УРОК 1 - Базы данных и SQL

ТЕМЫ:

A. Базы данных и СУБД
B. SQL

ДЕТАЛИ:

- Создание БД
- Инструмент Sql Lite Studio
- Создание таблиц и типы данных в БД
- Добавление, изменение и удаление записей
- Выборка данных из таблицы (WHERE -> LIKE, BETWEEN, IN, IS NULL)
- Логические операторы OR, AND, NOT
- Выражение DISTINCT и функция - strftime('%d %m %Y', '2015-12-01')
- Условная конструкция CASE
- Сортировка результатов выборки

ДЗ:
Общее описание: SQL запросы описанные ниже необходимо написать и запустить в базе данных hw_2.db прикрепленной к домашнему заданию. В базе данных расположена 1 таблица с товарами, в таблице имеются следующие колонки: id - первичный ключ, title - название товара, category - категория товара (FD - еда, EL - электроника, CL - одежда), price - цена за единицу товара, quantity - количество товаров на складе.
1. Написать SQL запрос, который бы удалил все товары с ценой 0 долларов.
2. Написать SQL запрос UPDATE, который бы снизил цену на 10% всех товаров в категории EL (электроника), у которых цена выше 100 долларов.
3. Написать SQL запрос INSERT, который бы добавил 3 товара в каждую из категорий.
4. Написать SQL запрос, который бы показал все колонки из таблицы товаров, которых на складе осталось больше 10-ти штук, а также необходимо отсортировать товары по категории и по цене от большей к меньшей.
5. Написать SQL запрос, который бы показал все колонки из таблицы товаров из категорий FD - еда и CL - одежда, цена которых в диапазоне от 10 до 50 долларов. Также необходимо просчитать в отдельной колонке результатов какова будет выручка при продаже всего количества товаров по указанной цене за единицу.

УРОК 2 - Реляционные базы данных

ТЕМЫ:

A. Реляционные базы данных
B. Агрегирование данных

ДЕТАЛИ:

- Реляционные базы данных (Foreign Key)
- Соединения таблиц (JOINS, UNION ALL) https://shra.ru/2017/09/sql-join-v-primerakh-s-opisaniem/
- Типы соотношений таблиц (one-to-one, one-to-many, many-to-one, many-to-many)
- Инструмент DBeaver
- Агрегационные функции и группировка данных (GROUP BY, HAVING)
- Вложенные запросы (Subqueries)
- Представления (VIEWS)

ДЗ:

Общее описание: SQL запросы описанные ниже необходимо написать и запустить в базе данных hw_3.db прикрепленной к домашнему заданию. В базе данных расположено 3 таблицы:
products таблица с товарами, в таблице имеются следующие колонки: id - первичный ключ, title - название товара, category - категория товара (FD - еда, EL - электроника, CL - одежда), price - цена за единицу товара, quantity - количество товаров на складе,
employees таблица с сотрудниками в таблице имеются следующие колонки: id - первичный ключ, full_name - фио сотрудника, salary - зарплата сотрудника, hobby - хобби, birth_date - дата рождения сотрудника и is_married - семейное положение,
sales таблица с записями каждой продажи единицы товара, в таблице имеются следующие колонки: id - первичный ключ, product_id - внешний ключ на таблицу products (то есть какой товар был продан), seller_id - внешний ключ на таблицу employees (то есть кем товар был продан) и sale_date - дата продажи.

1. Написать SQL запрос, который бы сформировал отчет по продажам каждой категории товара в отдельных месяцах, в отчете должны быть следующие колонки:

a) year_month - год и месяц в котором были продажи (подсказка используйте функцию strftime).

b) category - категория товара

c) quantity - количество проданных товаров

d) average_price - средняя цена проданных товаров дробную часть округлить до сотых

e) sales_amount - сумма продаж товара - дробную часть округлить до сотых

Отсортировать данные по колонкам year_month и category. Результат должен быть как на картинке result-1:

	year_month	category	quantity	avarage_price	sales_amount
1	2023-01	CL	14	83.06	1162.9
2	2023-01	EL	6	316.83	1901
3	2023-01	FD	8	7.75	62
4	2023-02	CL	15	169.29	2539.35
5	2023-02	EL	8	272.6	2180.8
6	2023-02	FD	5	2.6	13
7	2023-03	CL	21	129.8	2725.79
8	2023-03	EL	15	314.94	4724.05
9	2023-03	FD	20	3.36	67.2

2. Создать представление (VIEW) total_employee_sales в базе данных hw_3.db, в котором будут следующие данные:

a) employee - фио продавца

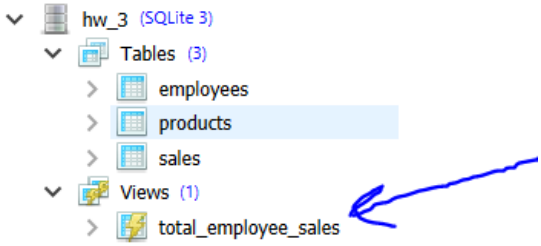
b) salary - зарплата продавца

c) quantity - количество проданных товаров

d) sales_amount - сумма продаж товара - дробную часть округлить до сотых

Отсортировать данные по колонке sales_amount. Также аггрегированные данные необходимо отфильтровать следующим образом: показывать только тех сотрудников, кто продал 10 и более единиц товара.

В результате в БД должен появиться view как на картинке result-2:



3. Сделать выборку всех колонок и записей из представления total_employee_sales. Результат должен быть как на картинке result-3:

	employee	salary	quantity	sales_amount
1	Diana Julls	1800	22	4163.2
2	Mark Daniels	1500	14	3113.95
3	Mark Daniels	1500	28	2906.96
4	Diana Julls	1800	11	1690.85
5	Diana Julls	1800	11	1482.85
6	Mark Daniels	1500	14	1298.7

УРОК 3 - Работа с фреймворком Pytest

ТЕМЫ:

A. Фреймфорк Pytest
B. Написание и запуск unit тестов в Python

ДЕТАЛИ:

- Импортирование библиотеки Pytest
- Создание тестовых файлов и тестовых функций. Оператор assert
- Запуск всех тестовых файлов с помощью pytest –v
- Запуск определенных файлов с помощью pytest <filename> -v
- Запуск тестов путем сопоставления подстрок pytest -k <substring> -v
- Запуск тестов по маркерам pytest -m <marker_name> -v
- Создание фикстур с помощью @pytest.fixture.
- conftest.py позволяет получить доступ к фикстурам из разных файлов
- Оператор yield
- Очередность запуска тестов с помощью библиотеки pytest-ordering
- Использование параметризированных тестов @pytest.mark.parametrize
- Генерация результатов в виде xml файлов pytest -v --junitxml = "result.xml"

ДЗ:

Общее описание: В предоставленном модуле hw_4.py в классе Utilities необходимо протестировать методы с помощью 2-х юнит-тестов (test_first_method.py и test_second_method.py), используя библиотеки pytest и pytest-ordering:

1. В файле test_first_method.py, в котором один параметризированный тест (test_is_positive), необходимо дополнить аннотацию параметрами из 4-х пар (входное число и результат): a) входное число на проверку позитивности 1 и результат ответа метода True; b) входное число на проверку позитивности -2 и результат ответа метода False; c) входное число на проверку позитивности 0 и результат ответа метода False; d) входное число на проверку позитивности 4 и результат ответа метода False. В теле метода всего 1 строка с оператором assert, который проверяет на соответствие результат полученный после вызова метода в классе Utilities “def is_positive(number):” с аргументом входного числа из параметров аннотации и результат из параметров аннотации.
2. Запустить файл с тестом test_first_method.py и результаты сохранить в файл first_result.xml.
3. В файле test_second_method.py вложена готовая фикстура и 4 тест-функции, которые необходимо дополнить в соответствии с описаниями в комментариях.
4. Запустить файл с тестом test_second_method.py и результаты сохранить в файл second_result.xml.
5. В домашнем задании необходимо отправить следующие файлы: test_first_method.py, first_result.xml, test_second_method.py и second_result.xml.

УРОК 4 - Введение в тестирование API. Инструменты тестирования

ТЕМЫ:

A. API и клиент-серверная архитектура
B. DEV Tool - Postman

ДЕТАЛИ:

- Что такое API
- Введение в клиент-серверную архитектуру
- HTTP протоколы (REST и SOAP)
- Методы HTTP запросов (GET, POST, PUT/PATCH, DELETE)
- Статус коды ответов
- XML и JSON
- API и рассмотрение примера документации
- Ручное тестирование API на Postman
- Коллекции, переменные и реализация автотестов в Postman

ДЗ:

Общее описание: В Postman необходимо протестировать API employees в соответствии с прикрепленной документацией в файле Документация Employees api.docx.

1. Создать коллекцию под названием hw_5, в коллекции создать переменные: name_var со значением “Ваше имя” surname_var со значением “Ваша фамилия”, переменную salary_var со значением “1000” и переменную department_var со значением “IT”.
2. В коллекции создать папку Positive_tests.
3. В папке Positive_tests создать GET запрос с названием Get All Employees и протестировать получение списка всех сотрудников.
4. В папке Positive_tests создать POST запрос с названием Create Employee и протестировать создание нового сотрудника. В теле запроса в JSON в качестве значения для для каждого поля необходимо подставить значения переменных созданных в коллекции.
5. Во вкладке Tests добавить автотест для запроса Create Employee на JavaScript, который бы после получения ответа сохранял поле id в переменную коллекции под названием employee_id_var.
6. Во складке Tests добавить автотест для запроса Create Employee на JavaScript, который бы после получения ответа проверял все поля кроме id на соответствие значениям переменных в коллекции: name_var, surname_var, salary_var и department_var.
7. В папке Positive_tests создать GET запрос с названием Get Employee by ID в URL запроса необходимо подставить значение id из переменной employee_id_var и протестировать получение информации сотрудника по ID, созданного в пункте 4.
8. В коллекции создать еще одну переменную: new_salary_var со значением “2000”.
9. В папке Positive_tests создать PUT запрос с названием Update Employee и протестировать изменение данных сотрудника, созданного в пункте 4. При этом необходимо изменить только зарплату сотрудника на значение переменной new_salary_var, а id сотрудника подставить из переменной employee_id_var.

10. Во складке Tests добавить автотест для запроса Update Employee на JavaScript, который бы после получения ответа проверял соответствие нового значения поля salary и значения переменной new_salary_var.
11. В папке Positive_tests создать DELETE запрос с названием Delete Employee в URL запроса необходимо подставить значение id из переменной employee_id_var и протестировать удаление сотрудника, созданного в пункте 4 .
12. В коллекции создать папку Negative_tests.
13. В папку Negative_tests скопировать запросы Get Employee by ID, Update Employee и Delete Employee.
14. В папке Positive_tests во вкладке Tests создать автотест на проверку ответа на статус 200.
15. В папке Negative_tests во вкладке Tests создать автотест на проверку ответа на статус 404.
16. Запустить вашу коллекцию на выполнение всех автотестов.
17. Экспортировать вашу коллекцию в файл.

УРОК 5 - Основы автоматизации тестирования API

ТЕМЫ:

A. Разработка программы для тестирования API в процедурном стиле
B. Модуль requests, примеры HTTP запросов в Python

ДЕТАЛИ:

- Отправка запроса с методом GET
- Отправка нескольких запросов метода GET
- Отправка запроса с методом POST
- Отправка запроса с методом PUT
- Отправка запроса с методом DELETE
- Проверка результатов запроса

ДЗ:
Общее описание: Написать тесты на языке Python используя библиотеки pytest и requests. Необходимо протестировать API employees в соответствии с прикрепленной документацией в файле Документация Employees api.docx.
1. Создать модуль под названием test_employees_api.py
2. В модуле добавить тест-функцию с названием test_get_all_employees, в которой необходимо протестировать GET запрос на получение списка всех сотрудников. В тесте нужно проверить статус код ответа на соответствие коду 200, а также проверить соответствие полученного тела результата является списком.
3. В модуле добавить тест-функцию с названием test_create_employee, в которой необходимо протестировать POST запрос на добавление нового сотрудника. В тесте нужно проверить статус код ответа на соответствие коду 200.
4. В модуле добавить тест-функцию с названием test_create_wrong_employee, в которой необходимо протестировать POST запрос на добавление нового сотрудника с отправкой неполных данных в теле запроса (отправить нужно только имя). В тесте нужно проверить статус код ответа на соответствие коду 400.
5. В модуле добавить тест-функцию с названием test_get_created_employee, в которой необходимо протестировать GET запрос на получение информации о сотруднике, добавленном в пункте 3. В тесте нужно проверить статус код ответа на соответствие коду 200.
6. В модуле добавить тест-функцию с названием test_update_employee, в которой необходимо протестировать PUT запрос на изменение информации о сотруднике, добавленном в пункте 3 (необходимо поменять только зарплату сотрудника). В тесте нужно проверить статус код ответа на соответствие коду 200, а также значение поля в теле ответа на соответствие значению, заданному в теле запроса новой зарплаты.
7. В модуле добавить тест-функцию с названием test_delete_employee, в которой необходимо протестировать DELETE запрос на удаление сотрудника, добавленного в пункте 3. В тесте нужно проверить статус код ответа на соответствие коду 200, а также проверить соответствие поля info в ответе на запрос так, как заявлено в документации.

8. В модуле добавить тест-функцию с названием test_delete_forbitten_employee, в которой необходимо протестировать DELETE запрос на удаление сотрудника с id = 11. В тесте нужно проверить статус код ответа на соответствие коду 403.
9. В модуле добавить тест-функцию с названием test_get_deleted_employee, в которой необходимо протестировать GET запрос на получение информации о сотруднике, добавленном в пункте 3. В тесте нужно проверить статус код ответа на соответствие коду 404.
10. Запустить все тест-функции с помощью pytest и результаты сохранить в файл employee_api_result.xml.

УРОК 6 - Объектно ориентированное программирование в Python

ТЕМЫ:

A. Суть ООП, классы и объекты
B. Инкапсуляция
C. Атрибуты уровня класса, типы методов

ДЕТАЛИ:

- Что такое класс и что такое объект (Пример класса Car)
- Атрибуты объекта
- Конструктор (__init__(self))
- Значения по умолчанию в конструкторе
- Создание объектов (аргументы с названием атрибута и без)
- Адрес объекта - self
- Методы
- Основные принципы ООП - Инкапсуляция
- Модификаторы доступа public, private
- Создание геттеров и сеттеров, аннотации @property и имя_свойства_геттера.setter
- Сокрытие методов
- Методы класса @classmethod
- Статические методы класса @staticmethod

ДЗ:

1. Создать класс Employee (Сотрудник) с приватными атрибутами/полями first_name (имя), last_name (фамилия), position (должность) и experience (стаж в годах).
2. Добавить геттеры и сеттеры для каждого поля.
3. Добавить в класс Employee атрибут уровня класса standart_number_of_vacation (стандартное количество отпускных дней) и присвоить ему значение 28.
4. Добавить в класс Employee приватный, возвращаемый метод calculate_vacation, в котором рассчитывается фактическое количество дней отпуска для каждого сотрудника и возвращается в качестве результата. По следующей формуле, если должность сотрудника - преподаватель, то к стандартному количеству дней добавляется еще 15 дополнительных дней, а для всех остальных должностей к стандартному количеству дней добавляется по 2 дополнительных дня за каждый год стажа свыше 10 лет.
5. Добавить публичный метод в класс Employee info, в котором бы распечатывалась информация о сотруднике в следующем формате:

   ФИО: Асанов Максат
   ДОЛЖНОСТЬ: программист
   СТАЖ РАБОТЫ: 12 лет
   ОТПУСКНЫЕ: 28 дней + БОНУС 4 дня = 32 дня

6. В запускаемой части файла создать 3 объекта на основе класса Employee (преподавателя, дизайнера (стаж работы 15 лет) и бухгалтера (стаж работы 4 года)).
7. Затем распечатать полную информацию о каждом сотруднике с помощью метода info.

УРОК 7 - Построение проекта по автоматизации

ТЕМЫ:

A. Разработка проекта для тестирования API в ООП стиле
B. Тестирование API с библиотекой Pytest

ДЕТАЛИ:

- Построение структуры проекта. Создание кастомных методов для запросов
- Создание теста. Метод POST
- Метод GET, PUT, DELETE
- Метод для проверки статус кода
- Метод для проверки обязательных полей
- Метод для проверки содержимого полей ответа

ДЗ:
Общее описание: Написать тесты на языке Python в ООР стиле в проекте выполненном на 7-м уроке. Необходимо протестировать API Google Maps в соответствии с прикрепленной документацией в файле Документация Google api.docx.
1. В директории utils создать модуль google_maps_api.py.
2. В модуле google_maps_api.py создать класс для тестирования GoRestUsersAPI.
3. В классе GoRestUsersAPI создать переменную класса base_url, где необходимо сохранить базовую ссылку API и переменную класса key, куда необходимо поместить параметр ключа для всех запросов.
4. В классе GoRestUsersAPI создать статичный метод для создания новой локации - create_new_location(body), в методе необходимо распечатать url запроса, выполнить post запрос, распечатать тело ответа и вернуть методом результат запроса.
5. В классе GoRestUsersAPI создать статичный метод для проверки информации локации по place ID - get_location(place_id), в методе необходимо распечатать url запроса, выполнить get запрос, распечатать тело ответа и вернуть методом результат запроса.
6. В классе GoRestUsersAPI создать статичный метод для изменения адреса локации по place ID - update_location_address(place_id, body), в методе необходимо распечатать url запроса, выполнить put запрос, распечатать тело ответа и вернуть методом результат запроса.
7. В классе GoRestUsersAPI создать статичный метод для удаления локации по place ID - delete_location(place_id), в методе необходимо распечатать url запроса, выполнить delete запрос, распечатать тело ответа и вернуть методом результат запроса.
8. В директории test_api_oop создать модуль test_google_map_api.py.
9. В модуле test_google_map_api.py создать класс для тестирования TestGoogleAPI
10. В классе TestGoogleAPI создать статичный метод для полного цикла тестирования - test_full_cycle.
11. В теле метода test_full_cycle_google_maps создать переменную (dictionary), в которой необходимо сохранить тело JSON для создания новой локации.
12. В теле метода test_full_cycle_google_maps выполнить запрос на создание новой локации, проверить статус код ответа на соответствие 200, сохранить значение из тела ответа в переменную

place_id, сделать проверку наличия обязательных полей в ответе запроса, проверить значение поля status на соответствие значению “OK”.
13. В теле метода test_full_cycle_google_maps выполнить запрос для получения информации о созданной локации, проверить статус код ответа на соответствие 200, сделать проверку наличия обязательных полей в ответе запроса, далее проверить соответствие значения поля address в теле ответа со значением в словаре.
14. В теле метода test_full_cycle_google_maps создать переменную (dictionary), в которой необходимо сохранить тело JSON для изменения адреса локации.
15. В теле метода test_full_cycle_google_maps выполнить запрос изменения адреса созданной локации, проверить статус код ответа на соответствие 200, проверить соответствие значение поля msg в теле ответа со значением, как заявлено в документации.
16. В теле метода test_full_cycle_google_maps выполнить запрос для получения информации об измененной локации, проверить статус код ответа на соответствие 200, проверить соответствие значение поля address в теле ответа со значение словаря по ключу “address”.
17. В теле метода test_full_cycle_google_maps выполнить запрос удаления созданной локации, проверить статус код ответа на соответствие 200, проверить значение поля status на соответствие значению “OK”.
18. В теле метода test_full_cycle_google_maps выполнить запрос для получения информации об удаленной локации, проверить статус код ответа на соответствие 404, проверить соответствие значения поля msg в теле ответа со значением, как заявлено в документации.
19. Запустить тесты в директории tests, тесты должны быть в статусе PASSED.
20. Отправить весь проект в zip архиве.

УРОК 8 - Логирование и отчетность тестов

ТЕМЫ:

A. Логирования
B. Отчеты Allure
C. Итоговый тест

ДЕТАЛИ:

- Работа с файлами в Python
- Добавление нового теста в проект
- Подключение логирования в проекте
- Подключения отчетов тестирования библиотека allure-pytest
- Создание отчетности pytest -v -s --alluredir=..\test_results ..\oop_api_tests
- Объяснение сгенерированных отчетов allure

ДЗ:
Общее описание: Написать тесты на языке Python в ООР стиле в проекте выполненном на 8-м уроке. Необходимо протестировать API Employees в соответствии с прикрепленной документацией в файле Документация Employees api.docx.
1. В директории utils создать модуль employees_api.py.
2. В модуле employees_api.py создать класс для тестирования EmployeesAPI.
3. В классе EmployeesAPI создать переменную класса base_url, где необходимо сохранить базовую ссылку API.
4. В классе EmployeesAPI создать статичный метод для создания нового сотрудника - create_new_employee(body), в методе необходимо распечатать url запроса, выполнить post запрос, распечатать тело ответа и вернуть методом результат запроса.
5. В классе EmployeesAPI создать статичный метод для проверки информации сотрудника по employee ID - get_employee(employee_id), в методе необходимо распечатать url запроса, выполнить get запрос, распечатать тело ответа и вернуть методом результат запроса.
6. В классе EmployeesAPI создать статичный метод для изменения информации сотрудника по employee ID - update_employee(employee_id, body), в методе необходимо распечатать url запроса, выполнить put запрос, распечатать тело ответа и вернуть методом результат запроса.
7. В классе EmployeesAPI создать статичный метод для удаления сотрудника по employee ID - delete_employee(employee_id), в методе необходимо распечатать url запроса, выполнить delete запрос, распечатать тело ответа и вернуть методом результат запроса.
8. В директории tests создать модуль test_employees_api.py.
9. В модуле test_employees_api.py создать класс для тестирования TestEmployeesAPI
10. В классе TestEmployeesAPI создать статичный метод для полного цикла тестирования - test_full_cycle_employees.
11. В теле метода test_full_cycle_employees создать переменную (dictionary), в которой необходимо сохранить тело JSON для создания нового сотрудника.
12. В теле метода test_full_cycle_employees выполнить запрос на создание нового сотрудника, проверить статус код ответа на соответствие 200, сохранить значение из тела ответа в переменную

employee_id, сделать проверку наличия обязательных полей в ответе запроса, проверить значения всех ключей словаря на соответствие значениям полей в теле ответа.
13. В теле метода test_full_cycle_employees выполнить запрос для получения информации о созданном сотруднике, проверить статус код ответа на соответствие 200, сделать проверку наличия обязательных полей в ответе запроса, проверить соответствие значения поля id в теле ответа со значением переменной employee_id, далее проверить значения всех ключей словаря на соответствие значениям полей в теле ответа.
14. В теле метода test_full_cycle_employees создать переменную (dictionary), в которой необходимо сохранить тело JSON для изменения зарплаты сотрудника.
15. В теле метода test_full_cycle_employees выполнить запрос изменения зарплаты созданного сотрудника, проверить статус код ответа на соответствие 200, проверить соответствие значения поля salary в теле ответа со значением словаря по ключу “salary”.
16. В теле метода test_full_cycle_employees выполнить запрос для получения информации об измененном сотруднике, проверить статус код ответа на соответствие 200, проверить соответствие значения поля salary в теле ответа со значение словаря по ключу “salary”.
17. В теле метода test_full_cycle_employees создать переменную (dictionary), в которой необходимо сохранить тело JSON для изменения отделения сотрудника, но в данном словаре должны быть не все поля сотрудника, например можно исключить зарплату.
18. В теле метода test_full_cycle_employees выполнить запрос изменения отделения созданного сотрудника, проверить статус код ответа на соответствие 400, проверить соответствие значения поля info на присутствие фразы “wrong employee data provided”.
19. В теле метода test_full_cycle_employees выполнить запрос удаления созданного сотрудника, проверить статус код ответа на соответствие 200, проверить значение поля info на присутствие фразы “was deleted”.
20. В теле метода test_full_cycle_employees выполнить запрос для получения информации об удаленном сотруднике, проверить статус код ответа на соответствие 404, проверить соответствие значения поля info на присутствие фразы “no employee”.
21. Запустить все тесты в директории tests, сформировать отчеты allure, тесты должны быть в статусе PASSED
22. Отправить весь проект в zip архиве.