

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №8
дисциплины «Анализ данных»

Выполнил:

Гайчук Дарья Дмитриевна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А.-доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Цель работы: исследовать базовые возможности системы управления базами данных SQLite3

Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.

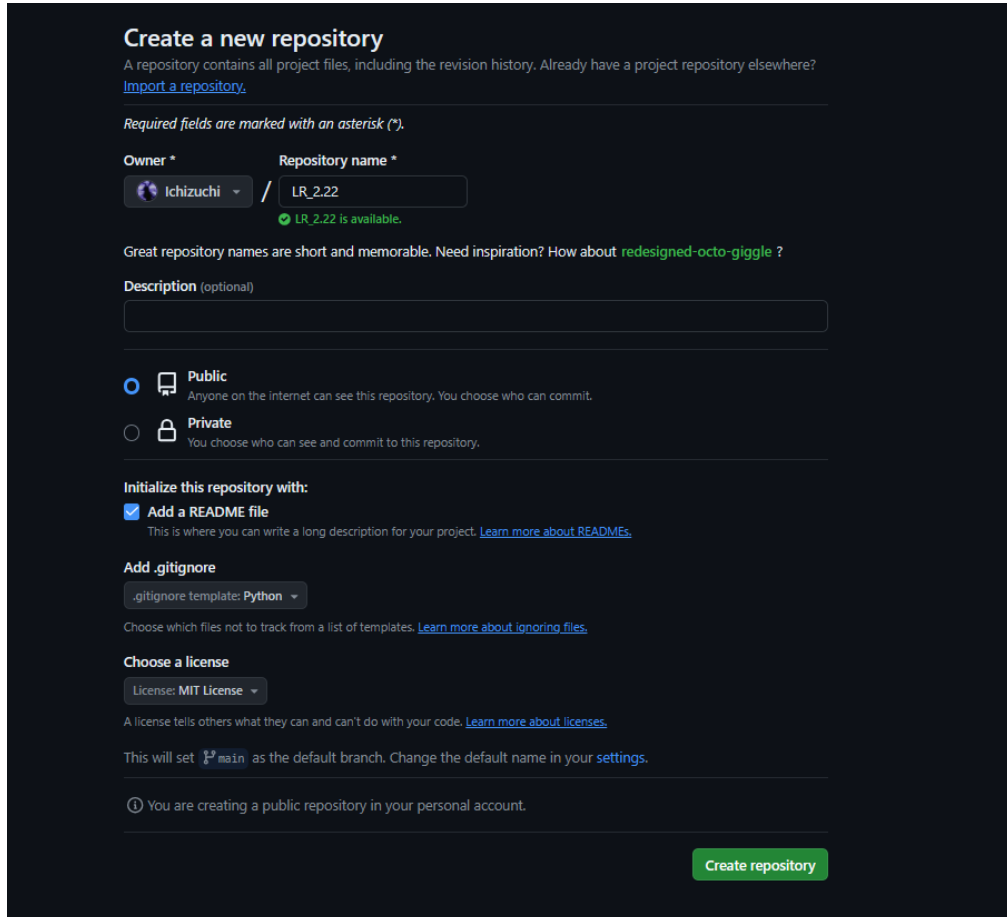


Рисунок 1. Создан новый репозиторий

2. Клонировала репозиторий на свой компьютер.

```
@Ichizuchi →/workspaces/LR_2.22 (main) $ git clone https://github.com/Ichizuchi/LR_2.22.git
Cloning into 'LR_2.22'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
@Ichizuchi →/workspaces/LR_2.22 (main) $ git branch develop
@Ichizuchi →/workspaces/LR_2.22 (main) $ git checkout develop
Switched to branch 'develop'
@Ichizuchi →/workspaces/LR_2.22 (develop) $
```

Рисунок 2. Клонирование и модель ветвления git-flow

3. Создал виртуальное окружение Anaconda с именем репозитория.

```
@Ichizuchi → /workspaces/LR_2.22 (develop) $ conda create -n myenv python=3.10
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda/envs/myenv

added / updated specs:
  - python=3.10

The following packages will be downloaded:
```

package	build	
_____	_____	
_libgcc_mutex-0.1	main	3 KB
_openmp_mutex-5.1	1_gnu	21 KB
bzip2-1.0.8	h5eee18b_5	262 KB
ca-certificates-2024.3.11	h06a4308_0	127 KB
ld_impl_linux-64-2.38	h1181459_1	654 KB
libffi-3.4.4	h6a678d5_0	142 KB
libgcc-ng-11.2.0	h1234567_1	5.3 MB

Рисунок 3. Создание виртуального окружения

Выполнение индивидуального задания

Вариант 3

Для индивидуального задания лабораторной работы 2.21 добавьте тесты с использованием модуля unittest, проверяющие операции по работе с базой данных.

```
114 # тестирование поиска магазина по идентификатору
115 find_store(self.cursor, 1)
116
117 # Проверка результата
118 expected_output = "Найден магазин:\nID: 1\nНазвание: Store A\nАдрес: Address A\n"
119 self.assertEqual(captured_output.getvalue(), expected_output)
120
121 # Очистка вывода
122 captured_output.truncate(0)
123 captured_output.seek(0)
124
125 # Тестирование с (variable) cursor: Cursor
126 find_store(self.cursor, 3)
127
128 # Проверка результата
129 expected_output = "Магазин с указанным ID не найден.\n"
130 self.assertNotIn(captured_output.getvalue(), expected_output)
```

PROBLEMY Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ КОММЕНТАРИИ

FAIL: test_display_table (_main_.TestDatabaseOperations)

Traceback (most recent call last):
 File "/workspaces/LR_2.22/tests.py", line 104, in test_display_table
 self.assertEqual(captured_output.getvalue(), expected_output)
AssertionError: '[77 chars] | Название | Адрес |' != '[256 chars]'\n' '[77 chars] | Название | Адрес |' [265 chars]'\n'

ID	Название	Адрес
1	Store A	Address A
2	Store B	Address B

Ran 2 tests in 0.002s

Рисунок 4. Успешное выполнение тестов

Ответы на контрольные вопросы

1. Автономный тест – это автоматизированная часть кода, которая вызывает тестируемую единицу работы и затем проверяет некоторые предположения о единственном конечном результате этой единицы.

2. В мире Python существуют три framework'а, которые получили наибольшее распространение: unittest, nose ,pytest .

3. Test fixture Test fixture – обеспечивает подготовку окружения для выполнения тестов, а также организацию мероприятий по их корректному завершению (например очистка ресурсов). Подготовка окружения может включать в себя создание баз данных, запуск необходим серверов и т.п.

Test case Test case – это элементарная единица тестирования, в рамках которой проверяется работа компонента тестируемой программы (метод, класс, поведение и т. п.). Для реализации этой сущности используется класс TestCase.

Test suite Test suite – это коллекция тестов, которая может в себя включать как отдельные test case'ы так и целые коллекции (т.е. можно создавать коллекции коллекций). Коллекции используются с целью объединения тестов для совместного запуска.

Test runner Test runner – это компонент, которые оркестрирует (координирует взаимодействие) запуск тестов и предоставляет пользователю результат их выполнения.

Test runner может иметь графический интерфейс, текстовый интерфейс или возвращать какое-то заранее заданное значение, которое будет описывать результат прохождения тестов.

4. Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор UPDATE.

5. Оператор SELECT используется для выбора данных из определенной таблицы. Если вы хотите выбрать все столбцы данных из таблицы, вы можете использовать звездочку (*).

6. SQLite3 rowcount используется для возврата количества строк, которые были затронуты или выбраны последним выполненным SQLзапросом.

7. Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы sqlite_master, а затем использовать fetchall() для получения результатов из инструкции SELECT

8. При создании таблицы мы должны убедиться, что она еще не существует. Аналогично, при удалении/удалении таблицы она должна существовать. Чтобы проверить, не существует ли таблица уже, мы используем IF NOT EXISTS с оператором CREATE TABLE следующим образом.

9. Метод executemany можно использовать для вставки нескольких строк одновременно.

10. В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль datetime . Следующие форматы являются наиболее часто используемыми форматами для datetime:

Вывод: в результате выполнения работы были приобретены навыки написания автоматизированных тестов на языке программирования Python версии 3.x.