Q1.

1. Write a function named `freq_table()` that generates a frequency table for any column in our iOS apps data set.
    - The function should take the index number of a column in as an input (name the parameter as you want).
    - Inside the function's body:
        - Loop through the `apps_data` data set (don't include the header row) and extract the value you want by using the parameter (which is expected to be an index number).
        - Build the frequency table as a dictionary.
    - The function should return the frequency table as a dictionary.
2. Use the `freq_table()` function to generate a frequency table for the `user_rating` column (the index number of this column is `7`). Store the table in a variable named `ratings_ft`.

Script.py

```
opened_file = open('AppleStore.csv', encoding = 'utf8')
from csv import reader
read_file = reader(opened_file)
apps_data = list(read_file)
```

Q2.

1. Update the current `freq_table()` function to make it more reusable.
    - The function should take in two inputs this time: a data set and the index of a column (name the parameters as you want).
    - Inside the function's body:
        - Loop through the data set using that parameter which is expected to be a data set (a list of lists). For each iteration, select the value you want by using the parameter which is expected to be an index number.
        - Build the frequency table as a dictionary.
    - The function should return the frequency table as a dictionary.

2. Use the updated `freq_table()` function to generate a frequency table for the `user_rating` column (the index number of this column is `7`). Store the table in a variable named `ratings_ft`.

Script.py

```
opened_file = open('AppleStore.csv', encoding = 'utf8')
from csv import reader
read_file = reader(opened_file)
apps_data = list(read_file)

def freq_table(index):
        frequncy_table = {}

        for row in apps_data[1:]:
                value = row[index]
                if value in frequncy_table:
                        frequncy_table[value] += 1
                else:
                        frequncy_table[value] = 1

        return frequncy_table
```

Q3.

1. Add an extra parameter to the `open_dataset()` function (already written in the code editor) such that it only returns data sets without header rows.
    - If the parameter indicates that data set has a header row, the function removes the header row before returning the data set.

- Else (if the parameter doesn't indicate that data set doesn't have a header row), the function returns the data set as it is.
- It's up to you whether you use default arguments or not.

2. Use the updated `open_dataset()` function to open the `AppleStore.csv` file — recall that the `AppleStore.csv` data set has a header row. Assign the data set to a variable named `apps_data`.

Script.py

```
def open_dataset(file_name='AppleStore.csv'):
    opened_file = open(file_name, encoding = 'utf8')
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)

    return data
```