In [52]:

```
subscriber_x = hours_subscriber.keys()
subscriber_y = hours_subscriber.values()

plt.plot(subscriber_x, subscriber_y,  label="Subscriber's")


customer_x = hours_customer.keys()
customer_y = hours_customer.values()

plt.plot(customer_x, customer_y, label="Customer's")


plt.xlabel('Start Hour')
plt.ylabel('Frequency')
plt.title("Busy Hours for Subscriber's and Customer's")
plt.axis([0, 23, 0, 83000])
plt.legend(loc='upper left')
plt.show()
```



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

Answer: O interresante é ver os diferentes comportamento dos usuários, onde os os usuários do tipo *'subscriber'* apresentam pico bem definidos nos horários de rotina de trabalho(8h, 12h, 18h) caracterizando-o. Já os usuários do tipo *'customer'* não apresentam picos de movimento, apenas ocorre um pequeno aumento (em uma taxa bem menor) entre o período das 9h as 18h, mostrando um comportamento compatível com turistas.

Esse gráfico foi selecionado pelo fato de fornecer informações significativas e marcantes sobre os tipos de usuários, podendo, assim, caracterila-los em alguns aspectos.

Pergunta 5b

Faça um gráfico que demonstre alguma particularidade dos dados:

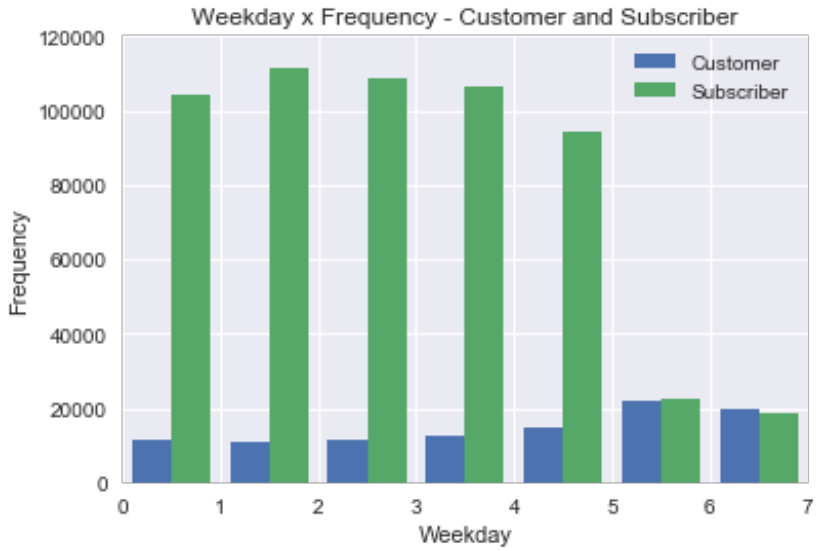
Gráfico comparando a frequencia de utilização dos dois tipos de usuários em todos os dias da semana.

In [53]:

```
# Gráfico Final 2
fig, ax = plt.subplots()
ax.hist([customer_weekday, subscriber_weekday], histtype="bar", label=("Customer", "Subscriber"), range=(0,7), bins=7)
ax.set_title("Weekday x Frequency - Customer and Subscriber")
ax.set_xlabel('Weekday')
ax.set_ylabel('Frequency')
ax.set_xlim(0,7)
ax.set_ylim(0,120000)
ax.legend()
```

Out[53]:

```
<matplotlib.legend.Legend at 0xd4b5400>
```



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

Answer: O gráfico a cima mostra uma grande diferencça entre os dois tipos de usuários emrelação aos dias da semana de utilização. Os usuários do tipo *'subscriber'* utilizam o serviço durando os dias da semana e bem menos aos finais de semana. O oposto ocorre com os usuários do tipo *'customer'*. Os *"subscriber's"* utilizam o serviço como locomoção no dia a dia, em suas rotinas e não como lazer. O oposto ocorre com os *"customer's"* onde ocorre um aumento de utilização do serviço aos finais de semana, porém a diferença de número de usuários nos dias de semana não é tão diferente aos finais de semana (quando compardo com os usuários do tipo *'subscriber'*) portanto esse tipo de usuários devem ser turistas, em sua maioria. A escolha desse gráfico foi motivada pelo fato de representar um comportamento significativo sobre os usuários, mostrando grandes diferenças nos tipos de usuários.

Conclusões

Parabéns pela conclusão do projeto! Esta é apenas uma amostragem do processo de análise de dados: gerando perguntas, limpando e explorando os dados. Normalmente, neste momento no processo de análise de dados, você pode querer tirar conclusões realizando um teste estatístico ou ajustando os dados a um modelo para fazer previsões. Há também muitas análises potenciais que podem ser realizadas se evoluirmos o código fornecido. Em vez de apenas olhar para o número de

viagens no eixo de resultados, você pode ver quais recursos afetam coisas como a duração da viagem. Nós também não analisamos como os dados meteorológicos se encaixam no uso de bicicletas.

Pergunta 6

Pense em um tópico ou campo de interesse onde você gostaria de poder aplicar as técnicas da ciência dos dados. O que você gostaria de aprender com o assunto escolhido?

Responda: Eu gostaria de aplicar análise de dados para diversas empresas e indústrias da minha cidade. Gostaria de aprender como realizaar análise de dados para diferentes ramos de atividade, contruindo, assim, um grande portifólio de projetos. Por outro lado, meu objetivo é evoluir para o aprendizado de Machine Learning, visando me tornar um especialista de Inteligência Artificial e conseguir, no futuro, concretizar um sonho de fundar uma empresa de técnologia ou solfware baseada nessas ferramentas.

> **Dica**: se quisermos compartilhar os resultados de nossa análise com os outros, existe uma outra opção que não é enviar o arquivo jupyter Notebook (.ipynb). Também podemos exportar a saída do Notebook de uma forma que pode ser aberto mesmo para aqueles sem o Python instalado. No menu **File** na parte superior esquerda, vá para o submenu **Download as**. Você pode então escolher um formato diferente que pode ser visto de forma mais geral, como HTML (.html) ou PDF (.pdf). Você pode precisar de pacotes adicionais ou software para executar essas exportações.