Отчёт по заданию №8

Файл courses.py:

Описание: генерация списков курсов и пользователей и сохранение их в файлы JSON для дальнейшего использования.

Код:

import datetime  
import json  
import numpy as np  
from faker import Faker  
def generate\_courses\_list():  
 *'''  
 Функция для генерации списка курсов и пользователей и сохранения в виде JSON-файлов.  
 :return: courses\_list - список со словарями курсов, attendees\_list - список со словарями пользователей.  
 '''* np\_seed = 73  
 np.random.seed(np\_seed)  
 date\_start = datetime.date(2023, 2, 6)  
 date\_finish = datetime.date(2023, 6, 5)  
 num\_courses = 5  
 num\_attendees = 50  
 courses\_list = []  
 faker\_inst = Faker()  
 faker\_inst.seed\_locale("en\_US", 73)  
 for i in range(num\_courses):  
 d1 = faker\_inst.date\_between(start\_date=date\_start, end\_date=date\_finish)  
 d2 = faker\_inst.date\_between(start\_date=d1, end\_date=date\_finish)  
 dict\_obj = {"\_id": i + 1, "name": "Course #" + str(i + 1), "duration": np.random.randint(3, 13) \* 10,  
 "date\_begin": d1.isoformat(), "date\_end": d2.isoformat(),  
 "description": "Course #" + str(i + 1) + " description.", "lecturer": faker\_inst.unique.name(),  
 "price": np.random.randint(2, 9) \* 10000}  
 courses\_list.append(dict\_obj)  
 with open("courses\_list.json", "w") as courses\_list\_file:  
 json.dump(courses\_list, courses\_list\_file)  
 courses\_list\_file.close()  
 attendees\_list = []  
 for i in range(num\_attendees):  
 d = faker\_inst.date\_between(start\_date=datetime.date(1990, 1, 1),  
 end\_date=datetime.date(2010, 12, 31)).isoformat()  
 dict\_obj = {"\_id": i + 1, "course\_id": np.random.randint(1, num\_courses + 1), "name": faker\_inst.unique.name(),  
 "birthdate": d, "corporate\_client": bool(np.random.randint(0, 2))}  
 country\_name = faker\_inst.country()  
 dict\_obj["country"] = country\_name  
 dict\_obj["city"] = faker\_inst.city()  
 dict\_obj["phone\_number"] = faker\_inst.unique.msisdn()  
 dict\_obj["online"] = bool(np.random.randint(0, 2))  
 dict\_obj["wishes"] = "Wishes for " + str(dict\_obj["name"] + ".")  
 attendees\_list.append(dict\_obj)  
 with open("attendees\_list.json", "w") as attendees\_list\_file:  
 json.dump(attendees\_list, attendees\_list\_file)  
 attendees\_list\_file.close()  
 return courses\_list, attendees\_list

Файл main.py:

Описание: чтение список курсов и пользователей из файлов JSON или генерация список при отсутствии файлов и основная логика для выполнения заданий. Для создания базы данных и поиска объектов была использована библиотека Py2neo.

Код:

import datetime  
import json  
import os  
from dateutil import relativedelta  
import courses  
if not os.path.isfile("courses\_list.json") or not os.path.isfile("attendees\_list.json"):  
 courses\_list, attendees\_list = courses.generate\_courses\_list()  
 print("Сгенерированы новые списки курсов и их посетителей.")  
else:  
 f = open("courses\_list.json", "r")  
 courses\_list = json.load(f)  
 f.close()  
 f = open("attendees\_list.json", "r")  
 attendees\_list = json.load(f)  
 f.close()  
 print("Списки курсов и их посетителей загружены из JSON-файлов.")  
for i, course in enumerate(courses\_list):  
 courses\_list[i]["date\_begin"] = datetime.datetime.strptime(course["date\_begin"], "%Y-%m-%d")  
 courses\_list[i]["date\_end"] = datetime.datetime.strptime(course["date\_end"], "%Y-%m-%d")  
for i, attendee in enumerate(attendees\_list):  
 attendees\_list[i]["birthdate"] = datetime.datetime.strptime(attendee["birthdate"], "%Y-%m-%d")  
print("Количество курсов: " + str(len(courses\_list)) + ".")  
print("Количество посетителей курсов: " + str(len(attendees\_list)) + ".")  
from py2neo import Graph, Node, Relationship, NodeMatcher, RelationshipMatcher  
courses\_nodes = []  
for course in courses\_list:  
 n = Node("Course", name=course["name"], duration=course["duration"], date\_begin=course["date\_begin"],  
 date\_end=course["date\_end"], description=course["description"], lecturer=course["lecturer"],  
 price=course["price"])  
 courses\_nodes.append(n)  
attendees\_nodes = []  
relationships = []  
for attendee in attendees\_list:  
 n = Node("Attendee", name=attendee["name"], birthdate=attendee["birthdate"],  
 corporate\_client=attendee["corporate\_client"], country=attendee["country"], city=attendee["city"],  
 phone\_number=attendee["phone\_number"], online=attendee["online"], wishes=attendee["wishes"])  
 attendees\_nodes.append(n)  
 relationships.append(Relationship(n, "ATTENDS", courses\_nodes[attendee["course\_id"] - 1]))  
graph = Graph(password="SQL\_is\_better")  
graph.delete\_all()  
tx = graph.begin()  
for course\_node in courses\_nodes:  
 tx.create(course\_node)  
for attendee\_node in attendees\_nodes:  
 tx.create(attendee\_node)  
for relationship in relationships:  
 tx.create(relationship)  
graph.commit(tx)  
nm = NodeMatcher(graph)  
rm = RelationshipMatcher(graph)

1. Вывести средний возраст пользователей по каждому курсу.  
   Код:

print("Задание 1.")  
today\_without\_time = datetime.datetime.fromordinal(datetime.date.today().toordinal())  
c\_nodes = nm.match("Course")  
for c\_node in c\_nodes:  
 r\_nodes = RelationshipMatcher(graph).match((None, c\_node), "ATTENDS")  
 a\_nodes\_age = 0.0  
 for r\_node in r\_nodes:  
 a\_nodes\_age += relativedelta.relativedelta(today\_without\_time, r\_node.start\_node["birthdate"].to\_native()).years  
 a\_nodes\_age /= len(r\_nodes)  
 print(c\_node["name"] + ": " + str(a\_nodes\_age) + ".")

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

1. Вывести курсы, в которых физических клиентов больше, чем корпоративных клиентов, и их процентное соотношение.

Код:

print("Задание 2.")  
b = False  
for c\_node in c\_nodes:  
 r\_nodes = RelationshipMatcher(graph).match((None, c\_node), "ATTENDS")  
 a\_nodes\_noncorp\_count = 0  
 for r\_node in r\_nodes:  
 if not r\_node.start\_node["corporate\_client"]:  
 a\_nodes\_noncorp\_count += 1  
 f = a\_nodes\_noncorp\_count / len(r\_nodes) \* 100  
 if f > 50:  
 print(c\_node["name"] + ": " + str(f))  
 b = True  
if not b:  
 print("Таких курсов нет.")

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

1. Вывести курсы, которые уже завершились.

Код:

print("Задание 3.")  
b = False  
for c\_node in c\_nodes:  
 if today\_without\_time > c\_node["date\_end"]:  
 print(c\_node["name"] + ": " + str(c\_node["date\_end"]) + ".")  
 b = True  
if not b:  
 print("Таких курсов нет.")

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

1. Вывести курсы, которые еще предстоят, и количество записавшихся на них пользователей.

Код:

print("Задание 4.")  
b = False  
for c\_node in c\_nodes:  
 if today\_without\_time < c\_node["date\_begin"]:  
 r\_nodes = RelationshipMatcher(graph).match((None, c\_node), "ATTENDS")  
 print(c\_node["name"] + ": " + str(len(r\_nodes) + "."))  
 b = True  
if not b:  
 print("Таких курсов нет.")

Вывод:



1. Вывести полученную прибыль от курсов, которые завершились и уже проходят.

Код:

print("Задание 5.")  
f = 0  
for c\_node in c\_nodes:  
 if today\_without\_time > c\_node["date\_end"] or today\_without\_time < c\_node["date\_begin"]:  
 f += c\_node["price"]  
print("Полная прибыль: " + str(f) + ".")

Вывод:

