Midterm Project

COVIS

COVID-19 Vaccination Information System

Basic description

- This is a covid-19 vaccination records information & management application. (***manage_covis_records.py)***
- All data is stored in file: default filename: **covidVaccineData.csv**
- The application can be invoked with multiple options and modes.
- Next slides will explain the functions and features of this program as well as the desired input and output.
- Submit
  - **covidVaccineData.csv**
  - ***manage_covis_records.py***
  - ***gen_covis_data.py***

People attribute

All the current data is stored in a csv files. Following are the attributes. The attributes with * are required attributes. All attributes are known and described and entered and searched using their key names marked by ^

1. Person number* (Pno^): 8 digit number ***(Default: largest+1)***
2. Record date*(Rdt^): Date format <Month, dd, year>, E.g. June, 05, 2020
3. Age*^: Float
4. Gender (Gen^): {U: Unspecified, M: Male, F: Female} ***(default: U)***
5. Vaccination Status*(VStatus^): {R:Registered but not Vaccinated, P:Partially Vaccinated, F:Fully Vaccinated} ***(default: NV)***
6. Vaccine type: (VType^): {Pfizer:[2, 21], Moderna:[2, 28], J&J:[1, 0]} ***(default: empty)***
7. First Vaccine date (Vdt1^): Date format <Month, dd, year>, E.g. June, 05, 2020
8. Second Vaccine date (Vdt2^): Date format <Month, dd, year>, E.g. June, 05, 2020
9. Email^: String ***(default: empty)***
10. Notes^: ***(default: empty)***

Attribute validation

- Person number should be a unique 8 digit number. No two people should have the same membership number.
- All dates should be a valid dates.
- Vaccine Status has to be one of the {R, P, F}
- Vaccination dates should be a valid date after January 2020. Neither of the Rdt, Vdt1 or Vdt2 can be in the future.
- Email if given, needs to have the format <username@domain name> Domain name should be <string.string>
- Create a dictionary objects to represent the VStatus, Gen, VType as shown on slide 3. Maintain these dictionary objects so that new types can be added
    - E.g. VType is a dict object with key being Vaccine name and attributes being list of vaccine requirements attribute (required number of shots, minimum number of days between the shots). dict{Pfizer: [2, 21], Moderna: [2, 28], J&J:[1, 0]}
- The dates Vdt1, Vdt2 should be validated according to the VStatus. E.g. If VStatus is R, all VType, Vdt1 and Vdt2 should be empty. If the VStatus is P or F, VType cannot be empty and the Vdt1 and/or Vdt2 should satisfy the vaccine requirements.

Task 1: Application to Manage and Access data (60 points)

Generate an application python program ***manage_covis_records.py***.

- The application can be run as
  a. "**python *manage_covis_records.py -m <modename> -f <filename>***",
- If no file argument is given, use default filename as ***covidVaccineData.csv.***
- Following are modes
  a. <search> : Search a record
  b. <add> Add a new vaccination record
  c. <addA> Add or change dict attributes
  d. <change> Modify person's data
  e. <import> Import members (csv or a text file)
- Each mode and allotted points is described in details in next slides.
- **10 points are allotted for sanity of the program and correct overall operation**.
  a. Capture common user errors for data types and not using the arguments correctly.

<u>Search a record (10 pts):</u> -m search

1. Ask for the search criteria: (using any attribute with its key name). E.g. Pno: 345678 or VStatus: P. If no criteria is stated, exit with "help text about the program"
2. If there are multiple people who match the criteria, all of them are printed to the screen. If the number of people fitting the criteria is more than 10, please print a warning that "more than 10 matching the criteria, print (y/n)"
3. The user can always give multiple search criteria altogether in one string, such as "Gen:M, VStatus:F" etc. Do not worry about spaces.
4. Press 'e' to exit and "enter" for next if the user chooses to print them 10 by 10. You can always choose not to have this option.

Add a new vaccination record/s (10 pts): -m add <no>

1. <no> is number of records to add. If not given, use by default 1.
2. The program will ask attributes one after the other except the Pno.
3. The program will validate the format of the value entered for each attribute, give user a warning and remind the required format and give an opportunity to enter the same attribute again.
4. The user can simply press enter not to specify a value for non required attributes or attributes with default values. Look at the default values for the attributes on slide 3.
5. A new record should be successfully added to the file only if it passes all the checks and validations stated in slide 4, with a new Pno which is the largest no +1.

addA (Add or modify attributes) (10 pts)

Add a new vaccination type: -m addA VType <name of the vaccine> <no of shots> <no of days>

The function for adding a new vaccination type should convert the arguments into a dict type and append the existing dictionary object. If the arguments are not specified i.e. (-m addv) print help text stating how the mode should be used. Make sure you check the argument data type.

Add a new VStatus: -m addA VStatus <Keyword> <Value>:

E.g. -m addA VStatus RS “Registered and scheduled”

<u>Modify person status or data (10 pts):</u> -m change

This will again first ask for the person search criteria. It will print the person's data to the screen and ask for the attribute to be changed. You may use a menu if you like. Once you enter the attribute by its key name, display the current value and then ask the user to enter the new value for the attribute. The new value would go through same validation and checks as stated before. Attributes which cannot be changed are Pno, Vdt1 (if not empty), Vdt2 (if not empty)

<u>Import data(10 pts):</u> -m import <FileToImport>

This will allow user to import a new csv or a text file <**FileToImport**> (in the correct format) to import new records or modify existing records. The new file records will go through the same checks and validations. As part of processing the new file, the following would be reported to the user before appending the new records

- Follow all checks and balances for file check.
- If there are people with existing Pno in the new file, warn user about modifying the existing record. Do not overwrite the Notes string, concatenate it with the Notes in the new file.
- Print total number of new patient records detected. Append these records to the existing file.

Task 2: Average information (5 pts)

- “**python *manage_covis_records.py -i”*** will print the following:
  a. Number of total doses administered: <>
  b. Total number of people fully vaccinated: <>
  c. Total number of people partially vaccinated: <>
  d. Current vaccination rate: <> Total vaccinated / Total registered
  e. Average age: <> This is the average age of fully vaccinated people
  f. %Female: <> & %Male: <> This is percentage of total vaccinated females and males. It the gender is unspecified, do not count those cases.
- “**python *manage_covis_records.py -h”*** will print help about how to use the application
  a. It is also necessary to print short help text after running each mode about that mode.

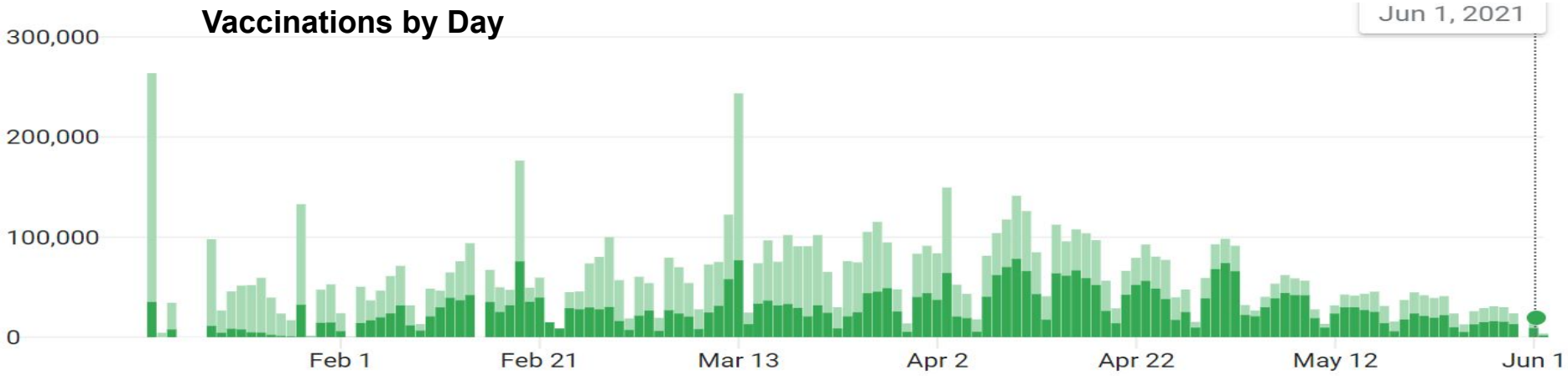
Task 3: Graphs (25 pts)

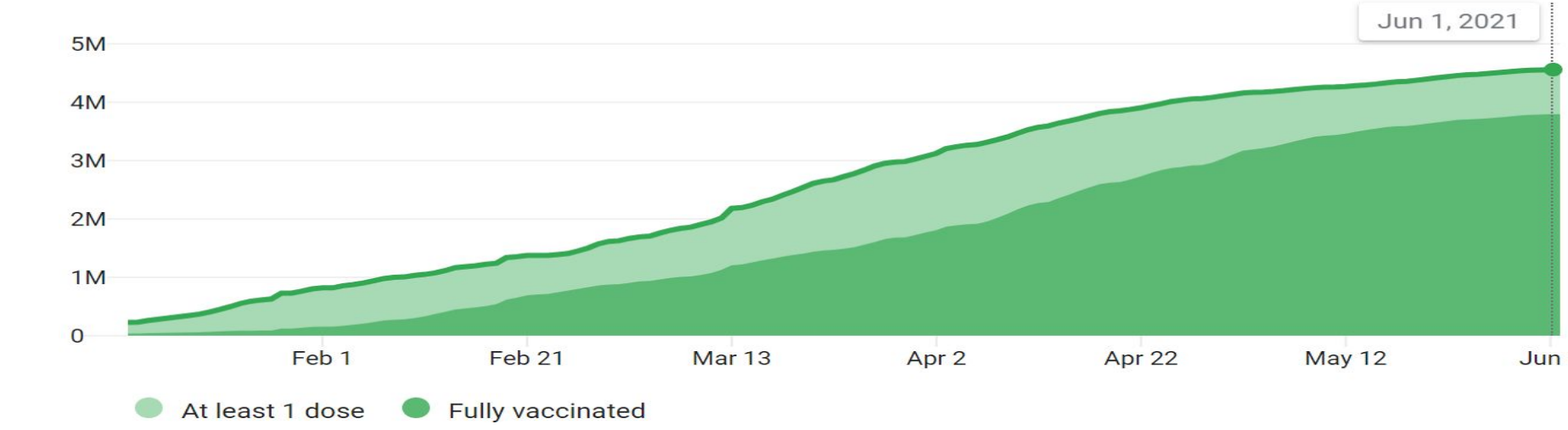
The following graphs will be plotted when ***manage_covis_records.py*** is invoked with given arguments. "***python manage_covis_records.py -g <mode>***" where mode can be one of the following:

- ***<day> vaccinations by Day***: This is a double graph showing on the top a subgraph with number of new vaccinations each day (Partially vaccinated and fully vaccinated) and bottom subgraph with total number of vaccinations until each day (partially and fully vaccinated.)
- ***<age>vaccinations by Age***: This will be dual bar graph showing of total number of partially and fully vaccinated people in following age category, 12 to 17, 17 to 24, 24 to 34, 35 to 44, 44 to 54, 54 to 64, 64 to 74, 74 to 84, > 84
- ***<type>vaccinations by VType:*** This is a Pie chart showing number of doses per vaccine type in VType.

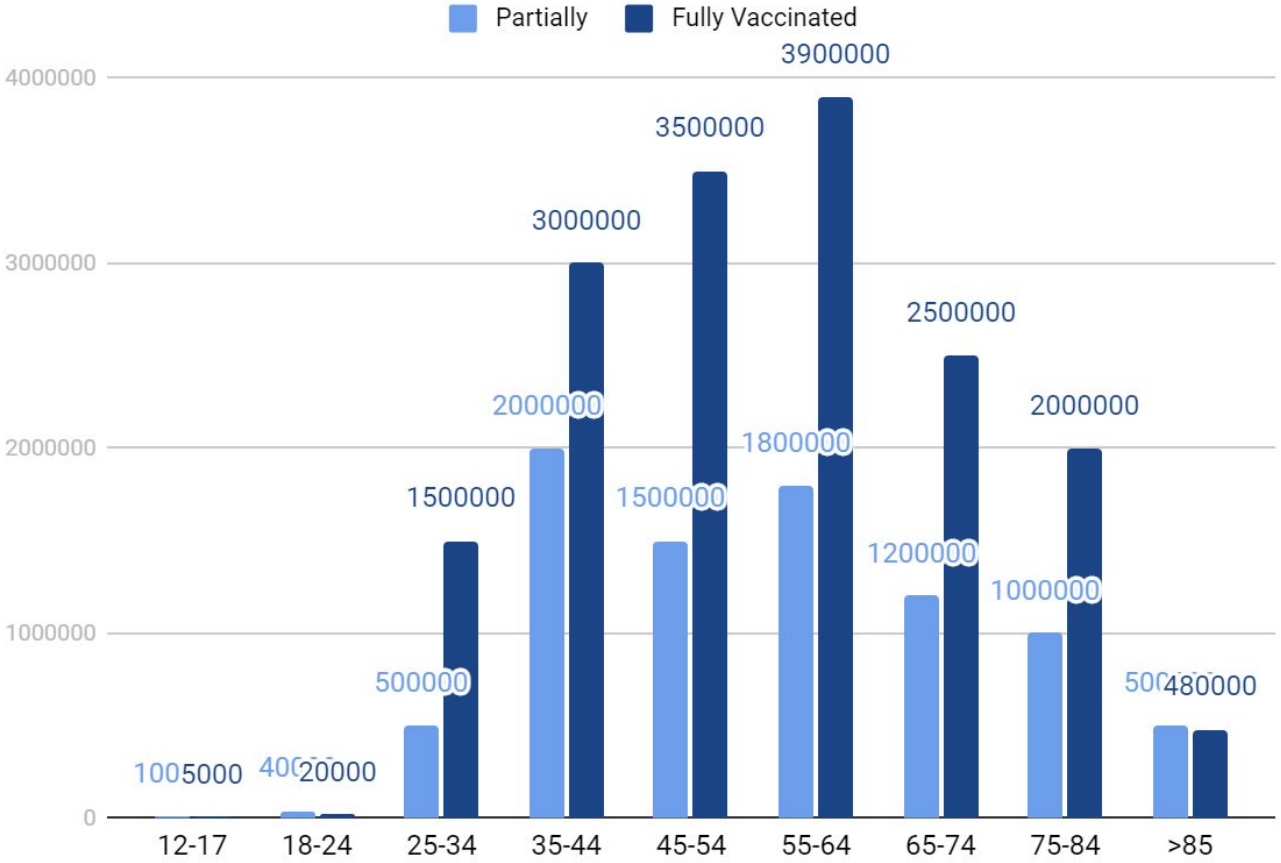
- All the graphs should have appropriate titles, x and y labels, legends, correct x and y tick labels, etc. I am not defining all the aesthetics but all graphs should be presentable and legible.
- See some examples of the graphs on the following pages

Vaccinations by Day

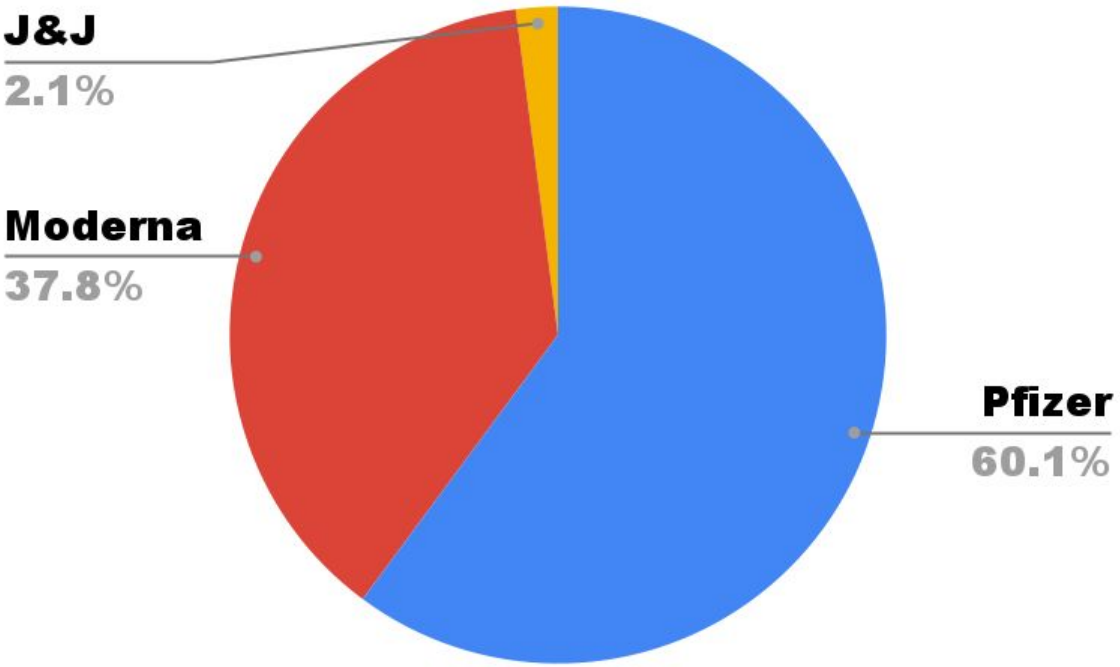




Vaccinations by Age



Doses by Vtype



Task 4: Generate data for testing purposes (10 pts)

You can start with the data published here. You can select North Carolina and start with the numbers there.

Write a python script "**gen_covis_data.py** to generate random data based on the numbers above.

- The script will take arguments in the following format "python gen_covis_data.py --no <no of records> --fname <filename>"
- <no of records> is number of random people to be generated in file with <filename>
- If no arguments are given, generate by default 100,000 records and save them in a new file "covidVaccineData.csv"
- While generating the covis data, keep the following checks in mind.
  - All required attributes need to be randomly generated for each record.
  - All the generated patient attributes should pass the checks stated in the previous slide.
  - The file should have the header with attribute keys listed on slide 3.
  - Try and keep the ratios of gender cases, vaccination rate, partially and fully vaccinated people same as the real NC data

Hints about generating data using a script

1. Start with NC numbers
2. Add member function in a loop
3. Import data using a file
4. Bulk member operation using change member option
5. Numpy.random,choice
    a. draw = np.randomchoice(list_of_candidates,
    number_of_items_to_pick,
    b. p=probability_distribution)