General Description
We want to build software that manages supermarket chains, called BGU Mart. The software should supp ort managing a large number of employees and the buying/selling of products. The software should also manage the inventory and thus contact various suppliers, who supply products. Sells and deliveries of pro ducts should be also registered and logged for tax purposes.

We need your help to build the software by implementing a tool using Python and SQLite.

Method and Technical Description
You will build a sqlite3 database that will hold the employee, supplier, product, branch, and activity tables.

The database filename will be bgumart.db.
You will have to implement three Python modules: initiate.py, action.py, and printdb.py.

The Database Structure
The database bgumart.db has five tables.

Employees: This table holds information about the employees.
Suppliers: This table holds information about the suppliers.
Products: This table holds information about the products.
Branches: This table holds information about the branches.
Activities: This table holds information about all activities of the chain including sales and deliveries.
Overall, the tables should be defined as follows:

```
CREATE TABLE employees (
   id              INT         PRIMARY KEY,
   name            TEXT        NOT NULL,
   salary          REAL        NOT NULL,
   branche    INT REFERENCES branches(id)
);

CREATE TABLE suppliers (
   id                  INTEGER    PRIMARY KEY,
   name                TEXT       NOT NULL,
   contact_information  TEXT
);

CREATE TABLE products (
   id         INTEGER PRIMARY KEY,
   description TEXT    NOT NULL,
   price       REAL NOT NULL,
   quantity    INTEGER NOT NULL
);

CREATE TABLE branches (
   id                 INTEGER     PRIMARY KEY,
   location           TEXT        NOT NULL,
   number_of_employees INTEGER
);

CREATE TABLE activities (
   product_id      INTEGER REFERENCES products(id),
   quantity        INTEGER NOT NULL,
   activator_id    INTEGER NOT NULL,
        date            TEXT    NOT NULL
    );
```

initiate.py
This module builds the database and inserts the initial data from the configuration file. When run, it will be given a configuration file as an argument. For example:

```
python3 initiate.py config.txt
```
If the database file already exists remove it.

Initiate.py should create a “fresh” database with the tables as specified, parse the configuration file, and store the data given in the configuration file in the database appropriately.

You may assume that the configuration file exists and that the syntax and the data are valid.

action.py
This module manages the supermarket activities, i.e. sales and deliveries (buys). When run, it will be given an actions file as an argument. It will perform each action in the order it appears in the file and then exits.

You may assume that the configuration file exists and that the syntax and the data are valid but you need to check that the quantity of the sold product is enough. If for any reason an action may not be fulfilled do NOTHING!! (Do not print an error message, Do not "sale" part of the quantity, etc...)

printdb.py
This module prints the database and will be checked automatically, therefore, you should follow the instruction very carefully.

For each table print in a new line the name of the table followed by its records/tuples (each in a row). The tuples should be printed in ascending order of the primary key except for the activities table which it should be ordered by the date. The printing order of the tables is also in ascending order: Activities, Branches, Employees, Products, and Suppliers.
Print a detailed employees report with the following information:
Name, Salary, Working location, total sales income. The output should be printed in ascending order of the employee name. If two or more employees share the same name their inner print order is not important. You should print each employee in a row using a single space between the fields.
Print a detailed activity report with the following information:
date of activity, item description, quantity, name of seller, and the name of the supplier. If the activity is a sale the name of the supplier should be ‘None’, If the activity is supplying the name of the seller should be ‘None’. The tuples should be printed from the oldest to the newest. If there is no activity do not print. You must use SQL SELECT (with ‘join’, ‘order by’ etc…) to complete this.
Note:

Use Python’s print() function to print your data and tuples, do not format your output just use the default print behavior. However, If you need to print a list of Python’s objects use the __str__ function to override the default print of an object since the default print prints only the reference. __str__ the equivalent of Java’s toString.
You may add modules as you need but you may not remove or change the file names of these three modules.
You are not obligated to use the template provided, but it should make your life easier
All three modules should run as a standalone program, i.e. can be run from the command line as the main module.
We will run each of your module from the command line, for example:
```
> initiate configfile.txt
> printdb
> action actionfile.txt
> printdb
```

Configuration and action Files

Configuration file

Each line in the configuration file represents either an Employee(E), Supplier(S), Product(P) or Branch(C).

For example:

B,3,Chicago,40 represents a branch with id 3 located in Chicago with 40 employees.

E,106,Sue Davis,75000,3 represents an employee with id 106, named Sue Davis with a 75000 yearly salary that is working in branch 3.

P,5,Mango,2,7 represents a product with id 5, its description is Mango, the price is 2 shekel and the quantity is 7.

S,6,Jkl Enterprises,(678) 901-2345 represents a supplier with id is 6, named Jkl Enterprises and its contact information is (678) 901-2345.

Note:

E, S, P, and B at the beginning of each input row define record types and should not be inserted into the database. An example of a configuration file and its output is supplied on the assignment page.
Employee IDs and Supplier IDs are unique, meaning an employee ID can not repeat itself as a supplier ID and vice versa.
(Employee-IDs   Supplier-IDs = Ø).

Action file

Each line in the action file represents an activity. An activity can be either a sale or a supply arrival. When quantity < 0 it is a sale activity, when quantity > 0 it is a supply arrival, and quantity=0 is illegal. For example:

3, 500, 56, 20230110 represents that supplier 56 supplied 500 units of product 3 on 10/Jan/2023.

100, -500, 1234, 20230110 represents that employee 1234 sold 500 units of product 100 on 10/Jan/2023.

If the current product quantity is less than the quantity in the sale activity the action should be ignored. Do not print any message.

Development Environment

You should use Python 3.9 (or above) and sqlite3. You can use a lower version of python, but it is not recommended.

You can use any text editor to program the assignment, but make sure your code works when running it from the terminal as specified.

Example

Here is an example. Note that you cannot assume the order of the information in the configuration file. You can only assume the validity of each line's syntax, and the validity of the configuration file (no double data, branch exist, no illegal products, etc.)

Suppose you are given the following configuration file config.txt and run python3 initiate.py config.txt, then the database should look like this .
You can open the provided file using various sqlite3 viewer tools such as this VSCode extension, some online viewer like this one, or even the sqlite3 command-shell program.

python3 printdb.py will print the following.

```
Activities
Branches
(1, ’New York’, 50)
(2, ’Los Angeles’, 60)
(3, ’Chicago’, 40)
(4, ’Houston’, 70)
(5, ’Philadelphia’, 45)
(6, ’Phoenix’, 55)
(7, ’San Antonio’, 35)
(8, ’San Diego’, 65)
(9, ’Dallas’, 75)
(10, ’San Jose’, 80)
Employees
(101, ’John Smith’, 50000.0, 1)
(102, ’Jane Doe’, 60000.0, 1)
(103, ’Bob Johnson’, 45000.0, 2)
(104, ’Alice Williams’, 55000.0, 2)
(105, ’Mike Brown’, 65000.0, 3)
(106, ’Sue Davis’, 75000.0, 3)
(107, ’Tom Davis’, 85000.0, 3)
(108, ’Jerry Smith’, 95000.0, 1)
(109, ’Alice Johnson’, 105000.0, 2)
(110, ’Bob Williams’, 115000.0, 2)
(111, ’Mike Davis’, 125000.0, 3)
(112, ’Sue Smith’, 135000.0, 1)
(113, ’Tom Johnson’, 145000.0, 2)
(114, ’Jerry Williams’, 155000.0, 2)
(115, ’Alice Brown’, 165000.0, 3)
(116, ’Bob Davis’, 175000.0, 3)
(117, ’Mike Smith’, 185000.0, 1)
(118, ’Sue Johnson’, 195000.0, 2)
(119, ’Tom Williams’, 205000.0, 2)
(120, ’Jerry Brown’, 215000.0, 3)
Products
(1, ’Apple’, 0.5, 10)
(2, ’Banana’, 0.25, 20)
(3, ’Orange’, 0.75, 15)
(4, ’Grapes’, 1.5, 5)
(5, ’Mango’, 2.0, 7)
(6, ’Peach’, 1.25, 12)
(7, ’Pineapple’, 3.0, 8)
(8, ’Strawberry’, 1.75, 9)
(9, ’Blueberry’, 2.5, 6)
(10, ’Raspberry’, 1.5, 11)
Suppliers
(1, ’Acme Inc.’, ’(123) 456-7890’)
(2, ’XYZ Corp.’, ’(234) 567-8901’)
(3, ’ABC Enterprises’, ’(345) 678-9012’)
(4, ’Def Co.’, ’(456) 789-0123’)
(5, ’Ghi Inc.’, ’(567) 890-1234’)
(6, ’Jkl Enterprises’, ’(678) 901-2345’)
(7, ’Mno Co.’, ’(789) 012-3456’)
(8, ’Pqr Inc.’, ’(890) 123-4567’)
```

```
(9, ’Stu Enterprises’, ’(901) 234-5678’)
(10, ’Vwx Co.’, ’(012) 345-6789’)

Employees report
Alice Brown 165000.0 Chicago 0
Alice Johnson 105000.0 Los Angeles 0
Alice Williams 55000.0 Los Angeles 0
Bob Davis 175000.0 Chicago 0
Bob Johnson 45000.0 Los Angeles 0
Bob Williams 115000.0 Los Angeles 0
Jane Doe 60000.0 New York 0
Jerry Brown 215000.0 Chicago 0
Jerry Smith 95000.0 New York 0
Jerry Williams 155000.0 Los Angeles 0
John Smith 50000.0 New York 0
Mike Brown 65000.0 Chicago 0
Mike Davis 125000.0 Chicago 0
Mike Smith 185000.0 New York 0
Sue Davis 75000.0 Chicago 0
Sue Johnson 195000.0 Los Angeles 0
Sue Smith 135000.0 New York 0
Tom Davis 85000.0 Chicago 0
Tom Johnson 145000.0 Los Angeles 0
Tom Williams 205000.0 Los Angeles 0

Activities report
```

Suppose now you are given the following action file, action.txt , and you run python3 action.py action1. Th
en, the database will look like this.
Now running python3 printdb.py will print the following.

```
Activities
(3, 100, 1, ’20230101’)
(6, 50, 2, ’20230101’)
(3, -20, 103, ’20230201’)
(3, -10, 104, ’20230301’)
(6, -50, 103, ’20230301’)
Branches
(1, ’New York’, 50)
(2, ’Los Angeles’, 60)
(3, ’Chicago’, 40)
(4, ’Houston’, 70)
(5, ’Philadelphia’, 45)
(6, ’Phoenix’, 55)
(7, ’San Antonio’, 35)
(8, ’San Diego’, 65)
(9, ’Dallas’, 75)
(10, ’San Jose’, 80)
Employees
(101, ’John Smith’, 50000.0, 1)
(102, ’Jane Doe’, 60000.0, 1)
(103, ’Bob Johnson’, 45000.0, 2)
(104, ’Alice Williams’, 55000.0, 2)
(105, ’Mike Brown’, 65000.0, 3)
(106, ’Sue Davis’, 75000.0, 3)
(107, ’Tom Davis’, 85000.0, 3)
```

```
(108, ’Jerry Smith’, 95000.0, 1)
(109, ’Alice Johnson’, 105000.0, 2)
(110, ’Bob Williams’, 115000.0, 2)
(111, ’Mike Davis’, 125000.0, 3)
(112, ’Sue Smith’, 135000.0, 1)
(113, ’Tom Johnson’, 145000.0, 2)
(114, ’Jerry Williams’, 155000.0, 2)
(115, ’Alice Brown’, 165000.0, 3)
(116, ’Bob Davis’, 175000.0, 3)
(117, ’Mike Smith’, 185000.0, 1)
(118, ’Sue Johnson’, 195000.0, 2)
(119, ’Tom Williams’, 205000.0, 2)
(120, ’Jerry Brown’, 215000.0, 3)
Products
(1, ’Apple’, 0.5, 10)
(2, ’Banana’, 0.25, 20)
(3, ’Orange’, 0.75, 85)
(4, ’Grapes’, 1.5, 5)
(5, ’Mango’, 2.0, 7)
(6, ’Peach’, 1.25, 12)
(7, ’Pineapple’, 3.0, 8)
(8, ’Strawberry’, 1.75, 9)
(9, ’Blueberry’, 2.5, 6)
(10, ’Raspberry’, 1.5, 11)
Suppliers
(1, ’Acme Inc.’, ’(123) 456-7890’)
(2, ’XYZ Corp.’, ’(234) 567-8901’)
(3, ’ABC Enterprises’, ’(345) 678-9012’)
(4, ’Def Co.’, ’(456) 789-0123’)
(5, ’Ghi Inc.’, ’(567) 890-1234’)
(6, ’Jkl Enterprises’, ’(678) 901-2345’)
(7, ’Mno Co.’, ’(789) 012-3456’)
(8, ’Pqr Inc.’, ’(890) 123-4567’)
(9, ’Stu Enterprises’, ’(901) 234-5678’)
(10, ’Vwx Co.’, ’(012) 345-6789’)

Employees report
Alice Brown 165000.0 Chicago 0
Alice Johnson 105000.0 Los Angeles 0
Alice Williams 55000.0 Los Angeles 7.5
Bob Davis 175000.0 Chicago 0
Bob Johnson 45000.0 Los Angeles 77.5
Bob Williams 115000.0 Los Angeles 0
Jane Doe 60000.0 New York 0
Jerry Brown 215000.0 Chicago 0
Jerry Smith 95000.0 New York 0
Jerry Williams 155000.0 Los Angeles 0
John Smith 50000.0 New York 0
Mike Brown 65000.0 Chicago 0
Mike Davis 125000.0 Chicago 0
Mike Smith 185000.0 New York 0
Sue Davis 75000.0 Chicago 0
Sue Johnson 195000.0 Los Angeles 0
Sue Smith 135000.0 New York 0
Tom Davis 85000.0 Chicago 0
Tom Johnson 145000.0 Los Angeles 0
Tom Williams 205000.0 Los Angeles 0

Activities report
('20230101', 'Orange', 100, None, 'Acme Inc.')
('20230101', 'Peach', 50, None, 'XYZ Corp.')
('20230201', 'Orange', -20, 'Bob Johnson', None)
('20230301', 'Orange', -10, 'Alice Williams', None)
('20230301', 'Peach', -50, 'Bob Johnson', None)
```

Important notes

We will test your modules together, and independently. Independently means that we will, for example, use our own database but use your modules, or put one module of our own and use your other module. Therefore, make sure your database has the structure we specified exactly, and that the behavior of each module is precisely as specified. Also, make sure that the database filename is bgumart.db. Failing to follow these guidelines will cause tests to fail, and your grade will suffer accordingly.

To save you time, you may assume the validity of input. For example, an employee will not be assigned to a non-existent branch. However, you may not assume that the quantity of a sell activity is enough.

We emphasize again, please make sure everything is as described, otherwise you will lose critical points from your grade! We care about your success as much as you do. Make sure the database filename is bgumart.db in lower case. Make sure your table and column names are also exactly as described. If you are not sure about something, then feel free to ask in the forum!

Submission

The submission is done in pairs only.

You must submit one file with all your code. The file should be named id1_id2.zip. This file should include at least the following files:

initiate.py action.py printdb.py

and any extra files that your code needs in order to work (repository, DTOs, DAOs, etc…).

We provide a template below for your convenience, but you don't have to use the template.