```
import sys
import sqlite3
import re

db_name = 'produce.db'
if len(sys.argv) <= 2 :
    exit("must provide db name followed by a file to parse")
db_name = sys.argv[1]

conn = sqlite3.connect(db_name)
c=conn.cursor()

def request_usr_fix(regionName, data):
    print('ERROR: ', regionName, data)
    exit()
    return None, None

start_dates = {'January':1, 'February':2,
'March':3,'April':4,'May':5,'June':6,'July':7,'August':8,'September':9,'October':10,
'November':11,'December':12, 'Spring':4, 'Summer':7, 'Fall':10,'Winter':1}
end_dates = {'January':1, 'February':2,
'March':3,'April':4,'May':5,'June':6,'July':7,'August':8,'September':9,'October':10,
'November':11,'December':12, 'Spring':7, 'Summer':10, 'Fall':1,'Winter':4}

# valid inputs:
#  'season' and/through 'season'
#  'month' and/through 'month
#  'month'
#  'season'
#  'year-round'
def insert_produce(regionName, data_line):
    if not regionName or not data_line:
        return
    #data_line
    data = data_line.split(',')
    if len(data)!=2: #exta commas in this line
        #data = [re.match(r'^(.*[,])[^,]*$',data_line).group(0), #everything before
first comma
        #        re.match(r'[^,]+$',data_line)] #everything after last comma
        return #throw it out!
    else:
        produce_name = re.sub(r"'","\\'",data[0].strip())
        if data[1] == "\n":
            return
    try:
        cleaned_date = re.sub('\(.*\)*','',data[1]) #remove comments at end one line
    except IndexError:
        return # No date range Specified
    #If a string like below appears:
    # Parsnips, April and May and again October through December
    #Then add the produce twice into db
    if re.search(r'\b(and again)\b', cleaned_date):
        split_data = cleaned_date.split('and again')
        if len(split_data) != 2 :
            request_usr_fix(regionName, data_line)
            return
        left = ''.join([produce_name, ', ', split_data[0]])
        right = ''.join([produce_name, ', ', split_data[1]])
        insert_produce(regionName, left)
        insert_produce(regionName, right)
        return
    date_range = re.sub(r'\b(through|and|though|into)\b', '-', cleaned_date) #get a
range
    if re.search(r'(year-round)', date_range):
        date_range = 'January-December'

    #Remove extraneous words and misspellings. This gets nasty, but it works!
    date_range = re.sub(r'\b(mid-)', '', date_range, flags=re.IGNORECASE) #remove
these sequence
    date_range = re.sub(r'\b(mis-)', '', date_range, flags=re.IGNORECASE) #remove
these sequence
    date_range = re.sub(r'\b(early)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(late)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(end of)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(harvested in)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(in)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(various)\b', '', date_range, flags=re.IGNORECASE)
    date_range = re.sub(r'\b(Septmeber)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
    date_range = re.sub(r'\b(Septmber)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
    date_range = re.sub(r'\b(Sept)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
    date_range = re.sub(r'\b(Novemeber)\b', 'November', date_range,
flags=re.IGNORECASE)#Novemeber
    date_range = re.sub(r'\b(p\])', '', date_range, flags=re.IGNORECASE)#p]
    date_range = re.sub(r'\b(fresh)', '', date_range, flags=re.IGNORECASE)#fresh
    date_range = re.sub(r'\b(best)', '', date_range, flags=re.IGNORECASE)#BEST
    date_range = re.sub(r'\b(into)', '', date_range, flags=re.IGNORECASE)#BEST
    #remove whitespace
    date_range = re.sub(r'\s+', '', date_range)

    start_id=0
    end_id = 0
    if re.search(r'(-)', date_range) : #it contains a range like month-month or
season-season
        date_range = re.sub(r'-+','-',date_range)
        split_data = date_range.split('-')
        start = split_data[0].capitalize()
        end = split_data[1].capitalize()
    else:
        start = date_range.capitalize()
        end = start
    try:
        start_id = start_dates[start]
        end_id = end_dates[end]
    except KeyError:
        print(date_range, start, end)
        request_usr_fix(regionName, data_line)
        return

    #now we can insert it!!
    try:
        #Insert Region
        s = ["INSERT INTO regions(name) VALUES( '", regionName, "')"]
        c.execute(''.join(s))
    except sqlite3.IntegrityError:
        pass #We've already added this region, just skip it.
    except sqlite3.OperationalError:
        print('ERROR',regionName)
    try :
        #Insert Produce
        s = ["INSERT INTO produces(name) VALUES( '", produce_name, "')"]
        c.execute(''.join(s))
    except sqlite3.IntegrityError:
        pass
    except sqlite3.OperationalError:
        print('DING',produce_name)

    c.execute("SELECT produceid FROM produces WHERE produces.name = ?",
(produce_name,) )
    produce_id = c.fetchone()[0] #returns a tuple with first element the produce id
    c.execute("SELECT regionid FROM regions WHERE regions.name = ?", (regionName,) )
    region_id = c.fetchone()[0] #returns a tuple with first element the region id
    #try:
    c.execute("INSERT INTO data(produceid, regionid, start, end) VALUES(?,?,?,?)",
(produce_id, region_id, start_id, end_id) )
    #except sqlite3.InterfaceError:
    #    print("BAD INSERT: ",(produce_id, region_id, start_id, end_id))
        #exit()
    #Insert this produce data
    #s = ["INSERT INTO data VALUES( ", produce_name, "')"]
    #c.execute(''.join(s))

#region data should be a text file named the region only, no extension
for i in range(2, len(sys.argv)):
    #print(sys.argv[i])
    f = open( sys.argv[i], "r")
    region = sys.argv[i].split('/')[-1]
    for line in f:
        if len(line) >2 :
            insert_produce(region, line)
    f.close()

conn.commit()

#Test
#c.execute("SELECT regionid, produceid FROM data NATURAL JOIN regions")
#c.execute("SELECT produceid FROM data, regions WHERE data.regionid =
regions.regionid")
#print(c.fetchall())

print ("done")

conn.close()
```