Доброго времени суток!

Меня зовут Александра Юсупова, я претендую на позицию стажер-разработчик на языке Python.

Ссылка на репозиторий с доп.файлами к заданиям:

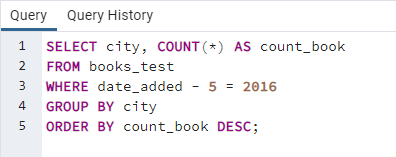
<https://github.com/Kobatoha/Test-tasks/tree/main/KORUS%20Consulting>

**Задание №1.**

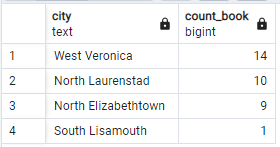
1. **Найти города, в которых в 2016 году было издано больше всего книг**

Поскольку в семпле базы данных все значения city уникальны, то для лучшей демонстрации работы запроса на языке SQL решила сделать загрузку в базу данных PostgreSQL обновленных данных с дублированием значений в колонке city.

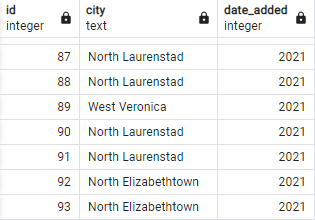
Запрос SQL:



Результат запроса:



Пример загруженных данных:



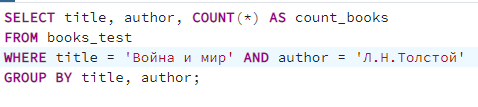
Примечание: обновленный семпл базы данных находится в репозитории в директории task\_1 – **books\_test.csv.**  В той же директории находится исполняющий **task\_1.py** файл, который заполняется базу данных значениями из .csv файла.

1. **Вывести количество экземпляров книг «Война и мир» Л.Н.Толстого, которые находятся в библиотеке.**

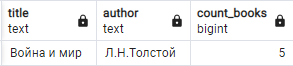
В предоставленном семпле нет такой позиции, как **«Война и мир» Л.Н.Толстого,** поэтому я использовала дополнительную выгрузку данных в базу данных. Файл находится в репозитории task\_1 – **books\_test2.csv**

В это вопросе нет уточнения, сколько книг «Война и мир» Л.Н.Толстого нужно найти в библиотеке в целом или конкретно сейчас. Если в целом и общем, то запрос выглядит так:

Запрос SQL:

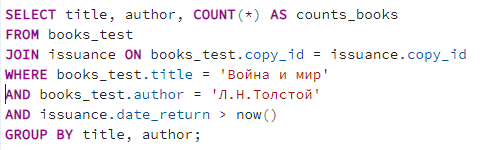


Результат запроса:

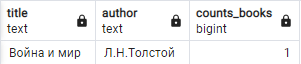


Чтобы узнать количество экземпляров книг взятых читателями и еще не возвращенных, то ответом будет результат такого запроса:

Запрос SQL:



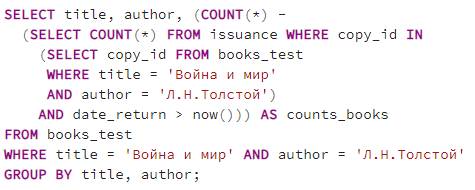
Результат запроса:



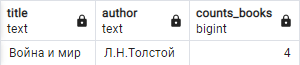
Тут мы видим, что из всех изданий «Война и мир» не вернули в библиотеку один экземпляр книги.

Если мы хотим точно знать количество книг, находящихся в библиотеке на данный момент, то мы используем этот запрос:

Запрос SQL:



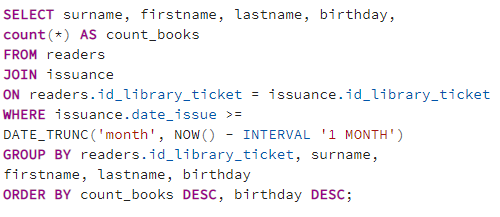
Результат запроса:



Здесь мы используем подзапрос, чтобы из общего количества книг «Война и мир», которые числятся в библиотечной системе, вычесть те экземпляры, срок возврата которых еще не наступил. Тем самым мы получаем количество книг, которые физически сейчас лежат на библиотечных полках.

1. **Найти читателей, которые за последний месяц брали больше всего книг в библиотеке c сортировкой читателей по возрасту (от молодых к старшим)**

Запрос SQL (на диалекте PostgreSQL):



В этом запросе мы объединяем таблицы **readers** и **issuance** по полю **id\_library\_ticket**. В строке **WHERE** задаются условия выборки: выбираются только записи, где дата выдачи книги больше или равна первому дню текущего месяца минус один месяц. Затем мы группируем данные и сортируем их по убыванию количества книг (**count\_books**) и возрастанию даты рождения (**birthday**) каждого читателя.

Самый читающий молодняк в библиотеке – 5летний Аллен Кристофер Дженниферович 

Результат запроса:



**Задание №2.**

Исполняющий файл – **task\_2.py**

Какая задача была поставлена? - Реализовать алгоритм для подсчета длины максимальной последовательности подряд идущих положительных чисел в массиве, состоящем из произвольного количества целых чисел (как положительных, так и отрицательных).

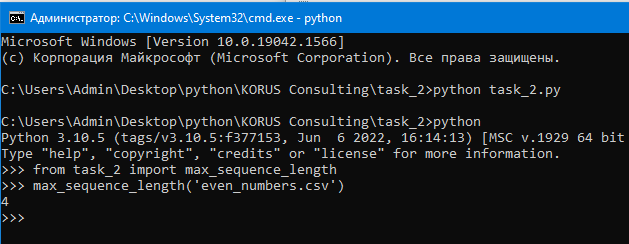
Данную задачу я реализовала в 3 этапа:

1. С помощью конструкции **with open** заполнила список данными из файла **numbers.csv**
2. Циклом **for** проверяла, является ли значение положительным. Если да, то оно добавляется в текущую последовательность. Если значение отрицательное, то текущая последовательность сравнивается с максимальной. Максимальная последовательность либо обновляется значениями текущей, либо остается неизменной, а текущая соответственно обнуляется. В итоге мы получаем самый длинный список положительных подряд идущих чисел.
3. Возвращаем **len()** максимальной последовательноти с помощью **return**

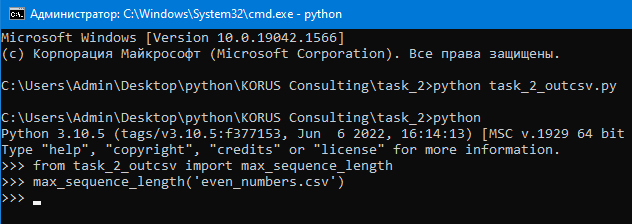
Дополнительная информация по задаче:

1. На вход функция **max\_sequence\_length(‘file.csv')** принимает любой файл .csv в формате «int\n». Например, **max\_sequence\_length(‘test.csv')** или **max\_sequence\_length(‘even\_numbers.csv')**.
2. Файл .csv должен находиться в той же директории, что и исполняющий файл .py
3. Если в файле содержатся не-int значения, то они игнорируются на этапе чтения из файла и заполнения списка.
4. В **репозитории** в директории 'task\_2' находится файл - **task\_2\_outcsv.py**. Он имеет тот же функционал, что и **task\_2.py**, но результат выводит не в консоль через **return**, а создает в той же директории .csv файл, который называется так же, как и файл, из которого производилось чтение на этапе 1, с припиской '\_out'. Например, на вход функции задаем файл **'even\_numbers.csv'**, а на выходе получаем файл в той же директории **'even\_numbers\_out.csv'**

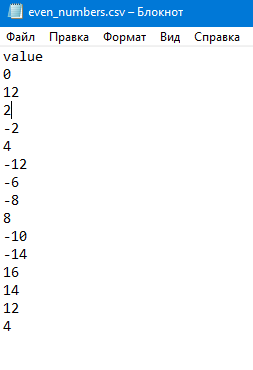
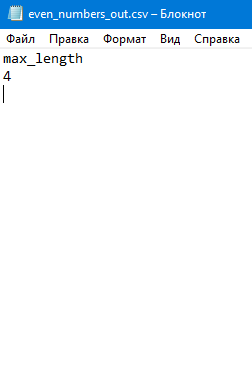
Пример вызова функции с выводом результата в консоль:



С выводом результата в «**even\_numbers\_out.csv**»



Входные данные из файла .csv Выходные данные из файла .csv

**Задание №3.**

Исполняющий файл – **task\_3.py**

Поставленная задача - Реализовать алгоритм для подсчета прибыли подразделений некоторой компании по месяцам.

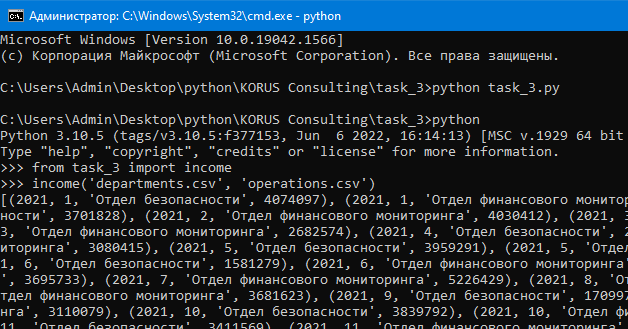
Данную задачу я реализовала в 3 этапа:

1. Конструкцией **with open** считала данные из файлов '**departments.csv**' and '**operations.csv**' и занесла их в списки кортежей **departments** and **operations** соответственно.
2. Для каждой операции циклом **for** в списке **operations** извлекла год, месяц, id departments и прибыль. Проверила данные на корректность через оператор сравнения **if** , что департамент работал в указанный период, а так же сформировала опять же циклом **for** ключи для созданного словаря **profits**. И наконец добавила прибыль к значениям словаря по соответствующему ключу.
3. Циклом **for** преобразовала словарь в список кортежей, отсортировала его по возрастанию даты. И вернула **return** результат.

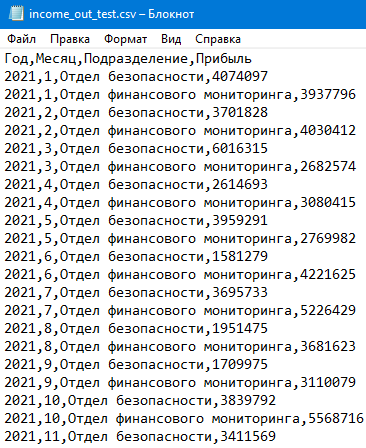
Дополнительная информация по задаче:

1. На вход функция **income(‘file.csv', ‘file.csv’)** принимает два файла .csv. Пример: **income(‘departments.csv', ‘operations.csv’)**
2. Файлы .csv должны находиться в той же директории, что и исполняющий файл .py
3. В **репозитории** в директории 'task\_3' находится файл - **task\_3\_outcsv.py**. Он имеет тот же функционал, что и **task\_3.py**, но результат выводит не в консоль через **return**, а создает в той же директории .csv файл, который называется **'income\_out\_test.csv'**.

Запуск **task\_3.py** через консоль и вывод результата:



Запуск **task\_3\_outcsv.py** и вывод результата в файл **'income\_out\_test.csv’**:



На этом у меня все. Спасибо за внимание ☺

С уважением, Александра.