# Выпускное задание курса Python

Требуется реализовать прототип системы контроля качества данных о продажах, находящихся в некоторой «витрине». Витриной в данном случае будет являться таблица СУБД, параметры подключения к которой описаны далее в этом документе. Необходимо автоматизировать выполнение проверочных SQL-запросов, и сообщать о результатах их выполнения.

## Функциональные требования к системе

Что должна позволять делать система:

* Запускать последовательность проверок через вызов файла из командной строки
* Печатать результаты выполнения проверок в консоль
* Проверки должны быть реализованы в виде параметризованных классов

## Спецификация витрины

Тип СУБД: MS SQL Server

Сервер: yand.dyndns.org

Название БД: MyDb

Логин для доступа к БД: northwind

Пароль: northwind

Название таблицы: sales

Поля:

* Date date not null - дата совершения продажи
* ItemId int not null - товар, который продали
* Category string, null - название категории, которой принадлежит товар
* Qty int not null - количество товара, который продали
* Price numeric(18,4) not null - отпускная цена единицы товара
* ClientPremium bool not null - признак того, что клиент на момент совершения покупки обладал премиум-подпиской

Данные, содержащиеся в таблице, относятся к диапазону дат с 10.10.2021 по 19.10.2021, несколько десятков на каждую дату. *При выполнении проверок качества данных мы всегда фиксируем в запросе к витрине какой-то день Date = YYYY-MM-DD.*

## Требования к проверкам качества

Проверка качества - логическая единица, возвращающая SQL-запрос, выполнение которого необходимо для оценки качества данных на определенную дату по определенному критерию. Каждую проверку качества следует оформить в виде отдельного класса, и все проверки наследуются от базового класса, код которого будет приведен ниже.

Критерии качества для бизнес-проверок, которые следует реализовать:

* Количество товара (Qty) строго больше нуля (отсутствуют "пустые" покупки). Критерий успешности проверки: более 98% записей удовлетворяют условию.
* Заполнена категория товара (Category). Критерий успешности проверки: более 95% записей удовлетворяют условию.
* Есть покупки от премиум-клиентов (ClientIsPremium) в категории (Category) "Ювелирные изделия". Критерий успешности проверки: булевой результат - есть покупки или нет.
* Присутствуют покупки из категории (Category) "Еда". Критерий успешности проверки: булевой результат - есть покупки, или нет.

Функциональные требования:

* Оформить каждый тип проверки в отдельный класс
* Написать unit-тесты, проверяющие ожидания того, какой SQL-запрос формирует проверка

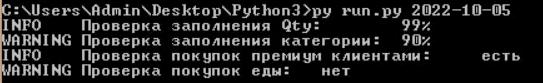
## Требования к пользовательскому интерфейсу взаимодействия с системой

В качестве инструмента взаимодействия с системой будет выступать скрипт, вызываемый из командной строки. Внутри скрипта читается входной параметр командной строки, последовательно выполняются проверки из приведённого выше списка проверок и для каждой из них печатается результат выполнения. Дата, на которую нужно выполнить проверку, передается как аргумент командной строки. Параметры подключения к базе данных можно сохранить непосредственно в коде скрипта.

Функциональные требования:

* Соединяться с базой данных, хранящей витрину
* Получать для каждой проверки SQL-запрос, отправлять его в БД, анализировать и печатать результат
* Для печати результатов использовать модуль logging. В случае прохождения проверки - INFO, в случае непрохождения – WARNING

Пример вызова и работы:



## Детали реализации

Решение должно быть реализовано в виде 4 файлов (пожалуйста, никаких вложенных папок!):

* classes.py – содержит все классы проверок и их базовый класс;
* tests.py – содержит все unit тесты для каждого из классов проверок;
* logger.py – содержит конфигурацию логирования для всей системы;
* run.py – скрипт, запускаемый пользователем из командной строки при работе с системой.

Код базового класса проверок:

from datetime import date

class ValidatorBase:

def \_\_init\_\_(self, date\_of\_interest: date):

self.date\_of\_interest = date\_of\_interest

def query(self) -> str:

# Должен возвращать sql запрос, необходимый

# для проведения проверки на дату date\_of\_interest

raise NotImplementedError

## Как предоставить решение

1. Сделайте на гитхабе **приватный** репозиторий
2. Пригласите в него коллаборатора yand1960 (это ваш преподаватель)
3. Впишите в документ <https://docs.google.com/spreadsheets/d/1sUp2_27-3QQG3uHwchtTHrIQ5B-LB-FiBkYNB1-GT9c/edit?usp=sharing> свои имя/фамилию и адрес этого репозитория
4. В репозиторий поместите 4 файла вашего решения. Пожалуйста, никаких вложенных папок, все файлы должны быть в корне репозитория!
5. В случае каких-то авралов пишите [yand@outlook.com](mailto:yand@outlook.com)

Срок: до 23:59 9 октября 2022