In [19]:
```
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['species'] = iris.target
```

In [20]:
```
data.shape
```

Out[20]:
```
(150, 5)
```

In [21]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [22]:
```
# Statistical summary
data.describe()
```

Out[22]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [23]:
```
# Breakdown by class
data['species'].value_counts()
```

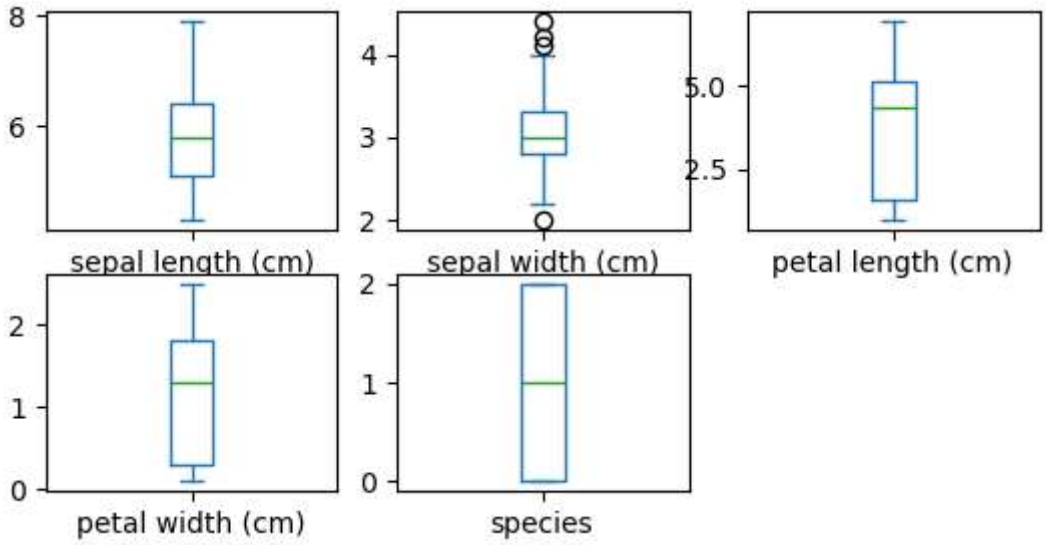
Out[23]:
```
species
0    50
1    50
2    50
Name: count, dtype: int64
```

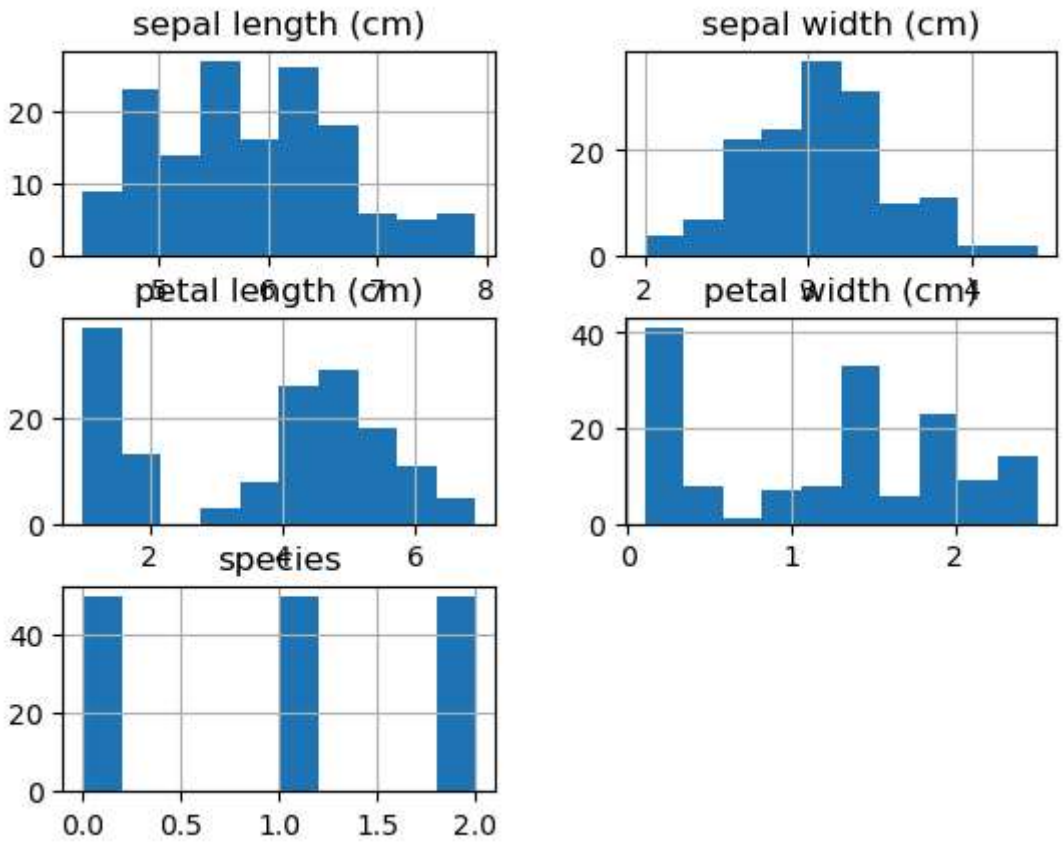
In [24]:
```
# Univariate Plots

import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot
data.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
plt.show()

# Histogram
data.hist()
plt.show()
```

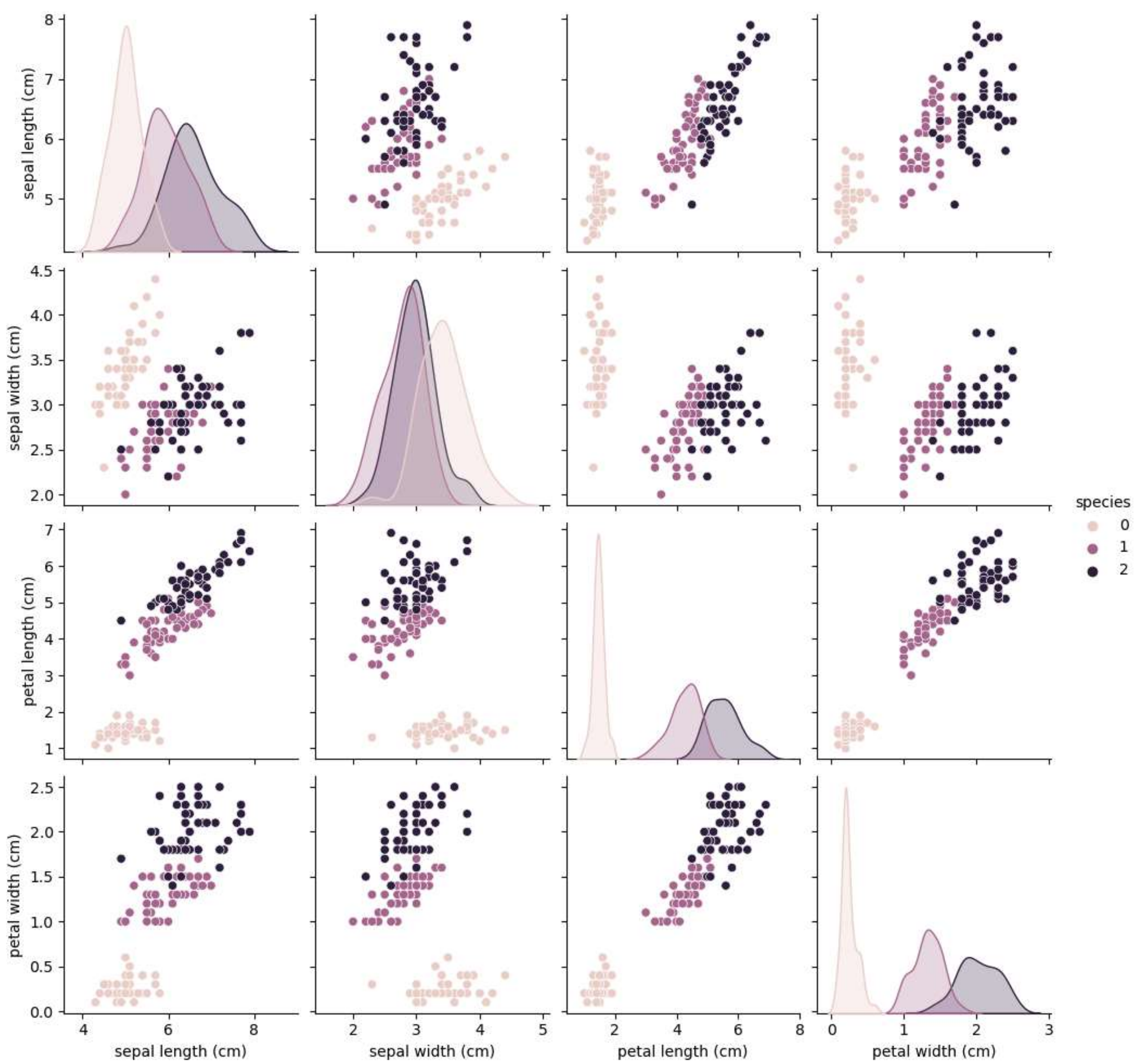




In [25]:
```
# Multivariate Plots

# Pairplot
sns.pairplot(data, hue='species')
plt.show()
```

```
C:\Users\User\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\User\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\User\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\User\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWar
ning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [26]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Split dataset into training and validation sets
X = data.drop(columns=['species'])
y = data['species']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_

# Set-Up the Test Harness Using 10-Fold Cross-Validation
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

# List of models to evaluate
models = []
models.append(('LR', LogisticRegression(max_iter=500)))  # Increased max_iter
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# Evaluate each model
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='a
    results.append(cv_results)
    names.append(name)
    print(f"{name}: {cv_results.mean():.3f} ({cv_results.std():.3f})")
```

```
LR: 0.967 (0.041)
LDA: 0.975 (0.038)
KNN: 0.958 (0.042)
CART: 0.942 (0.053)
NB: 0.950 (0.055)
SVM: 0.967 (0.041)
```

In [27]:
```
# Predictions

# Train the best model
model = LinearDiscriminantAnalysis()
model.fit(X_train, y_train)

# Make predictions on the validation set
predictions = model.predict(X_val)

# Evaluate predictions
from sklearn.metrics import accuracy_score, classification_report
print(accuracy_score(y_val, predictions))
print(classification_report(y_val, predictions))
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```