Отчёт по заданию №7

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.

Код:

**import** datetime  
**import** json  
**import** os  
**import** pymongo  
**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)) + ".")  
client = pymongo.MongoClient("localhost", 27017)  
db = client["CoursesDB"]  
courses\_collection = db["Courses"]  
courses\_collection.drop()  
courses\_collection.insert\_many(courses\_list)  
print("Данные о курсах добавлены в коллекцию Courses.")  
**del** courses\_list  
attendees\_collection = db["Attendees"]  
attendees\_collection.drop()  
attendees\_collection.insert\_many(attendees\_list)  
print("Данные о посетителях курсов добавлены в коллекцию Attendees.")  
**del** attendees\_list

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

print("Задание 1.")  
stage\_lookup\_attendees = {  
 "$lookup": {"from": "Attendees", "localField": "\_id", "foreignField": "course\_id", "as": "related\_attendees"}}  
stage\_sum\_age\_attendees = {"$reduce": {"input": "$related\_attendees.birthdate", "initialValue": "", "in": {  
 "$sum": [{"$subtract": [{"$year": "$$NOW"}, {"$year": "$$this"}]}, "$$value"]}}}  
stage\_add\_attendees\_avg\_age = {  
 "$addFields": {"avg\_age": {"$divide": [stage\_sum\_age\_attendees, {"$size": "$related\_attendees.birthdate"}]}}}  
pipeline = [stage\_lookup\_attendees, stage\_add\_attendees\_avg\_age]  
results = courses\_collection.aggregate(pipeline)  
**for** course **in** results:  
 print(course["name"] + ": " + str(course["avg\_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.

print("Задание 2.")  
stage\_sum\_corp\_attendees = {"$reduce": {"input": "$related\_attendees.corporate\_client", "initialValue": "",  
 "in": {"$sum": ["$$value", {"$toInt": "$$this"}]}}}  
stage\_perc\_noncorp\_attendees = {"$subtract": [100, {  
 "$multiply": [{"$divide": [stage\_sum\_corp\_attendees, {"$size": "$related\_attendees.corporate\_client"}]}, 100]}]}  
stage\_add\_corp\_attendees\_count = {"$addFields": {"corp\_clients": stage\_sum\_corp\_attendees}}  
stage\_add\_non\_corp\_attendees\_perc = {"$addFields": {"noncorp\_clients\_perc": stage\_perc\_noncorp\_attendees}}  
stage\_match\_non\_corp\_perc\_over50 = {"$match": {"noncorp\_clients\_perc": {"$gt": 50}}}  
pipeline = [stage\_lookup\_attendees, stage\_add\_corp\_attendees\_count, stage\_add\_non\_corp\_attendees\_perc,  
 stage\_match\_non\_corp\_perc\_over50]  
k = 0  
results = courses\_collection.aggregate(pipeline)  
**for** course **in** results:  
 print(course["name"] + ": " + str(course["noncorp\_clients\_perc"]) + ".")  
 k += 1  
**if** k < 1:  
 print("Таких курсов нет.")

Вывод:  
 Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

1. Вывести курсы, которые уже завершились.

print("Задание 3.")  
today\_without\_time = datetime.datetime.fromordinal(datetime.date.today().toordinal())  
print("Полночь сегодня: " + str(today\_without\_time) + ".")  
stage\_match\_course\_ended = {"$match": {"date\_end": {"$lt": today\_without\_time}}}  
pipeline = [stage\_match\_course\_ended]  
results = courses\_collection.aggregate(pipeline)  
k = 0  
**for** course **in** results:  
 print(course["name"] + ": " + str(course["date\_end"]) + ".")  
 k += 1  
**if** k < 1:  
 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.

print("Задание 4.")  
stage\_add\_count\_attendees = {"$addFields": {"attendees\_amount": {"$size": "$related\_attendees"}}}  
stage\_match\_course\_future = {"$match": {"date\_begin": {"$gt": today\_without\_time}}}  
pipeline = [stage\_lookup\_attendees, stage\_add\_count\_attendees, stage\_match\_course\_future]  
results = courses\_collection.aggregate(pipeline)  
k = 0  
**for** course **in** results:  
 print(course["name"] + ": " + str(course["attendees\_amount"]) + ".")  
 k += 1  
**if** k < 1:  
 print("Таких курсов нет.")

Вывод:  


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

print("Задание 5.")  
stage\_match\_course\_ended\_or\_current = {  
 "$match": {"$or": [{"date\_end": {"$lt": today\_without\_time}}, {"date\_start": {"$lt": today\_without\_time}}]}}  
stage\_group\_sum\_price = {"$group": {"\_id": -1, "total\_price": {"$sum": "$price"}}}  
pipeline = [stage\_match\_course\_ended\_or\_current, stage\_group\_sum\_price]  
results = courses\_collection.aggregate(pipeline)  
**for** result **in** results:  
 print("Полная прибыль: " + str(result["total\_price"]) + ".")

Вывод:  
 