DA_Fall21_HW_1 Due at Mid-night 10/19/2021

This homework will cover the following skills set

- Using sql
- Checking for null values and outliers
- Calculate simple statistics using both SQL and Pandas
- Calculate skew and correlation
- Basic Data Visualization
- How to fix missing values

Late Policy: Take off 50% after one day, 80% after two days

Make sure you have pandasql installed. If not, make sure you run the code in the following cell

In [180]:

```
import os
try:
    import pandasql as ps
except:
    print("Failed in import pandasql")
    os.system("pip install pandasql")
```

Import all standard libaries

In [181]:

```
import pandas as pd
import pandasql as ps
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading Car Crashes dataset

In [182]:

```
crash = pd.read_csv("car_crashes2.csv")
print(crash.shape)
crash.head()
```

```
(54, 10)
```

Out[182]:

	accidents	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losses	state
0	18.8	7.332	5.640	18.048	15.040	784.55	145.08	AL
1	18.1	7.421	4.525	16.290	17.014	1053.48	133.93	AK
2	18.6	6.510	5.208	15.624	17.856	899.47	110.35	AZ
3	22.4	4.032	5.824	21.056	21.280	827.34	142.39	AR
4	12.0	4.200	3.360	10.920	10.680	878.41	165.63	CA

- accidents is the total number of crashes for each state
- speeding is the average speeding in each state

In [183]:

```
crash.describe()
```

Out[183]:

	accidents	speeding	alcohol	not_distracted	no_previous	ins_premium	ins_losse
count	54.000000	53.000000	54.000000	54.000000	54.000000	53.000000	54.00000
mean	15.668519	5.028755	5.506778	13.545407	13.931500	908.318113	133.96740
std	4.087386	2.000479	5.157681	4.463259	3.730217	232.237787	24.70339
min	5.900000	1.792000	1.593000	1.760000	5.900000	641.960000	82.75000
25%	12.725000	3.774000	3.891000	10.345000	11.262000	768.950000	112.33250
50%	15.450000	4.608000	4.542000	13.816000	13.717000	861.180000	135.84000
75%	18.350000	6.510000	5.622000	16.215000	16.579000	1011.140000	152.06500
max	23.900000	9.450000	40.642000	23.661000	21.280000	2000.030000	194.78000

Question 1. Check if there are any null or NA

Type in your code here

In [184]:

```
crash.isnull().any()
```

Out[184]:

```
accidents         False
speeding           True
alcohol           False
not_distracted    False
no_previous       False
ins_premium        True
ins_losses        False
state             False
region            False
division          False
dtype: bool
```

Question 2. Remove any rows that has null or NA

Type in your code here

In [221]:

```
crash = crash[ ~np.isnan(crash.speeding)]
crash = crash[ ~np.isnan(crash.ins_premium)]
crash.shape
crash.isnull().any()
```

Out[221]:

```
accidents         False
speeding          False
alcohol           False
not_distracted    False
no_previous       False
ins_premium       False
ins_losses        False
state             False
region            False
division          False
dtype: bool
```

In [ ]:

```

```

Question 3. Check if there is any outliers in all numerical fields using both scatter and boxplot

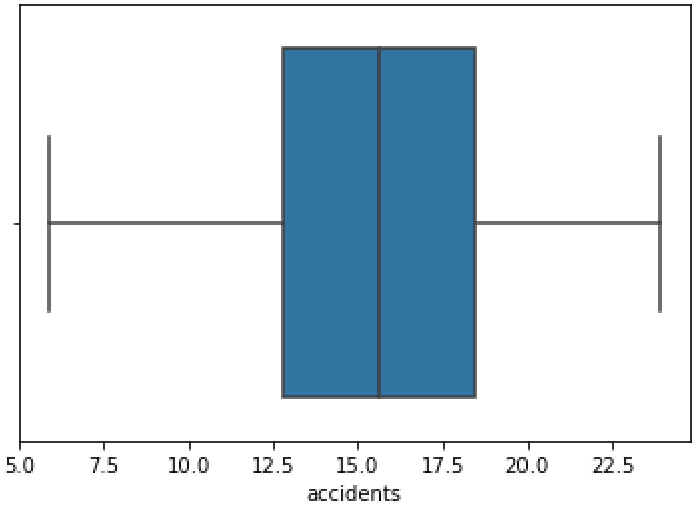
Type in your code here

In [187]:

```
sns.boxplot(x=crash['accidents'])
```

Out[187]:

```
<AxesSubplot:xlabel='accidents'>
```

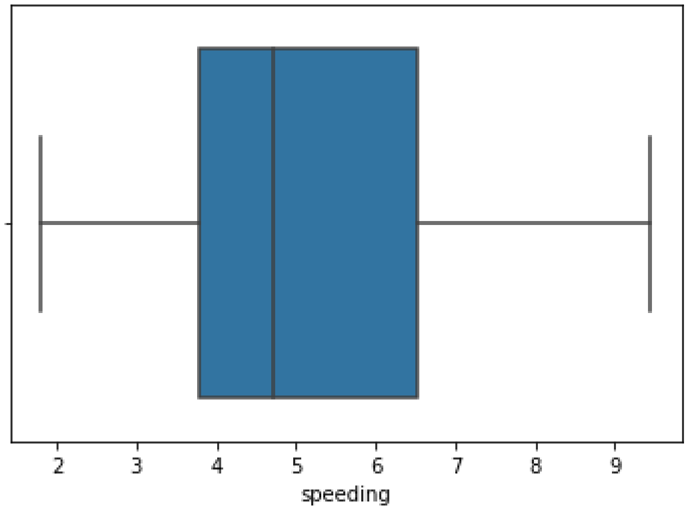


In [188]:

```
sns.boxplot(x=crash['speeding'])
```

Out[188]:

```
<AxesSubplot:xlabel='speeding'>
```

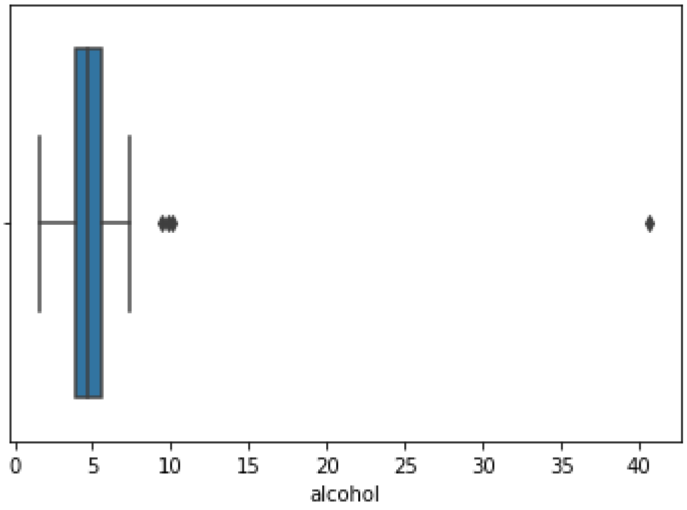


In [189]:

```
sns.boxplot(x=crash['alcohol'])
```

Out[189]:

```
<AxesSubplot:xlabel='alcohol'>
```

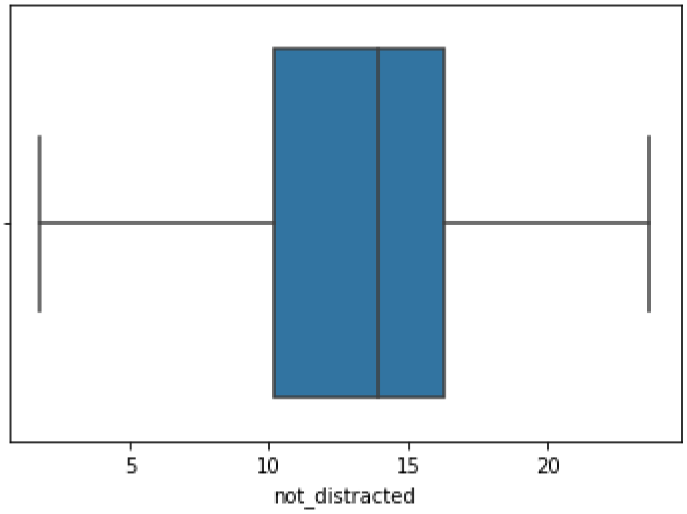


In [190]:

```
sns.boxplot(x=crash['not_distracted'])
```

Out[190]:

```
<AxesSubplot:xlabel='not_distracted'>
```

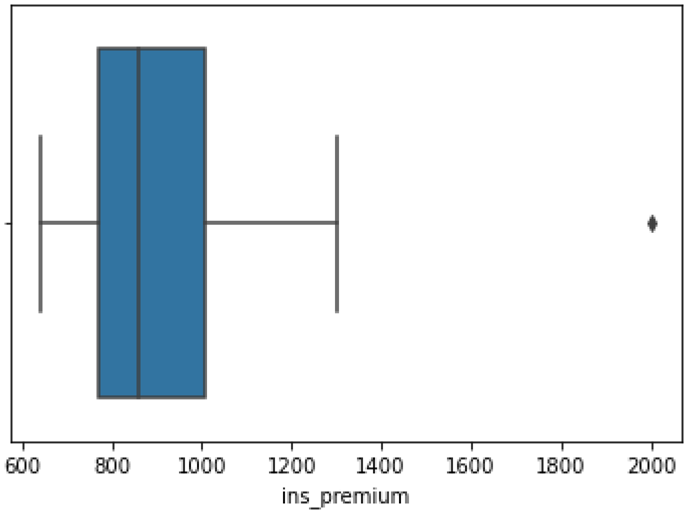


In [191]:

```
sns.boxplot(x=crash['ins_premium'])
```

Out[191]:

```
<AxesSubplot:xlabel='ins_premium'>
```

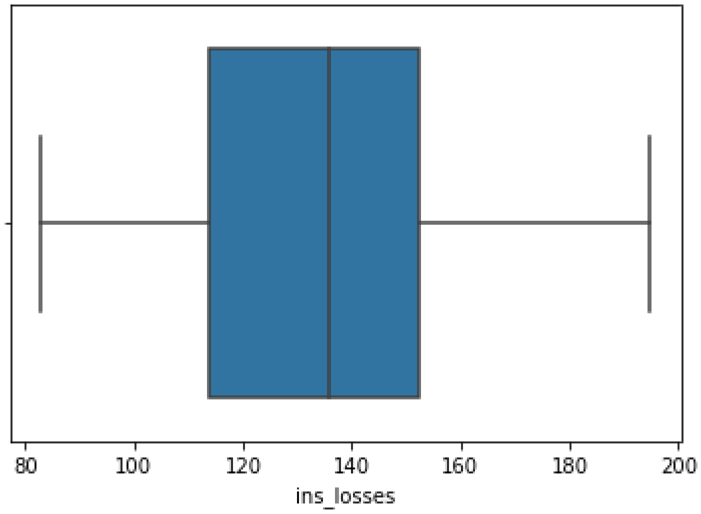


In [192]:

```
sns.boxplot(x=crash['ins_losses'])
```

Out[192]:

```
<AxesSubplot:xlabel='ins_losses'>
```

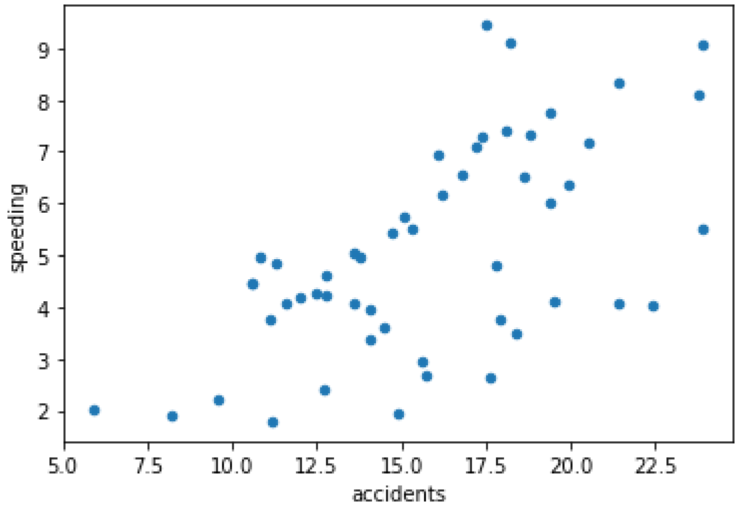


In [193]:

```
crash.plot.scatter(x='accidents', y='speeding')
```

Out[193]:

```
<AxesSubplot:xlabel='accidents', ylabel='speeding'>
```

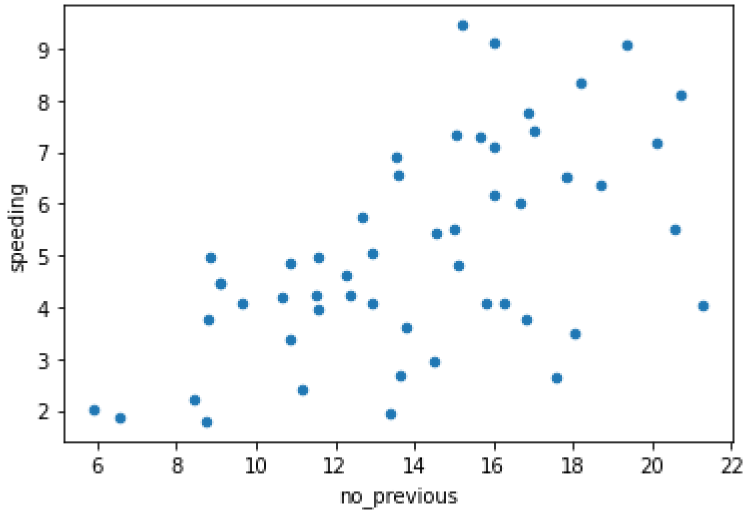


In [194]:

```
crash.plot.scatter(x='no_previous', y='speeding')
```

Out[194]:

```
<AxesSubplot:xlabel='no_previous', ylabel='speeding'>
```

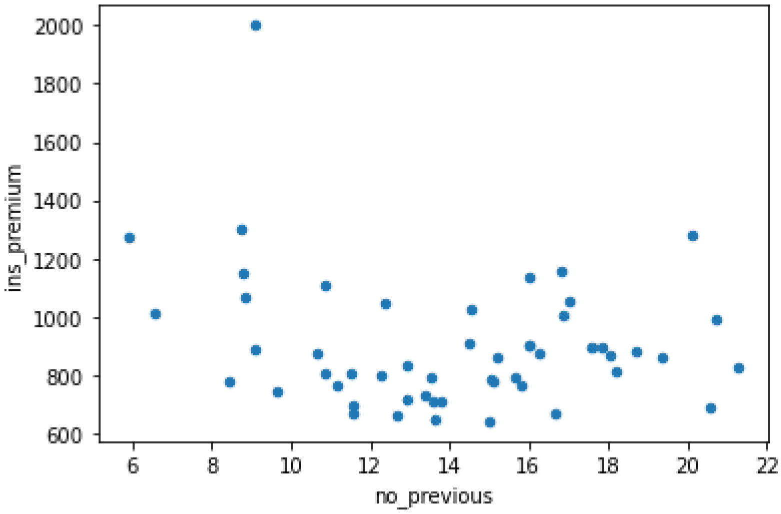


In [195]:

```
crash.plot.scatter(x='no_previous', y='ins_premium')
```

Out[195]:

```
<AxesSubplot:xlabel='no_previous', ylabel='ins_premium'>
```

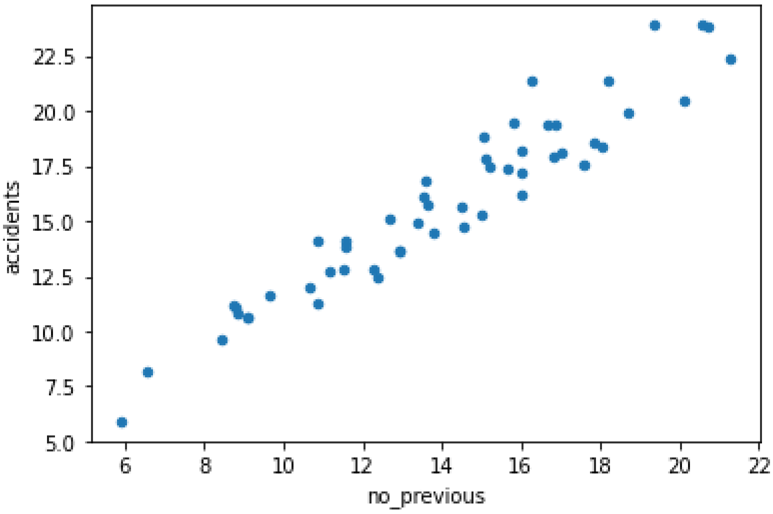


In [196]:

```
crash.plot.scatter(x='no_previous', y='accidents')
```

Out[196]:

```
<AxesSubplot:xlabel='no_previous', ylabel='accidents'>
```

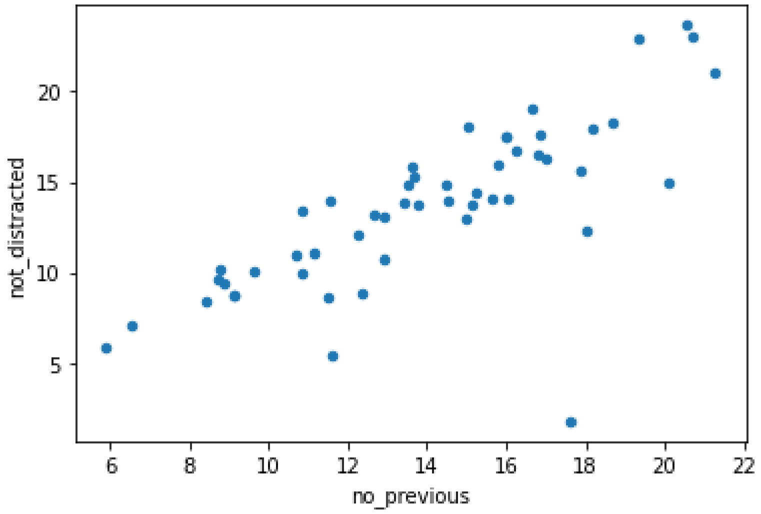


In [197]:

```
crash.plot.scatter(x='no_previous', y='not_distracted')
```

Out[197]:

```
<AxesSubplot:xlabel='no_previous', ylabel='not_distracted'>
```

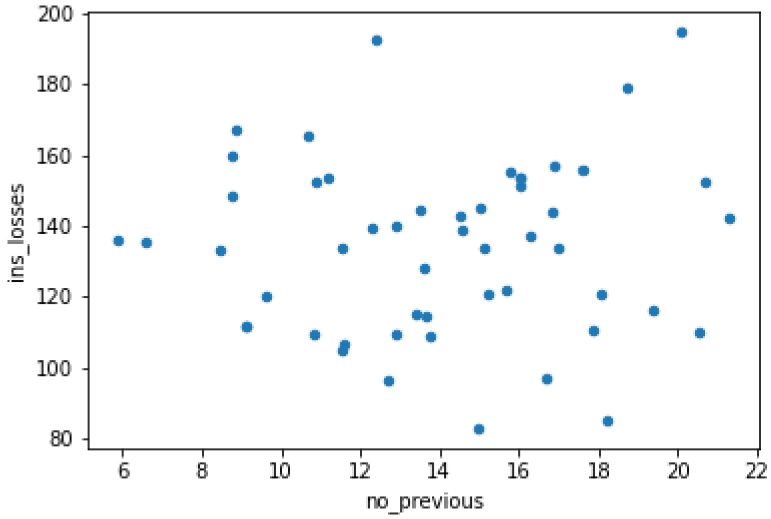


In [198]:

```
crash.plot.scatter(x='no_previous', y='ins_losses')
```

Out[198]:

```
<AxesSubplot:xlabel='no_previous', ylabel='ins_losses'>
```



In [ ]:

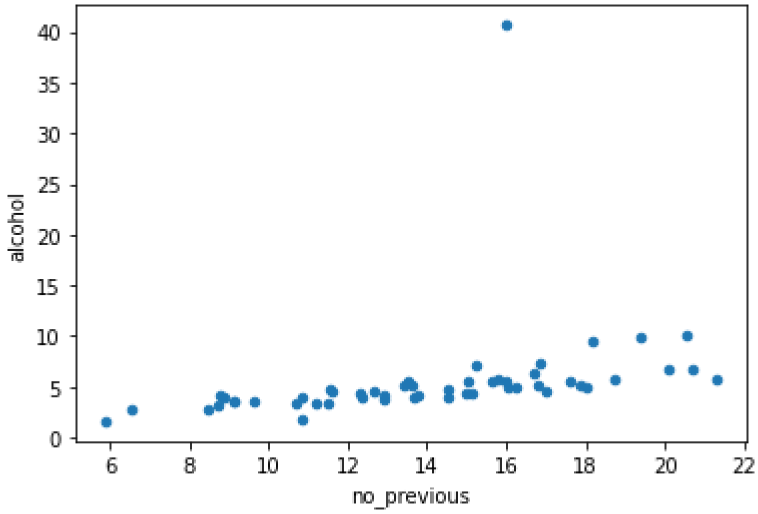
```

```

In [199]:

```
crash.plot.scatter(x='no_previous', y='alcohol')
```

Out[199]:

```
<AxesSubplot:xlabel='no_previous', ylabel='alcohol'>
```



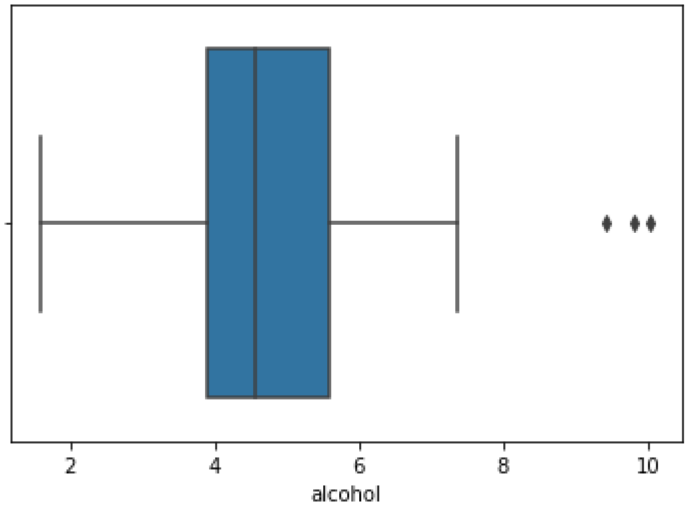
Question 4. Display and Remove (the real) outliers

In [200]:

```
crash = crash[crash['alcohol'] < 35]
sns.boxplot(x=crash['alcohol'],data=crash)
crash.shape
```

Out[200]:

```
(51, 10)
```

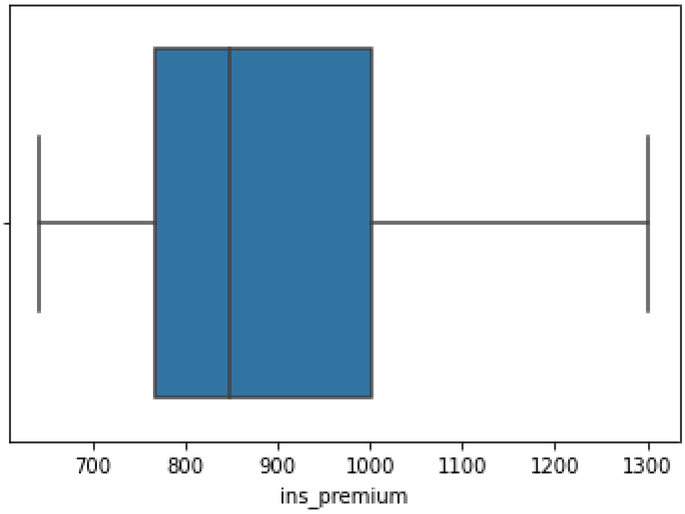


In [201]:

```
crash = crash[crash['ins_premium'] <1800]
sns.boxplot(x=crash['ins_premium'],data=crash)
crash.shape
```

Out[201]:

```
(50, 10)
```



In [ ]:

```

```

Type in your code here

Question 5. Calculate the average speeding in usa and Northeast region using SQL

Type in your code here

In [202]:

```
sql = "select region, avg(speeding) from crash where region = 'Northeast'"
df = ps.sqldf(sql)
df
```

Out[202]:

	region	avg(speeding)
0	Northeast	4.42475

Question 6. Calculate the average alcohol for each region using SQL

Type in your code here

In [203]:

```
sql = "select region, avg(alcohol) from crash group by region"
df = ps.sqldf(sql)
df
```

Out[203]:

	region	avg(alcohol)
0	Midwest	4.996917
1	Northeast	3.980500
2	South	5.414647
3	West	4.754077

Question 7. Calculate the number of occurrences, mean, min, max of the speeding for each divison using SQL

Type in your code here

In [204]:

```
sql = "select division, count(*) as occurrences, avg(speeding), min(speeding), max(speeding)
df = ps.sqldf(sql)
df
```

Out[204]:

	division	occurrences	avg(speeding)	min(speeding)	max(speeding)
0	East North Central	5	4.106600	3.384	4.968
1	East South Central	4	4.533250	2.640	7.332
2	Middle Atlantic	2	5.446000	1.792	9.100
3	Mountain	8	5.812250	3.496	8.346
4	New England	6	4.084333	1.886	5.738
5	Pacific	5	5.949400	4.200	9.450
6	South Atlantic	9	5.030444	2.006	9.082
7	West North Central	7	4.293429	1.937	6.923
8	West South Central	4	6.333750	4.032	7.760

Question 8. Answer Question 7 but use Pandas functions

Type in your code here

In [220]:

```
crash.groupby(['division'])['speeding'].describe()
```

Out[220]:

division	count	mean	std	min	25%	50%	75%	max
East North Central	5.0	4.106600	0.665705	3.384	3.62500	3.9480	4.60800	4.968
East South Central	4.0	4.533250	1.985597	2.640	3.70950	4.0805	4.90425	7.332
Middle Atlantic	2.0	5.446000	5.167536	1.792	3.61900	5.4460	7.27300	9.100
Mountain	8.0	5.812250	1.524017	3.496	4.98875	5.4735	6.70950	8.346
New England	6.0	4.084333	1.299621	1.886	3.84550	4.0700	4.74600	5.738
Pacific	5.0	5.949400	2.382195	4.200	4.22400	4.4520	7.42100	9.450
South Atlantic	9.0	5.030444	2.546183	2.006	2.96400	4.2500	6.55200	9.082
West North Central	7.0	4.293429	2.005279	1.937	2.43850	4.8060	5.75550	6.923
West South Central	4.0	6.333750	1.637184	4.032	5.78400	6.7715	7.32125	7.760

Question 9. Calculate the Standard deviation, Skew and Kurtosis for the accidents, speeding, alcohol for each region

Type in your code here

In [206]:

```
crash.groupby(['region'])['accidents'].std()
```

Out[206]:

```
region
Midwest      3.588988
Northeast    3.077453
South        4.537037
West         3.325118
Name: accidents, dtype: float64
```

In [207]:

```
crash.groupby(['region'])['speeding'].std()
```

Out[207]:

```
region
Midwest      1.537451
Northeast    2.327773
South        2.220711
West         1.803138
Name: speeding, dtype: float64
```

In [208]:

```
crash.groupby(['region'])['alcohol'].std()
```

Out[208]:

```
region
Midwest      1.824432
Northeast    0.872950
South        1.773576
West         1.910356
Name: alcohol, dtype: float64
```

In [209]:

```
crash.groupby(['region'])['speeding'].skew()
```

Out[209]:

```
region
Midwest      0.121197
Northeast    1.051666
South        0.172035
West         0.697441
Name: speeding, dtype: float64
```

In [210]:

```
crash.groupby(['region'])['alcohol'].skew()
```

Out[210]:

```
region
Midwest      2.093467
Northeast    0.764250
South        0.330988
West         1.159752
Name: alcohol, dtype: float64
```

In [211]:

```
crash.groupby(['region'])['accidents'].skew()
```

Out[211]:

```
region
Midwest      0.950845
Northeast    0.777587
South       -1.125839
West         0.064819
Name: accidents, dtype: float64
```

In [212]:

```
crash.groupby(['region'])['speeding'].apply(pd.DataFrame.kurtosis)
```

Out[212]:

```
region
Midwest     -0.737061
Northeast    1.763520
South       -1.327699
West        -0.465455
Name: speeding, dtype: float64
```

In [213]:

```
crash.groupby(['region'])['alcohol'].apply(pd.DataFrame.kurtosis)
```

Out[213]:

```
region
Midwest      5.597372
Northeast    0.890501
South        2.125568
West         2.219220
Name: alcohol, dtype: float64
```

In [214]:

```
crash.groupby(['region'])['accidents'].apply(pd.DataFrame.kurtosis)
```

Out[214]:

```
region
Midwest      2.067092
Northeast    0.726713
South        1.860064
West        -1.048692
Name: accidents, dtype: float64
```

Question 10. Plot the histogram of accidents distribution for different region using 5 bins

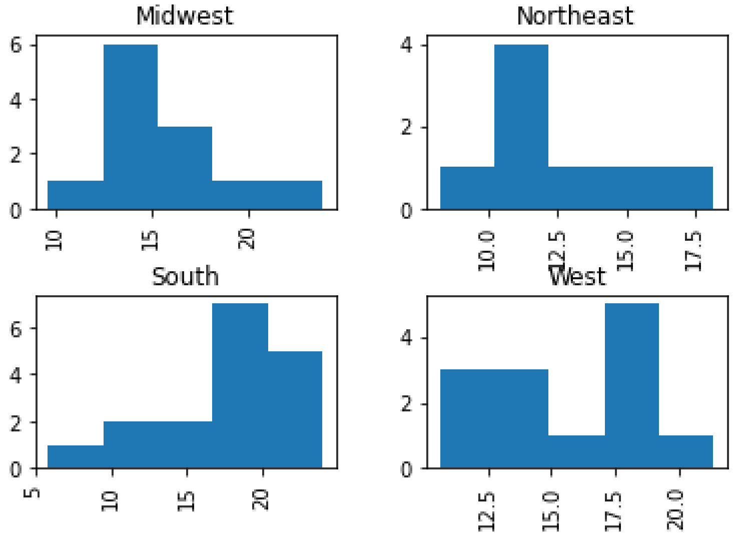
Type in your answer here

In [215]:

```
crash['accidents'].hist(bins=5,by=crash['region'])
```

Out[215]:

```
array([[<AxesSubplot:title={'center':'Midwest'}>,
        <AxesSubplot:title={'center':'Northeast'}>],
       [<AxesSubplot:title={'center':'South'}>,
        <AxesSubplot:title={'center':'West'}>]], dtype=object)
```



Question 11. How would you describe the accidents distribution for different region based on the Skew and Kurotsis?

(i.e. who has positive and negative skew and who has positive and negative Kurotsis)

Type in your answer here

In [ ]:

```
Midwest, Northeast and West is positive skew and south is negative Skew
```

In [ ]:

```
Midwest, Northeast and South is positive kurtosis and West is negative kurtosis
```

Question 12. Calculate correlation of all the factors among themselves and determine which factors among speeding, alcohol or ins_premium will affect accidents the most?

Type in your code and answers here

In [216]:

```
crash.corr()
```

Out[216]:

	accidents	speeding	alcohol	not_distracted	no_previous	ins_premium	in
accidents	1.000000	0.608632	0.850706	0.826209	0.956009	-0.174157	-
speeding	0.608632	1.000000	0.667377	0.585337	0.568831	-0.059254	-
alcohol	0.850706	0.667377	1.000000	0.730435	0.780696	-0.146921	-
not_distracted	0.826209	0.585337	0.730435	1.000000	0.745712	-0.157416	-
no_previous	0.956009	0.568831	0.780696	0.745712	1.000000	-0.119067	
ins_premium	-0.174157	-0.059254	-0.146921	-0.157416	-0.119067	1.000000	
ins_losses	-0.025507	-0.059624	-0.103849	-0.068733	0.007872	0.625381	

Now Load a different dataset (MPG dataset number 3)

In [217]:

```
df = pd.read_csv("mpg3.csv")
df.head()
```

Out[217]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	n
0	18.0	8	307.0	130.0	3504.0	12.0	70	usa	chev che ma
1	15.0	8	350.0	165.0	3693.0	11.5	70	usa	b sky
2	18.0	8	318.0	150.0	3436.0	11.0	70	usa	plym sat
3	16.0	8	304.0	150.0	3433.0	12.0	70	usa	rebe
4	17.0	8	302.0	140.0	3449.0	10.5	70	usa	tc

Question 13: Check to see if there are any missing values. Fix the missing values by imputing value from the mean.

Type your code to fix the missing values by imputing value from the mean

In [218]:

```

df.isnull().any()
```

Out[218]:

```
mpg              True
cylinders       False
displacement    False
horsepower       True
weight           True
acceleration    False
model_year      False
origin          False
name            False
dtype: bool
```

In [219]:

```
df['mpg'] = df['mpg'].fillna(df['mpg'].mean())
df['horsepower'] = df['horsepower'].fillna(df['horsepower'].mean())
df['weight'] = df['weight'].fillna(df['weight'].mean())
print(df.shape)
df.isnull().any()
```

```
(405, 9)
```

Out[219]:

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model_year      False
origin          False
name            False
dtype: bool
```

In [ ]:

```

```