parsing the nl using nltk

In [1]:
```
tokens = 'employee has the highest salary'.split()
from nltk import load_parser
cp = load_parser('grammars/book_grammars/thiru_siddharth.fcfg')
for tree in cp.parse(tokens):
    print(tree)
```

```
(S[SQL=(SELECT, , , Employee_Name FROM Employee_Salaries WHERE Salary =
(SELECT Max(Salary) FROM Employee_Salaries), )]
  (NP[SQL='SELECT'] employee)
  (VP[SQL=(, , Employee_Name FROM Employee_Salaries WHERE Salary = (SEL
ECT Max(Salary) FROM Employee_Salaries), )]
    (V[SQL=''] has)
    (NP[SQL=(, Employee_Name FROM Employee_Salaries WHERE Salary = (SEL
ECT Max(Salary) FROM Employee_Salaries), )]
      (DET[SQL=''] the)
      (JJS[SQL='Employee_Name FROM Employee_Salaries WHERE Salary = (SE
LECT Max(Salary) FROM Employee_Salaries)']
        highest)
      (N[SQL=''] salary))))
```

giving tags to the nl tokens

In [2]:
```
from nltk.tag import pos_tag

sentence = "employee has the highest salary"
tagged_sent = pos_tag(sentence.split())
print(tagged_sent)
```

```
[('employee', 'NN'), ('has', 'VBZ'), ('the', 'DT'), ('highest', 'JJS'),
('salary', 'NN')]
```

parsed nl will look like this in a syntactic tree

In [4]:
```
import nltk
nltk.data.show_cfg('grammars/book_grammars/thiru_siddharth.fcfg')
```

```
% start S
S[SQL=(?np + ?vp)] -> NP[SQL=?np] VP[SQL=?vp]
VP[SQL=(?v + ?np)] -> V[SQL=?v] NP[SQL=?np]
NP[SQL=(?det + ?jjs + ?n)] -> DET[SQL=?det] JJS[SQL=?jjs] N[SQL=?n]
DET[SQL=''] -> 'Which' | 'the' | 'What'
JJS[SQL='Employee_Name FROM Employee_Salaries WHERE Salary = (SELECT Ma
x(Salary) FROM Employee_Salaries)'] -> 'highest'
JJS[SQL='AVG(Salary) FROM Employee_Salaries'] -> 'average'
N[SQL=''] -> 'salary'
V[SQL=''] -> 'is' | 'are' | 'has'
NP[SQL='SELECT'] -> 'employees' | 'employee'
P[SQL=''] -> 'of' | 'in'
```

converting natural language to sql queries

In [5]:
```
import nltk
queries = ['employees has the average salary','employee has the highest
salary']
for query in queries:
    from nltk import load_parser
    cp = load_parser('grammars/book_grammars/thiru_siddharth.fcfg') # w
e can use trace= 1/2/3
                                                            #here to see
syntactic parsing
    trees = list(cp.parse(query.split()))
    answer = trees[0].label()['SQL']
    answer = [s for s in answer if s]
    q = ' '.join(answer)
    print(q)
```

```
SELECT AVG(Salary) FROM Employee_Salaries
SELECT Employee_Name FROM Employee_Salaries WHERE Salary = (SELECT Max
(Salary) FROM Employee_Salaries)
```

testing our outputs

In [6]:
```
import pandas as pd

data = pd.read_excel (r'C:/Users/thiru/Downloads/Employee Salaries.xls
x')

Employee_Salaries = pd.DataFrame(data, columns= ['Salary','State','Empl
oyee Name',])
Employee_Salaries = Employee_Salaries.rename(columns={"Employee Name":
"Employee_Name"})

from pandasql import sqldf

query1 = sqldf("SELECT AVG(Salary) FROM Employee_Salaries")
query2 = sqldf("SELECT Employee_Name FROM Employee_Salaries WHERE Salar
y = (SELECT Max(Salary) FROM Employee_Salaries)")
print(f'average salary of emplyees:{query1}')
print(f'highest paid emplyee:{query2}')
```

```
average salary of emplyees:   AVG(Salary)
0      49000.0
highest paid emplyee:   Employee_Name
0  Ranjeet Kumar
```