1. Problem Set 2.1
3. **import** pandas
4. **import** pandasql

7. **def** num\_rainy\_days(filename):
8. weather\_data = pandas.read\_csv(filename)
10. q = """
11. SELECT COUNT(\*)
12. FROM weather\_data
13. WHERE cast(rain as integer) = 1;
14. """
16. #Execute your SQL command against the pandas frame
17. rainy\_days = pandasql.sqldf(q.lower(), locals())
18. **return** rainy\_days

21. Problem Set 2.2
23. **import** pandas
24. **import** pandasql

27. **def** max\_temp\_aggregate\_by\_fog(filename):
28. weather\_data = pandas.read\_csv(filename)
30. q = """
31. SELECT fog, MAX(cast (maxtempi as integer))
32. FROM weather\_data
33. GROUP BY fog;
34. """
36. #Execute your SQL command against the pandas frame
37. foggy\_days = pandasql.sqldf(q.lower(), locals())
38. **return** foggy\_days
40. Problem Set 2.3
42. **import** pandas
43. **import** pandasql
45. **def** avg\_weekend\_temperature(filename):
46. weather\_data = pandas.read\_csv(filename)
48. q = """
49. SELECT avg(cast (meantempi as integer))
50. FROM weather\_data
51. WHERE cast (strftime('%w', date) as integer) IN (6,0);
52. """
54. #Execute your SQL command against the pandas frame
55. mean\_temp\_weekends = pandasql.sqldf(q.lower(), locals())
56. **return** mean\_temp\_weekends


60. Problem Set 2.4
62. **import** pandas
63. **import** pandasql
65. **def** avg\_min\_temperature(filename):
66. weather\_data = pandas.read\_csv(filename)
68. q = """
69. SELECT avg(cast (mintempi as integer))
70. FROM weather\_data
71. WHERE cast(rain as integer) = 1
72. AND cast(mintempi as integer) > 55;
73. """
75. #Execute your SQL command against the pandas frame
76. avg\_min\_temp\_rainy = pandasql.sqldf(q.lower(), locals())
77. **return** avg\_min\_temp\_rainy



82. Problem Set 2.5
84. **import** csv
86. **def** fix\_turnstile\_data(filenames):
87. **for** name **in** filenames:
88. f\_in = open(name, 'r')
89. f\_out = open('updated\_'+name, 'w')
90. reader\_in = csv.reader(f\_in, delimiter=',')
91. writer\_out = csv.writer(f\_out, delimiter=',')
92. **for** x **in** reader\_in:
93. c1 = x[0]
94. c2 = x[1]
95. c3 = x[2]
97. **for** k **in** range(len(x)/5):
98. out = [c1, c2, c3,
99. x[k\*5 + 3],
100. x[k\*5 + 4],
101. x[k\*5 + 5],
102. x[k\*5 + 6],
103. x[k\*5 + 7]]
104. writer\_out.writerow(out)
105. Problem Set 2.6
107. **def** create\_master\_turnstile\_file(filenames, output\_file):
108. with open(output\_file, 'w') as master\_file:
109. master\_file.write('C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn\n')
110. **for** filename **in** filenames:
111. f\_in = open(filename, 'r')
112. f\_in.next()
113. **for** line **in** f\_in:
114. master\_file.write(line)
115. f\_in.close

118. Problem Set 2.7
120. **import** pandas
122. **def** filter\_by\_regular(filename):
124. turnstile\_data = pandas.read\_csv(filename)
125. turnstile\_data = turnstile\_data[turnstile\_data['DESCn'] == 'REGULAR']
126. # your code here
127. # more of your code here
129. **return** turnstile\_data

132. Problem Set 2.8
134. **import** pandas
136. **def** get\_hourly\_entries(df):
138. df['ENTRIESn\_hourly'] = (df['ENTRIESn'] -  df['ENTRIESn'].shift(1)).fillna(1)
139. **return** df

142. Problem Set 2.9
144. **import** pandas
146. **def** get\_hourly\_exits(df):
148. df['EXITSn\_hourly'] = (df['EXITSn'] - df['EXITSn'].shift(1)).fillna(0)
149. **return** df

152. Problem Set 2.10
154. **import** pandas
156. **def** time\_to\_hour(time):
158. hour = int(time[0:2])# your code here
159. **return** hour

162. Problem Set 2.11
164. **import** datetime
166. **def** reformat\_subway\_dates(date):
168. date\_formatted = datetime.datetime.strptime(date, "%m-%d-%y").strftime("%Y-%m-%d")# your code here
169. **return** date\_formatted