Data exploration
Pandas and Visualization

In [6]:
```
%matplotlib inline
import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import os
import warnings
```

In [7]:
```
df=pd.read_csv("airline-safety.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	airline	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	in
0	Aer Lingus	320906734	2	0	0	
1	Aeroflot*	1197672318	76	14	128	
2	Aerolineas Argentinas	385803648	6	0	0	
3	Aeromexico*	596871813	3	1	64	
4	Air Canada	1865253802	2	0	0	

In [9]:
```
df.dtypes
```

Out[9]:
```
airline                    object
avail_seat_km_per_week      int64
incidents_85_99             int64
fatal_accidents_85_99       int64
fatalities_85_99            int64
incidents_00_14             int64
fatal_accidents_00_14       int64
fatalities_00_14            int64
dtype: object
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   airline                 56 non-null     object
 1   avail_seat_km_per_week  56 non-null     int64 
 2   incidents_85_99         56 non-null     int64 
 3   fatal_accidents_85_99   56 non-null     int64 
 4   fatalities_85_99        56 non-null     int64 
 5   incidents_00_14         56 non-null     int64 
 6   fatal_accidents_00_14   56 non-null     int64 
 7   fatalities_00_14        56 non-null     int64 
dtypes: int64(7), object(1)
memory usage: 3.6+ KB
```

In [13]:
```
df.memory_usage() # in bytes
```

Out[13]:
```
Index                     128
airline                   448
avail_seat_km_per_week    448
incidents_85_99           448
fatal_accidents_85_99     448
fatalities_85_99          448
incidents_00_14           448
fatal_accidents_00_14     448
fatalities_00_14          448
dtype: int64
```

In [14]:
```
df.memory_usage().sum()
```

Out[14]: 3712

In [15]:
```
df.describe()
```

Out[15]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_0
count	5.600000e+01	56.000000	56.000000	56.000000	56.00
mean	1.384621e+09	7.178571	2.178571	112.410714	4.12
std	1.465317e+09	11.035656	2.861069	146.691114	4.54
min	2.593733e+08	0.000000	0.000000	0.000000	0.00
25%	4.740362e+08	2.000000	0.000000	0.000000	1.00
50%	8.029089e+08	4.000000	1.000000	48.500000	3.00
75%	1.847239e+09	8.000000	3.000000	184.250000	5.25
max	7.139291e+09	76.000000	14.000000	535.000000	24.00

Statistical moments
Mean (1st moment)
Variance (2nd moment)
Skewness (3rd moment)
Kurtosis (4th moment)

In [16]:
```
df.mean()
```

Out[16]:
```
avail_seat_km_per_week    1.384621e+09
incidents_85_99           7.178571e+00
fatal_accidents_85_99     2.178571e+00
fatalities_85_99          1.124107e+02
incidents_00_14           4.125000e+00
fatal_accidents_00_14     6.607143e-01
fatalities_00_14          5.551786e+01
dtype: float64
```

In [17]:
```
df['incidents_00_14'].mean()
```

Out[17]: 4.125

In [18]:
```
df.var()
```

Out[18]:
```
avail_seat_km_per_week    2.147154e+18
incidents_85_99           1.217857e+02
fatal_accidents_85_99     8.185714e+00
fatalities_85_99          2.151828e+04
incidents_00_14           2.065682e+01
fatal_accidents_00_14     7.373377e-01
fatalities_00_14          1.239498e+04
dtype: float64
```

Skewness
Skewness is the measure of the symmetry of a distribution compared to standard normal distribution

+ive - right skewed (mean is to the right of mode/median). Long tail in the +ive direction.
0 - symmetric
-ive - left skewed (mean is to the left of mode/median). Long tail in the -ive direction.

In [20]:
```
df.skew()
```

Out[20]:
```
avail_seat_km_per_week    2.337911
incidents_85_99           4.731159
fatal_accidents_85_99     2.296527
fatalities_85_99          1.316283
incidents_00_14           2.210143
fatal_accidents_00_14     0.907261
fatalities_00_14          2.674622
dtype: float64
```

Kurtosis
Kurtosis is a measure of the flatness or peakedness of a distribution compared to the normal distribution.
+ive - Leptokurtosis (sharper/spikier peak compared to the normal dist.)
0 - Mesokurtic (normal dist.)
-ive - Platykurtic (flatter peak compared to the normal dist.) eg. Uniform distribution

In [21]:
```
df.kurtosis()
```

Out[21]:
```
avail_seat_km_per_week     6.012276
incidents_85_99           27.874700
fatal_accidents_85_99      6.324671
fatalities_85_99           0.834040
incidents_00_14            6.474408
fatal_accidents_00_14     -0.500499
fatalities_00_14           7.553167
dtype: float64
```

min / max / median

In [22]:
```
# min of each column
df.min()
```

Out[22]:
```
airline                   Aer Lingus
avail_seat_km_per_week     259373346
incidents_85_99                    0
fatal_accidents_85_99              0
fatalities_85_99                   0
incidents_00_14                    0
fatal_accidents_00_14              0
fatalities_00_14                   0
dtype: object
```

In [23]:
```
# max of each column
df.max()
```

Out[23]:
```
airline                   Xiamen Airlines
avail_seat_km_per_week         7139291291
incidents_85_99                        76
fatal_accidents_85_99                  14
fatalities_85_99                      535
incidents_00_14                        24
fatal_accidents_00_14                   3
fatalities_00_14                      537
dtype: object
```

In [24]:
```
# median of each column
df.median()
```

Out[24]:
```
avail_seat_km_per_week    802908893.0
incidents_85_99                   4.0
fatal_accidents_85_99             1.0
fatalities_85_99                 48.5
incidents_00_14                   3.0
fatal_accidents_00_14             0.0
fatalities_00_14                  0.0
dtype: float64
```

Correlation

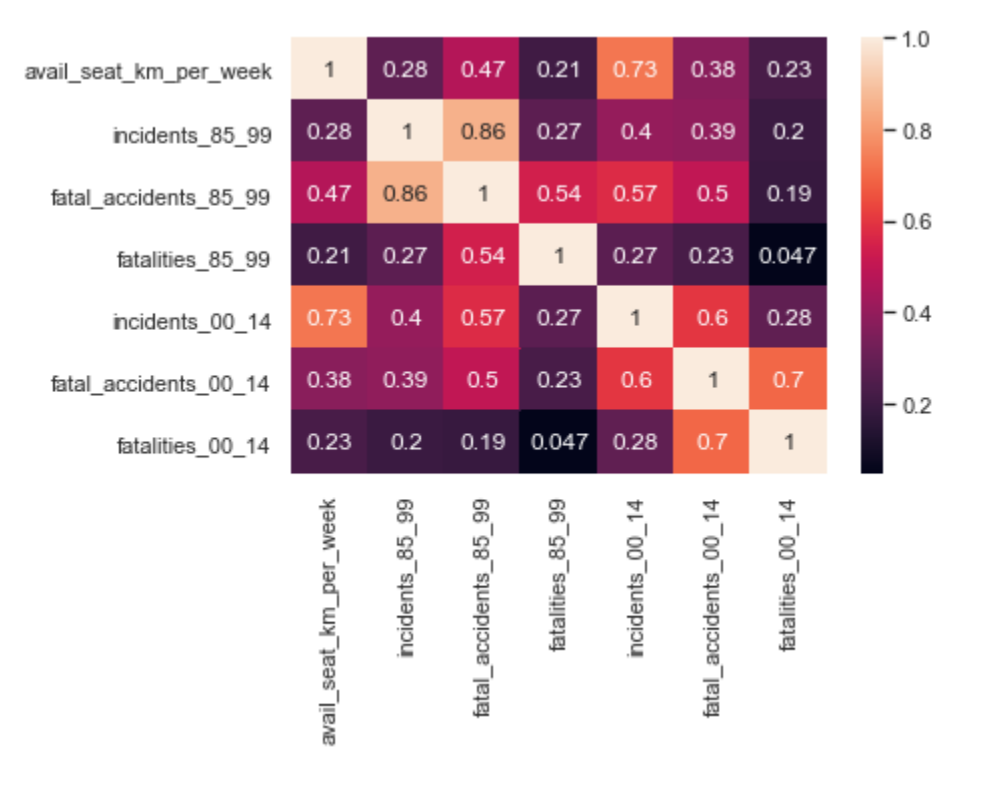
In [25]:
```
df.corr()
```

Out[25]:

	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_
avail_seat_km_per_week	1.000000	0.279538	0.468300	0.2
incidents_85_99	0.279538	1.000000	0.856991	0.2
fatal_accidents_85_99	0.468300	0.856991	1.000000	0.5
fatalities_85_99	0.209835	0.274394	0.540866	1.0
incidents_00_14	0.725917	0.403009	0.572923	0.2
fatal_accidents_00_14	0.375673	0.390249	0.498758	0.2
fatalities_00_14	0.228484	0.195337	0.186985	0.0

In [27]:
```
import seaborn as sns
sns.heatmap(df.corr(), annot=True)
```

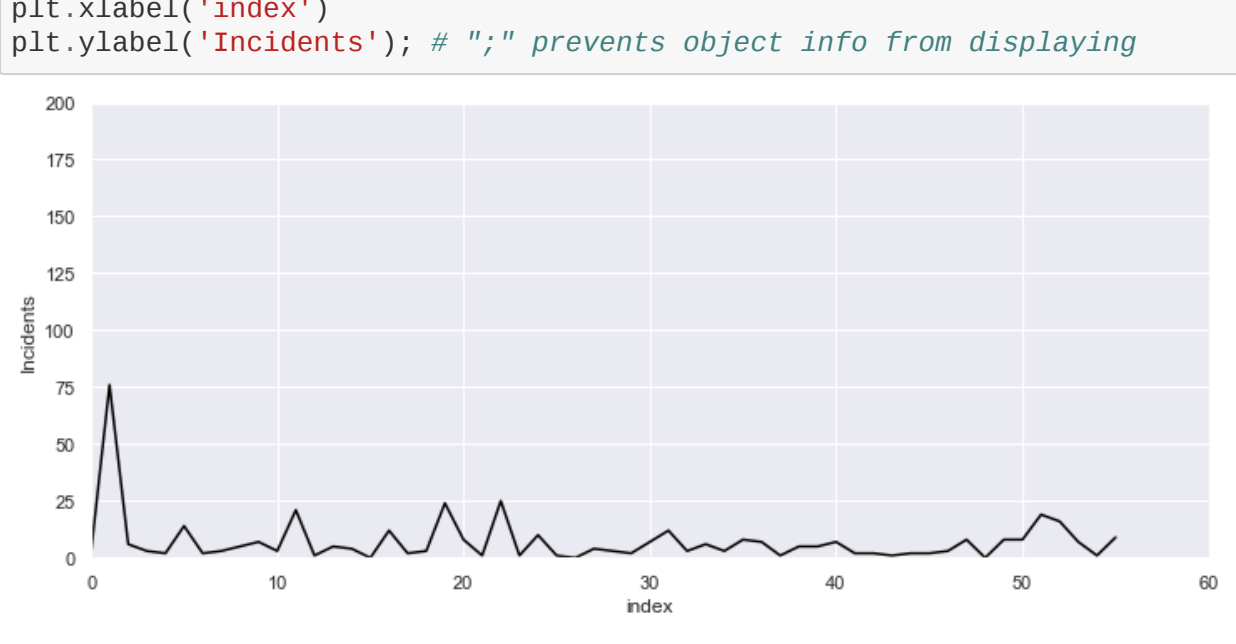
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x2191010f4c8>



Lineplot

In [31]:
```
# Plotting with index along the x-axis
df['incidents_85_99'].plot(figsize=(12, 5), color='black') # color and f
igsize changed

plt.xlim(0, 60) # range for x-axis
plt.ylim(0, 200) # range for x-axis
plt.xlabel('index')
plt.ylabel('Incidents'); # ";" prevents object info from displaying
```

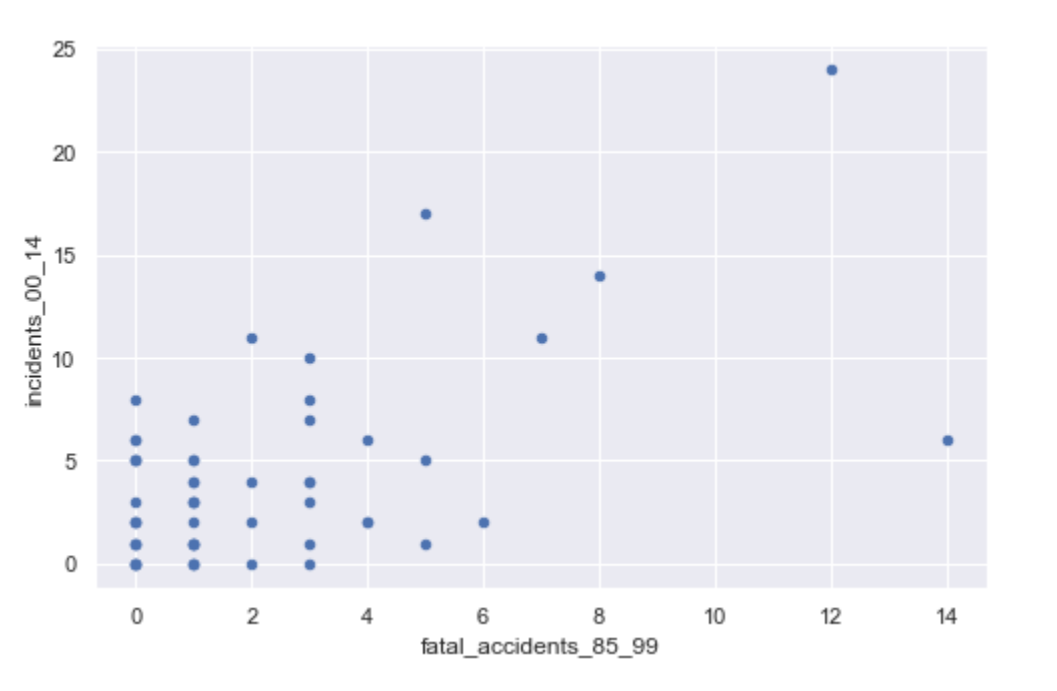


Scatter Plot

In [36]:
```
# plotting one variable against the other
df.plot.scatter('fatal_accidents_85_99', 'incidents_00_14', figsize=(8,
5))
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which sho
uld be avoided as value-mapping will have precedence in case its length
matches with 'x' & 'y'.  Please use a 2-D array with a single row if you
really want to specify the same RGB or RGBA value for all points.
```

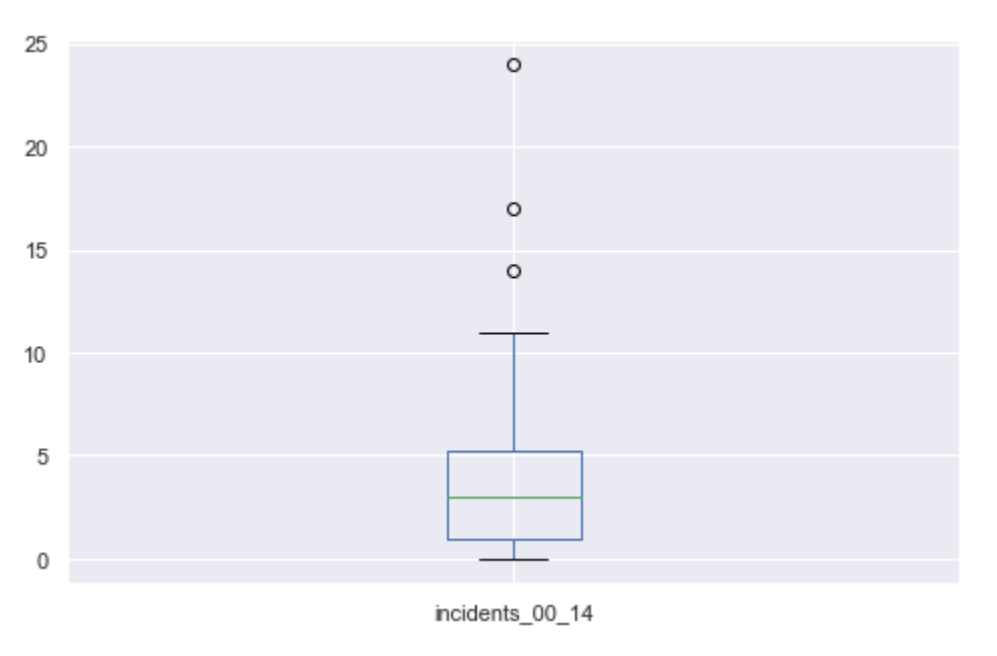
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x21910f11208>



Boxplot

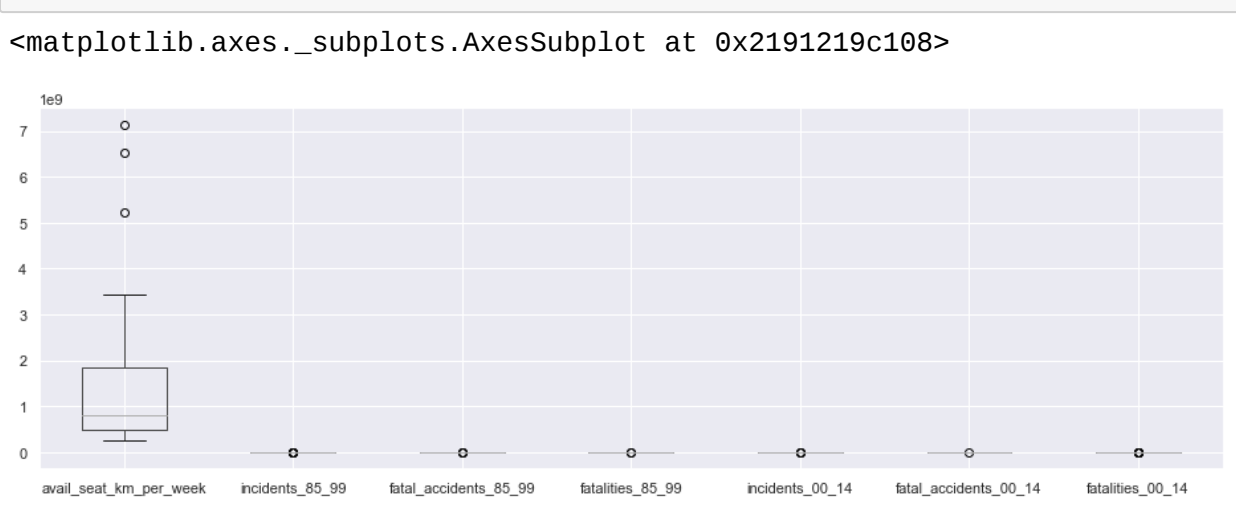
In [39]:
```
df['incidents_00_14'].plot.box(figsize=(8, 5))
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x2191222c208>



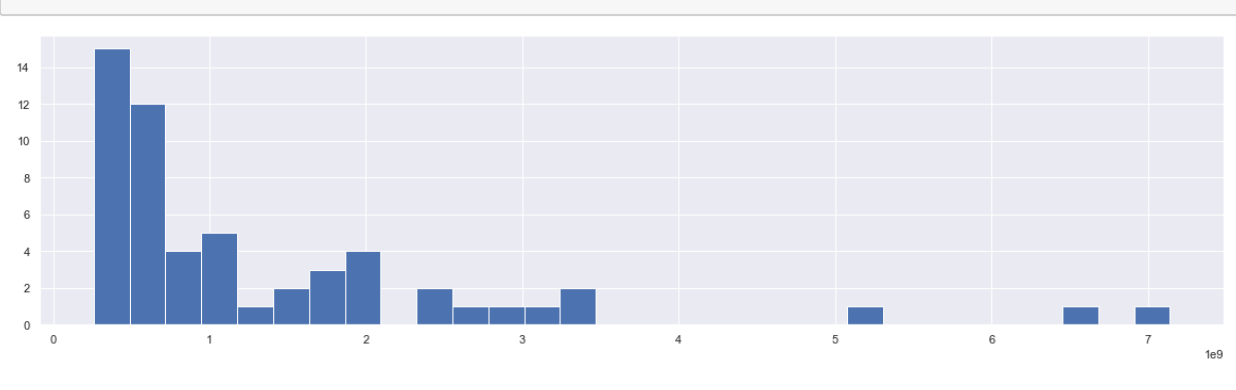
In [41]:
```
df.boxplot(figsize=(16, 5))
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x2191219c108>



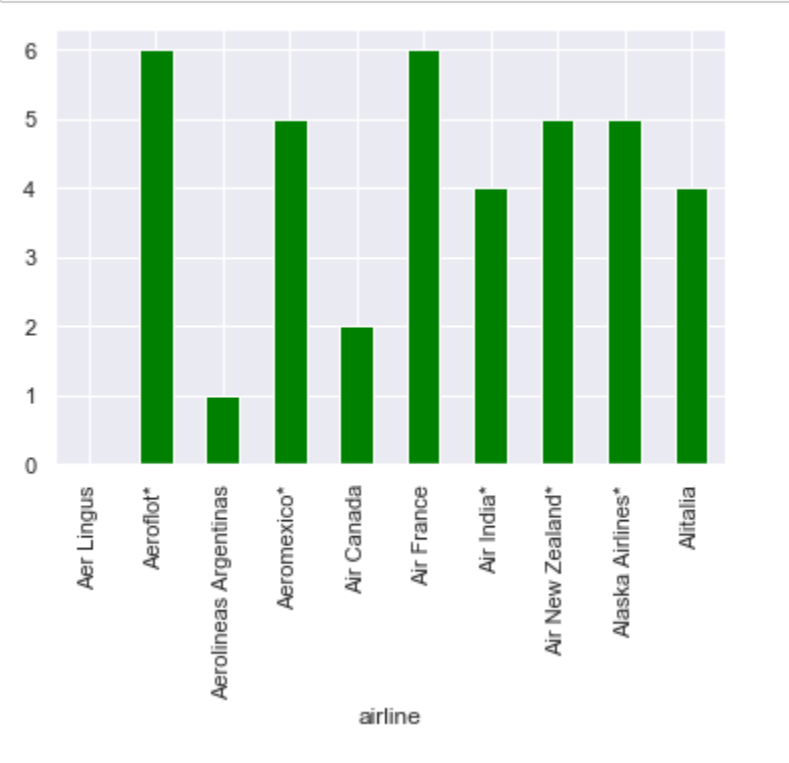
Histogram

In [43]:
```
df['avail_seat_km_per_week'].hist(bins=30, figsize=(20, 5));
```



Barplot

In [49]:
```
df_avg_BP = df.groupby('airline')['incidents_00_14'].mean()
df_avg_BP[:10].plot.bar(color='green');
```



In [ ]:

Data Cleaning

In [4]:
```
import pandas as pd
data = pd.read_csv('airline-safety.csv')
data.head()
```

Out[4]:

	airline	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	in
0	Aer Lingus	320906734	2	0	0	
1	Aeroflot*	1197672318	76	14	128	
2	Aerolineas Argentinas	385803648	6	0	0	
3	Aeromexico*	596871813	3	1	64	
4	Air Canada	1865253802	2	0	0	

In [5]:
```
data.tail()
```

Out[5]:

	airline	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	in
51	United / Continental*	7139291291	19	8	319	
52	US Airways / America West*	2455687887	16	7	224	
53	Vietnam Airlines	625084918	7	3	171	
54	Virgin Atlantic	1005248585	1	0	0	
55	Xiamen Airlines	430462962	9	1	82	

In [8]:
```
data.isna().any()
```

Out[8]:
```
airline                  False
avail_seat_km_per_week   False
incidents_85_99          False
fatal_accidents_85_99    False
fatalities_85_99         False
incidents_00_14          False
fatal_accidents_00_14    False
fatalities_00_14         False
dtype: bool
```

In [9]:
```
data.isna().sum()
```

Out[9]:
```
airline                  0
avail_seat_km_per_week   0
incidents_85_99          0
fatal_accidents_85_99    0
fatalities_85_99         0
incidents_00_14          0
fatal_accidents_00_14    0
fatalities_00_14         0
dtype: int64
```

In [10]:
```
data.isna().any().sum()
```

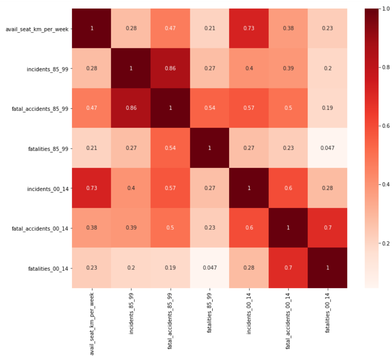
Out[10]:
```
0
```

Data selection Using Filter Method

In [16]:
```
import pandas as pd
data = pd.read_csv('airline-safety.csv')
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
```

In [21]:
```
df = pd.read_csv('airline-safety.csv')
```

In [22]:
```
plt.figure(figsize=(12,10))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [24]:
```
cor_target = abs(cor["avail_seat_km_per_week"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

Out[24]:
```
avail_seat_km_per_week    1.000000
incidents_00_14           0.725917
Name: avail_seat_km_per_week, dtype: float64
```

Data Split Using K-Fold Validation

In [69]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm

X, y = datasets.load_iris(return_X_y=True)
X.shape, y.shape
((150, 4), (150,))
```

Out[69]:
```
((150, 4), (150,))
```

In [70]:
```
X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.4, random_state=0)
```

In [71]:
```
X_train.shape, y_train.shape
((90, 4), (90,))
X_test.shape, y_test.shape
((60, 4), (60,))

clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[71]:
```
0.9666666666666667
```

In [75]:
```
from sklearn.datasets import make_classification
# define dataset
X, y = make_classification(n_samples=100, n_features=20, n_informative=15, n_redundant=5, random_state=1)
# summarize the dataset
print(X.shape, y.shape)
```

```
(100, 20) (100,)
```

In [77]:
```
# evaluate a logistic regression model using k-fold cross-validation
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
# create dataset
X, y = make_classification(n_samples=100, n_features=20, n_informative=15, n_redundant=5, random_state=1)
# prepare the cross-validation procedure
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = LogisticRegression()
# evaluate model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))

# evaluate a logistic regression model using k-fold cross-validation
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
# create dataset
X, y = make_classification(n_samples=100, n_features=20, n_informative=15, n_redundant=5, random_state=1)
# prepare the cross-validation procedure
cv = KFold(n_splits=10, random_state=1, shuffle=True)
# create model
model = LogisticRegression()
# evaluate model
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.850 (0.128)
Accuracy: 0.850 (0.128)
```

In [ ]: