Understanding Pandas Series and DataFrames - Lab

Introduction

In this lab, let's get some hands-on practice working with data cleanup using Pandas.

Objectives

You will be able to:

- Use the `.map()` and `.apply()` methods to apply a function to a pandas Series or DataFrame
- Perform operations to change the structure of pandas DataFrames
- Change the index of a pandas DataFrame
- Change data types of columns in pandas DataFrames

Let's get started!

Import the file `'turnstile_180901.txt'` .

```
# Import the required libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
# Import the file 'turnstile_180901.txt'
df = pd.read_csv('turnstile_180901.txt')

# Print the number of rows ans columns in df
print(df.shape)

# Print the first five rows of df
df.head()
```

```
(197625, 11)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	08/25/2018
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	08/25/2018

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	08/25/2018
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	08/25/2018
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	08/25/2018

Rename all the columns to lower case:

```
# Rename all the columns to lower case
df.columns = [col.lower() for col in df.columns]
```

Change the index to `'linename'` :

```
# Change the index to 'linename'
df = df.set_index('linename')
df.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	c/a	unit	scp	station	division	date	time
linename							

	c/a	unit	scp	station	division	date	time
linename							
NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	00:00:00
NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	04:00:00
NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	08:00:00
NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	12:00:00
NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	16:00:00

Reset the index:

```
# Reset the index
df = df.reset_index()
df.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	linename	c/a	unit	scp	station	division	date	tim

	linename	c/a	unit	scp	station	division	date	tim
0	NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	00:00
1	NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	04:00
2	NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	08:00
3	NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	12:00
4	NQR456W	A002	R051	02-00-00	59 ST	BMT	08/25/2018	16:00

Create another column `'Num_Lines'` that is a count of how many lines pass through a station. Then sort your DataFrame by this column in descending order.

Hint: According to the data dictionary, LINENAME represents all train lines that can be boarded at a given station. Normally lines are represented by one character. For example, LINENAME 456NQR represents trains 4, 5, 6, N, Q, and R.

```
# Add a new 'num_lines' column
df['num_lines'] = df['linename'].map(lambda x: len(x))
```

Write a function to clean column names:

```
def clean(col_name):
    # Clean the column name in any way you want to. Hint: think back to str methods
    cleaned = col_name.strip()
    return cleaned
```

```
# Use the above function to clean the column names
df.columns = [clean(col) for col in df.columns]
```

```
# Check to ensure the column names were cleaned
df.columns
```

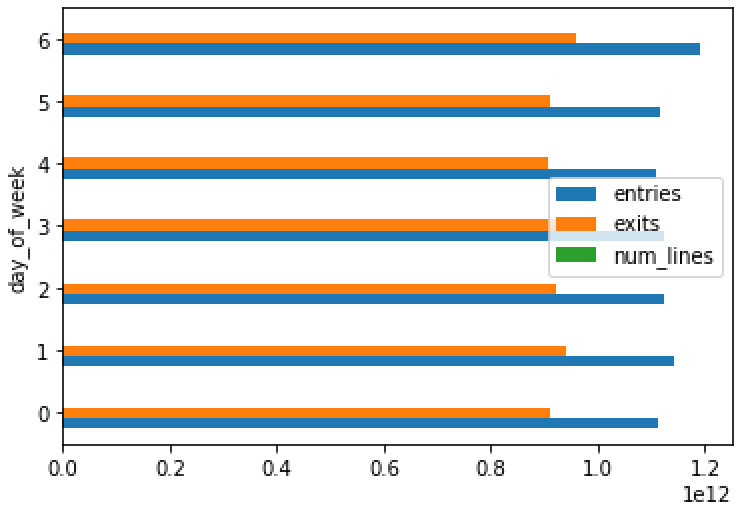
```
Index(['linename', 'c/a', 'unit', 'scp', 'station', 'division', 'date', 'time',
       'desc', 'entries', 'exits', 'num_lines'],
      dtype='object')
```

- Change the data type of the `'date'` column to a date
- Add a new column `'day_of_week'` that represents the day of the week

```
# Convert the data type of the 'date' column to a date
df['date'] = pd.to_datetime(df['date'])

# Add a new column 'day_of_week' that represents the day of the week
df['day_of_week'] = df['date'].dt.dayofweek
```

```
# Group the data by day of week and plot the sum of the numeric columns
grouped = df.groupby('day_of_week').sum()
grouped.plot(kind='barh')
plt.show()
```



- Remove the index of `grouped`
- Print the first five rows of `grouped`

```
# Reset the index of grouped
grouped = grouped.reset_index()
```

```
# Print the first five rows of grouped
grouped.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

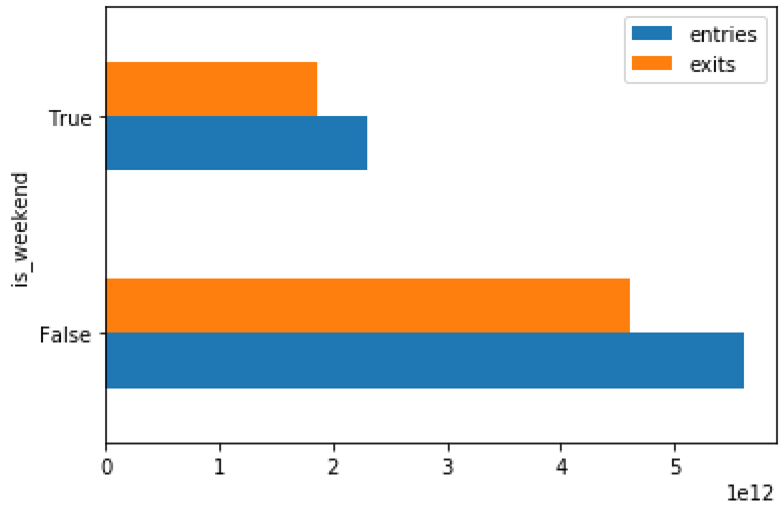
	day_of_week	entries	exits	num_lines
0	0	1114237052454	911938153513	76110
1	1	1143313287046	942230721477	77303
2	2	1123655222441	920630864687	75713
3	3	1122723988662	920691927110	76607
4	4	1110224700078	906799065337	75573

Add a new column `'is_weekend'` that maps the `'day_of_week'` column using the dictionary `weekend_map`

```
# Use this dictionary to create a new column
weekend_map = {0:False, 1:False, 2:False, 3:False, 4:False, 5:True, 6:True}

# Add a new column 'is_weekend' that maps the 'day_of_week' column using weekend_map
grouped['is_weekend'] = grouped['day_of_week'].map(weekend_map)
```

```
# Group the data by weekend/weekday and plot the sum of the numeric columns
wkend = grouped.groupby('is_weekend').sum()
wkend[['entries', 'exits']].plot(kind='barh')
plt.show()
```



Remove the `'c/a'` and `'scp'` columns.

```
# Remove the 'c/a' and 'scp' columns
df = df.drop(['c/a', 'scp'], axis=1)
df.head(2)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	linename	unit	station	division	date	time	desc	e
0	NQR456W	R051	59 ST	BMT	2018-08-25	00:00:00	REGULAR	67
1	NQR456W	R051	59 ST	BMT	2018-08-25	04:00:00	REGULAR	67

Analysis Question

What is misleading about the day of week and weekend/weekday charts you just plotted?

```
# Answer: The raw data for entries/exits is cumulative.
# As such, you would first need to order the data by time and station,
# and then calculate the difference in order to produce meaningful aggregations.
```

Summary

Great! You practiced your data cleanup skills using Pandas.

Releases

No releases published

Packages

No packages published

Contributors 8

Languages

● **Jupyter Notebook** 100.0%