HOMEWORK 7 – there is no template file, you will create a notebook or a python script for this

BIOINF 575

PROBLEM 1 (20 points): - add the solution to this problem in a markdown cell or as a comment

Identify **one** Python package that is used in your domain (e.g. genomics, transcriptomics, proteomics, epidemiology, etc). There are lot of places to look, so here are a few to get you started:

- PyPI, conda-forge, Anaconda, Bioconda

For the package that *you identify*, you must report the following items:

A. **(1 points)** What the package is?
B. **(2 points)** Why did you select this package?
C. **(2 points)** What version(s) of Python does it support?
D. **(2 points)** What operating system(s) does it support?
E. **(2 points)** What are the package's dependencies? That is, what other packages does your identified package require to work?
F. Identify one class definition and answer the following questions:

1. **(3 points)** What is the name of the class and where is it in the code base?
    - Report the file it is found in and the line number it starts at
2. **(2 points)** In your own words (not code), what does the object do or what is its purpose?
3. **(2 points)** What are the ways an object of that type can be instantiated? That is, how do you use it?
4. **(2 points)** Does it inherit from other classes? If so, from what classes?
5. **(2 points)** What functionality would you add to this object that you would need specifically for your own research?
    - You **cannot** write "nothing" or "none", you *must* write something meaningful here

PROBLEM 2 (50 points):

A. **(15 points)** Create a database called worm_genome.sqlite (if you connect to a file with this name the connection will automatically create the file with an empty database if the file does not exist)
- **(5 points)** Create a features table - see details below
- **(5 points)** Create an attributes table - see details below
- **(5 points)** Create indexes for the two tables on the following columns:
    - features: type, start, end
    - attributes: feature_id, name
    - Note: These are column that are typically used in queries in the WHERE clause

B. **(20 points)** Populate the database (the insert commands can be executed while parsing):
- **(10 points)** Parse GFF3 file: worm_genome.gff3:
    - This file be found in the class GitHub repository: https://github.com/dcmb-courses/bioinf575
    - You can also use the following link to read the file using python code: https://raw.githubusercontent.com/dcmb-courses/bioinf575/main/worm_genome.gff3
    - Study session 6 uses a short version of this file
- **(10 points)** Insert the data from the gff3 file into the features and attributes tables

C. **(15 points)** Query the database:
- After you have created the database and loaded the data, **report the result of the following 5 queries**:
    1. **(2 points)** The number of rows in the *features* table
    2. **(2 points)** The number of rows in the *attributes* table

3. **(3 points)** Display the type and the number of features for each type from the *features* table
4. **(4 points)** Display the seq_id and type from the *features* table and the attr_name and value from the *attributes* table for all features with the type chromosome
5. **(4 points)** Display the type from the *features* table and the number of attributes (using also the *attributes* table) for each feature type that has more than 50 attributes

Note: All these goals are achievable through SQL using Python

Given the definition of the GFF3 format described below, **create a database with two tables that have the structure described below and load the data from the GFF3 file into the tables**:

- There are lines that *start with* an octothorpe (#) scattered throughout the file. These are comment lines. They can be skipped.
- Rows in the gff file contain 9 tab-separated fields.
- The first 8 fields from each line in file get put into a row in the features table.
- The 9th field data will go into multiple rows in the attributes table. It consists of attributes which are name-value pairs.
    - Attributes are of name/value pairs that are separated by semicolon (;)
    - Name and values in the pairs separated by equal (=)
    - There can be multiple values *per* name. The values are comma-separated (,) in that case.

Table: *features*, with the following columns:

1. feature_id (primary key, auto increment: SEE BELOW) – integer **- does not come from the file**
2. seq_id - string
3. source - string
4. type - string
5. start - string (can make it integer if you want)
6. end - string (can make it integer if you want)
7. score - string (can make it a float if you want)
8. strand - string
9. phase - string

Table: *attributes*, with the following columns:

The data for this table should be taken from the 9th column of the GFF3 file).

1. attr_id (primary key, auto increment) – integer **- does not come from the file**
2. feature_id (foreign key) – integer **- does not come from the file**
3. attr_name - string
4. value - string

The featureid and attributeid do not come from the file you can keep track of each of these value though a variable you increment, or you can rely on the autoincrement functionality in the database to add and increment these columns automatically if you don't provide a value for it at insert.

When you insert a feature in the features table, you need the feature_id to add its attributes to the attributes table. If you do not provide a value at insert, the feature_id for the inserted feature can be retrieved using cursor.lastrowid, (e,g.: **id = cursor.lastrowid**). You then use that result as the feature_id in the insert statement for attributes table.

More detailed description of the GFF3 format:

The General Feature Format (GFF) file is a simple tab-delimited text file, wherein each line describes some genomic feature. A variation of the GFF format is the GFF3 format. It further extends the GFF format such that it:

1. Adds a mechanism for representing more than one level of hierarchical grouping of features and subfeatures
2. Separates the ideas of group membership and feature name/id
3. Constrains the feature type field to be taken from a controlled vocabulary
4. Allows a single feature, such as an exon, to belong to more than on group at a time
5. Provides an explicit convention for pairwise alignments
6. Provides an explicit convention for features that occupy disjunct regions

GFF3 files are ***nine-column, tab-delimited, plain text files***. The first line of a GFF3 file is a comment that identifies the version. All but the final field in each feature line must contain a value; "empty" columns should be denoted with a '.'. The descriptions of each of the 9 columns are as such:

1. **seqid:** name of the chromosome or scaffold; chromosome names can be given with or without the 'chr' prefix.
    - ***Important note****:* the seq ID must be one used within Ensembl, i.e. a standard chromosome name or an Ensembl identifier such as a scaffold ID, without any additional content such as species or assembly.
2. **source:** name of the program that generated this feature, or the data source (database or project name)
3. **type:** type of feature. Must be a term or accession from the SOFA sequence ontology
4. **start:** Start position of the feature, with sequence numbering starting at 1.
5. **end:** End position of the feature, with sequence numbering starting at 1.
6. **score:** A floating point value.
7. **strand:** Defined as + (forward) or - (reverse).
8. **phase:** One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on.
9. **attributes:** A semicolon-separated list of tag-value pairs, providing additional information about each feature. Some of these tags are predefined, e.g. ID, Name, Alias, Parent - see the GFF documentation for more details.