Лабораторная работа №7

Исходный код можно найти [здесь](https://github.com/Temikmus/2nd-part-of-LW_7)

**Работу выполнили**: Муслин А.М., Яшина Н.А., Языкова М.Д. , Шабаркина Д.А. (КНТ-5)

**Предметная область:**

Управление медицинский персоналом и операциями в медицинских учреждениях

**Описание схемы БД:**

1. Медперсонал

Таблица, которая хранит информацию о медицинском персонале.

Структура:

* Идентификатор
* Фамилия
* Адрес
* Налог (в процентах)

1. Место работы

Таблица, которая хранит информацию о местах работы.

Структура:

* Идентификатор
* Учреждение
* Адрес
* Отчисление (в процентах)

1. Типы операций

Таблица, которая хранит информацию о различных типах медицинских операций.

Структура:

* Идентификатор
* Наименование
* Опорный пункт
* Запасы
* Стоимость

1. Трудовая деятельность

Таблица, которая хранит информацию о трудовой деятельности медицинского персонала.

Структура:

* Договор
* Дата
* Медперсонал id
* Место работы id
* Операции id
* Количество
* Оплата

**Подключение к БД:**  
def connect\_cassandra():  
 cluster = Cluster(['127.0.0.1'])  
 session = cluster.connect()  
  
 session.execute("""  
 CREATE KEYSPACE IF NOT EXISTS my\_keyspace   
 WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};  
 """)  
  
 session.set\_keyspace('my\_keyspace')  
 return session

Создание таблиц и вставка значений

def create\_fill\_tables(session):  
 session.execute("DROP TABLE IF EXISTS operation\_types;")  
  
 session.execute("drop table if exists work\_activity;")  
  
 session.execute("drop table if exists medical\_staff;")  
  
 session.execute("drop table if exists workplaces;")  
  
 session.execute("""  
 CREATE TABLE work\_activity (  
 contract INT PRIMARY KEY,  
 date TEXT,  
 staff\_id INT,  
 workplace\_id INT,  
 operation\_id INT,  
 quantity INT,  
 payment DECIMAL  
 );  
 """)  
  
 operations1=[  
 (51040, 'Понедельник', 1, 1, 7, 4, 20000),  
 (51041, 'Понедельник', 3, 3, 6, 1, 30000),  
 (51042, 'Понедельник', 4, 3, 4, 3, 33000),  
 (51043, 'Понедельник', 4, 5, 1, 2, 36000),  
 (51044, 'Понедельник', 4, 4, 6, 1, 30000),  
 (51045, 'Среда', 2, 2, 5, 3, 30000),  
 (51046, 'Четверг', 3, 6, 4, 4, 44000),  
 (51047, 'Четверг', 4, 6, 2, 1, 28000),  
 (51048, 'Четверг', 5, 3, 3, 4, 44000),  
 (51049, 'Пятница', 2, 4, 5, 1, 10000),  
 (51050, 'Пятница', 3, 6, 4, 2, 22000),  
 (51051, 'Пятница', 3, 3, 1, 2, 36000),  
 (51052, 'Пятница', 5, 3, 2, 1, 14000),  
 (51053, 'Суббота', 3, 2, 7, 2, 10000),  
 (51054, 'Суббота', 4, 6, 4, 1, 11000),  
 (51055, 'Суббота', 5, 5, 4, 2, 22000),  
 (51056, 'Суббота', 3, 6, 3, 2, 22000)  
 ]  
  
 for contract, date, staff\_id, workplace\_id, operation\_id, quantity, payment in operations1:  
 session.execute("""  
 INSERT INTO work\_activity (contract, date, staff\_id, workplace\_id, operation\_id, quantity, payment)  
 VALUES (%s, %s, %s, %s, %s, %s, %s)  
 """, (contract, date, staff\_id, workplace\_id, operation\_id, quantity, Decimal(payment) ))  
  
 print("Данные успешно добавлены!")  
 session.execute("""  
 CREATE TABLE medical\_staff (  
 id INT PRIMARY KEY,  
 surname TEXT,  
 address TEXT,  
 tax\_percent DECIMAL  
 );  
 """)  
  
 operations2=[  
 ('Медина', 'Вознесенское', 14),  
 ('Севастьянов', 'Навашино', 14),  
 ('Бессонов', 'Выкса', 10),  
 ('Губанов', 'Выкса', 10),  
 ('Боева', 'Починки', 5)  
 ]  
 id1=1  
 for surname, address, tax\_percent in operations2:  
 session.execute("""  
 INSERT INTO medical\_staff (id, surname, address, tax\_percent)  
 VALUES (%s, %s, %s, %s)  
 """, (id1, surname, address, Decimal(tax\_percent)))  
 id1+=1  
 print("Данные успешно добавлены!")  
  
 session.execute("""  
 CREATE TABLE workplaces (  
 id INT PRIMARY KEY,  
 institution TEXT,  
 location TEXT,  
 local\_tax\_percent DECIMAL  
 );  
 """)  
  
 operations3=[  
 ('Районная больница', 'Вознесенское', 10),  
 ('Травм. пункт', 'Выкса', 3),  
 ('Больница', 'Навашино', 4),  
 ('Род. дом', 'Вознесенское', 12),  
 ('Больница', 'Починки', 4),  
 ('Травм. пункт', 'Лукояново', 3)  
 ]  
  
 id2=1  
 for institution, location, local\_tax\_percent in operations3:  
 session.execute("""  
 INSERT INTO workplaces (id, institution, location, local\_tax\_percent)  
 VALUES (%s, %s, %s, %s)  
 """, (id2, institution, location, Decimal(local\_tax\_percent)))  
 id2+=1  
 print("Данные успешно добавлены!")  
  
 session.execute("""  
 CREATE TABLE operation\_types (  
 id INT PRIMARY KEY,  
 name TEXT,  
 strong\_point TEXT,  
 stocks DECIMAL,  
 price DECIMAL  
 );  
 """)  
  
 operations4 = [  
 ('Наложение гипса', 'Выкса', 2000, 18000),  
 ('Блокада', 'Навашино', 10000, 14000),  
 ('Инъекция поливитаминов', 'Навашино', 20000, 11000),  
 ('Инъекция алоэ', 'Навашино', 12000, 11000),  
 ('ЭКГ', 'Вознесенское', 115, 10000),  
 ('УЗИ', 'Вознесенское', 20, 30000),  
 ('Флюорография', 'Выкса', 1000, 5000)  
 ]  
  
 id3=1  
 for name, strong\_point, stocks, price in operations4:  
 session.execute("""  
 INSERT INTO operation\_types (id, name, strong\_point, stocks, price)  
 VALUES (%s, %s, %s, %s, %s)  
 """, (id3, name, strong\_point, Decimal(stocks), Decimal(price)))  
 id3+=1  
  
 print("Данные успешно добавлены!")

**Разделение задач:**

Артём: 5a, 6b, 7d, 11a, 12, 13d, 14d, 15d

Нина: 4c, 6a, 7c, 10c, 11d, 13c, 14c, 15c

Маша: 4a, 5b, 7a, 10a, 11b, 13a, 14a, 15a

Даша: 4b, 5c, 7b, 10b, 11c, 13b, 14b, 15b

**Уровень 1.**

**Задание 4:**

Вывести с помощью запросов:

1. различные адреса всех медработников;
2. список всех различных медучреждений;
3. различные дни, для которых хранится информация о трудовой деятельности.

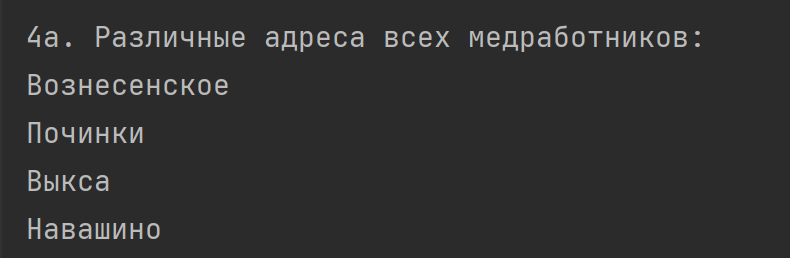
**Реализация:**

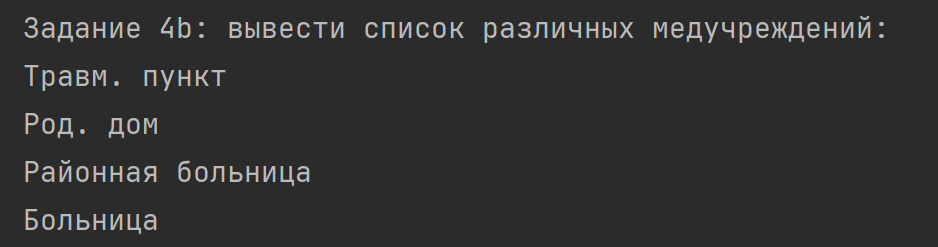
def task\_4a(session):  
 # 4a вывести различные адреса всех медработников  
 rows = session.execute("""  
 select address from medical\_staff;  
 """)  
 unique\_addresses = set(row.address for row in rows)  
 print("\n4a. Различные адреса всех медработников:")  
 for address in unique\_addresses:  
 print(address)

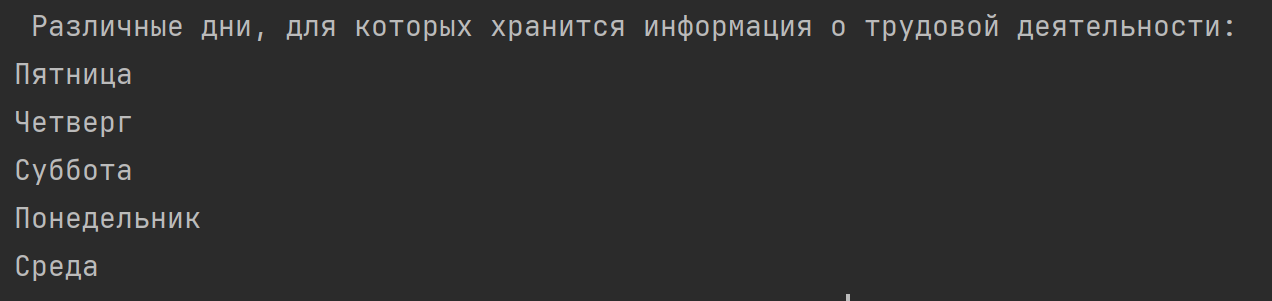
def task\_4b(session):  
 print("Задание 4b: вывести список различных медучреждений:")  
 rows = session.execute("""  
 SELECT institution FROM workplaces;  
 """)  
 unique\_institutions = set(row.institution for row in rows)  
 for institution in unique\_institutions:  
 print(institution)

def task\_4c(session):  
 rows = session.execute("SELECT date FROM work\_activity;")  
 unique\_dates = set(row.date for row in rows)  
  
 print("Ответ на задание 4с.\n Различные дни, для которых хранится информация о трудовой деятельности:")  
 for date in unique\_dates:  
 print(date)

**Результат запроса:**







**Задание 5:**

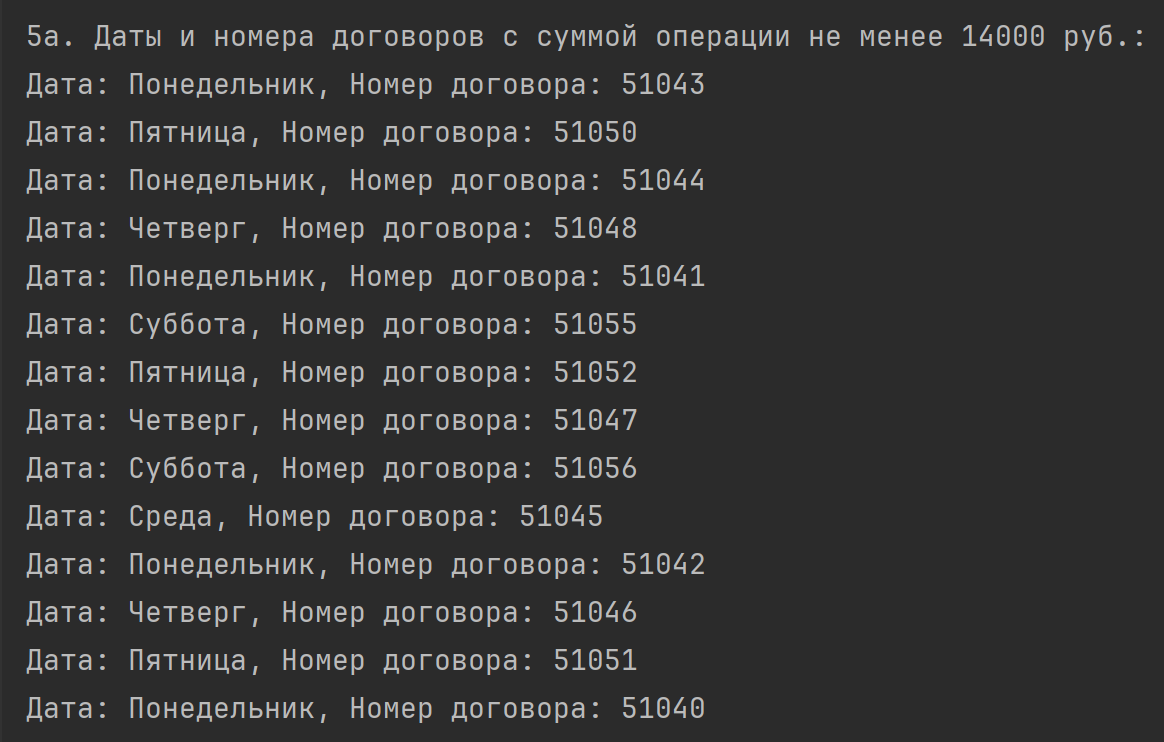
Найти:

1. даты и номера договоров, когда производились операции на сумму не менее 14000руб.

**Реализация:**

def task\_5a(session):  
 # 5a. Найти даты и номера договоров, когда производились операции на сумму не менее 14000 руб.  
 rows = session.execute("""  
 SELECT contract, date FROM work\_activity  
 WHERE payment >= 14000 ALLOW FILTERING;  
 """)  
  
 print("\n5a. Даты и номера договоров с суммой операции не менее 14000 руб.:")  
 for row in rows:  
 print(f"Дата: {row.date}, Номер договора: {row.contract}")

**Результат запроса:**

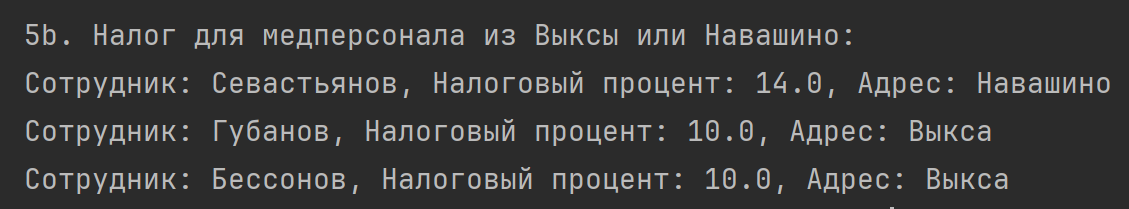
****

1. размер налога для медперсонала из Выксы или Навашино;

**Реализация:**

def task\_5b(session):  
 # 5b найти размер налога для медперсонала из Выксы или Навашино  
 rows\_2 = session.execute("""  
 select address, surname, tax\_percent from medical\_staff  
 where address IN ('Выкса', 'Навашино') allow filtering;  
 """)  
 print("\n5b. Налог для медперсонала из Выксы или Навашино:")  
 for tax in rows\_2:  
 print(f"Сотрудник: {tax.surname}, Налоговый процент: {tax.tax\_percent}, Адрес: {tax.address}")

**Результат запроса:**

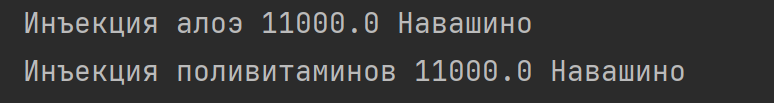


1. название, стоимость и адрес опорного пункта для операций, в названии которых есть слово “Инъекция”, и стоящих более 10000руб. Результат отсортировать по адресу и стоимости.

**Реализация:**

def task\_5c(session):  
 print(  
 "Задание 5c: вывести название, стоимость и адрес опорного пункта для операций, в названии которых есть слово Инъекция, и стоящих более 10000р. Результат отсортировать по адресу и стоимости")  
 rows = session.execute("""  
 select name, price, strong\_point  
 from operation\_types where price>10000  
 allow filtering  
 """)  
 result = list()  
 for row in rows:  
 if "Инъекция" in row.name:  
 result.append(row)  
 result.sort(key=lambda row: (row.strong\_point, row.price))  
 for res in result:  
 print(res.name, res.price, res.strong\_point)

**Результат запроса:**



**Задание 6:**

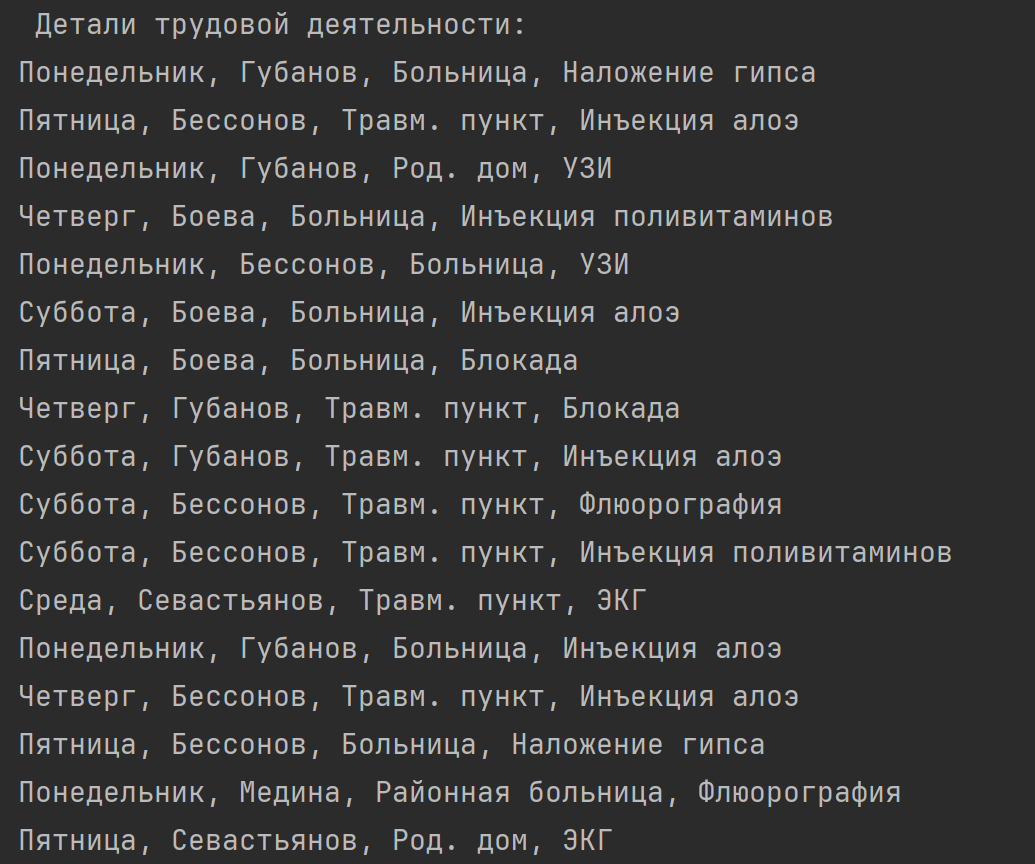
На основании данных о проведенных операциях вывести в следующем формате все записи:

1. дата, фамилия медперсонала, название места работы, название операции;

**Реализация:**

def task\_6a(session):  
 print(" Ответ на задание 6а.\n Детали трудовой деятельности:")  
 work\_rows = session.execute("SELECT \* FROM work\_activity;")  
  
 for work in work\_rows:  
 staff\_row = session.execute(  
 "SELECT surname FROM medical\_staff WHERE id = %s;", (work.staff\_id,)  
 ).one()  
 surname = staff\_row.surname if staff\_row else "Неизвестно"  
 workplace\_row = session.execute(  
 "SELECT institution FROM workplaces WHERE id = %s;", (work.workplace\_id,)  
 ).one()  
 institution = workplace\_row.institution if workplace\_row else "Неизвестно"  
 operation\_row = session.execute(  
 "SELECT name FROM operation\_types WHERE id = %s;", (work.operation\_id,)  
 ).one()  
 operation\_name = operation\_row.name if operation\_row else "Неизвестно"  
  
 print(f"{work.date}, {surname}, {institution}, {operation\_name}")

**Результат запроса:**

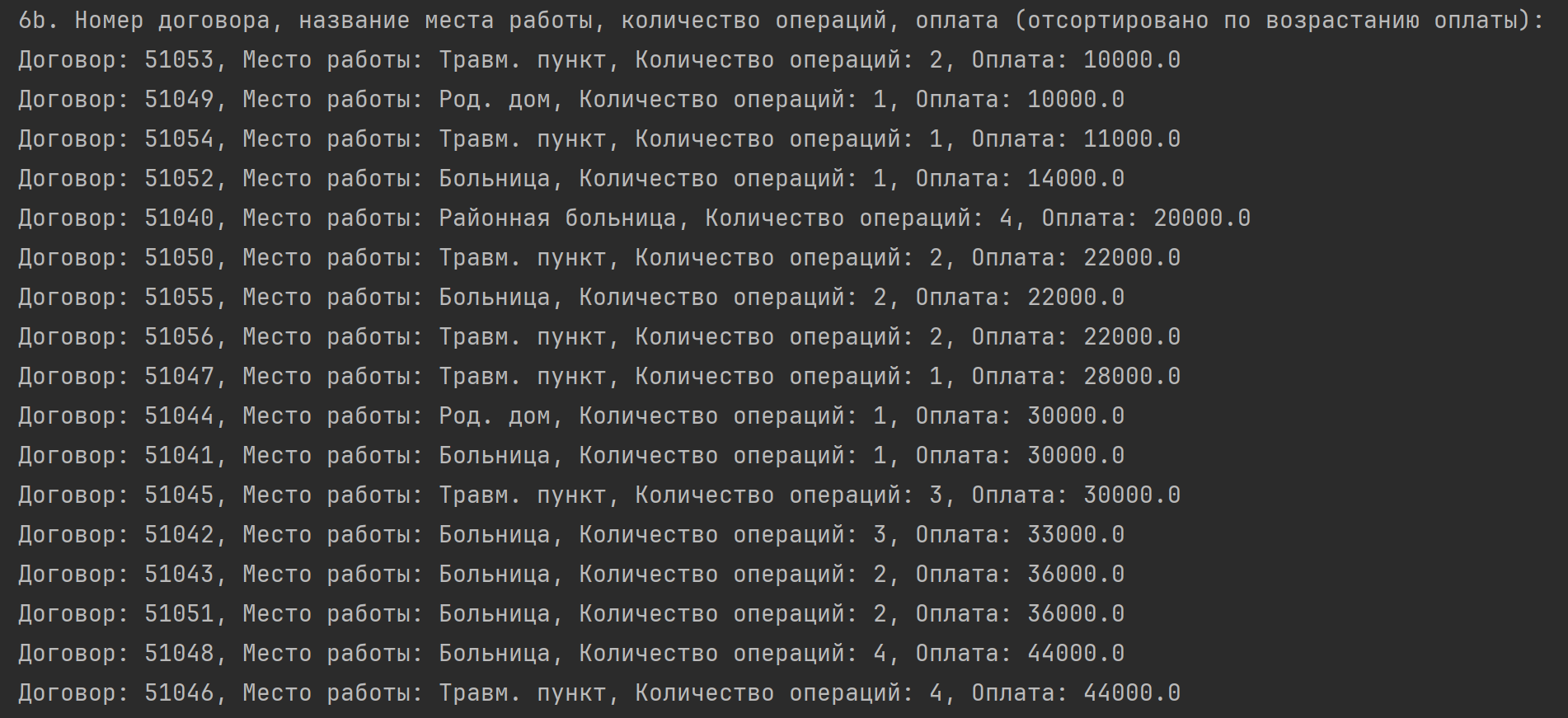


1. номер договора, название места работы, количество операций, оплата. Отсортировать по возрастанию оплаты.

**Реализация:**

def task\_6b(session):  
 # 6b. Вывести номер договора, название места работы, количество операций, оплату. Отсортировать по возрастанию оплаты.  
 rows = session.execute("""  
 SELECT contract, workplace\_id, quantity, payment  
 FROM work\_activity;  
 """)  
 workplaces = {}  
 workplace\_rows = session.execute("SELECT id, institution FROM workplaces;")  
 for row in workplace\_rows:  
 workplaces[row.id] = row.institution  
  
 sorted\_rows = sorted(rows, key=lambda x: x.payment)  
  
 print("\n6b. Номер договора, название места работы, количество операций, оплата (отсортировано по возрастанию оплаты):")  
 for row in sorted\_rows:  
 workplace\_name = workplaces.get(row.workplace\_id, "Неизвестное место работы")  
 print(f"Договор: {row.contract}, Место работы: {workplace\_name}, Количество операций: {row.quantity}, Оплата: {row.payment}")

**Результат запроса:**

****

**Задание 7:**

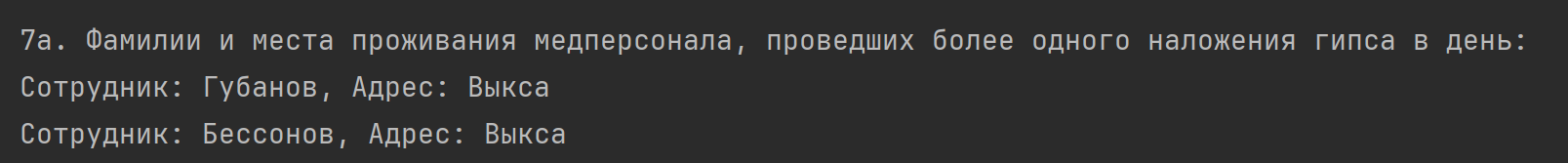
Определить:

1. фамилии и места проживания медперсонала, проведших более одного наложения гипса в день;

**Реализация:**

def task\_7a(session):  
 # 7a определить фамилии и места проживания медперсонала, проведших более одного наложения гипса в день  
 operation\_id = None  
 operations = session.execute("""  
 select id from operation\_types where name = 'Наложение гипса' allow filtering;  
 """)  
 for operation in operations:  
 operation\_id = operation.id  
  
 if operation\_id is not None:  
 rows\_3 = session.execute("""  
 select staff\_id, date, quantity  
 from work\_activity  
 where operation\_id = %s and quantity > 1 allow filtering;  
 """, (operation\_id,))  
  
 staff\_ids = {row.staff\_id for row in rows\_3}  
  
 rows\_4 = session.execute("""  
 select id, surname, address  
 from medical\_staff  
 """)  
  
 print("\n7a. Фамилии и места проживания медперсонала, проведших более одного наложения гипса в день:")  
 for row in rows\_4:  
 if row.id in staff\_ids:  
 print(f"Сотрудник: {row.surname}, Адрес: {row.address}")

**Результат запроса:**

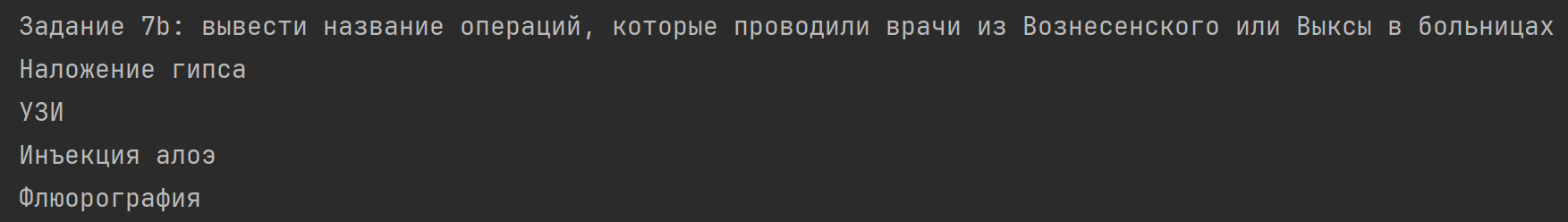
****

1. название операций, которые проводили врачи из Вознесенского или Выксы в больницах;

**Реализация:**

def task\_7b(session):  
 print("Задание 7b: вывести название операций, которые проводили врачи из Вознесенского или Выксы в больницах")  
 operation\_map = {}  
 rows\_operation = session.execute("SELECT \* FROM operation\_types")  
 for row\_operation in rows\_operation:  
 operation\_map[row\_operation.id] = (row\_operation.name)  
 place\_map = {}  
 rows\_place = session.execute("select \* from workplaces")  
 for row\_place in rows\_place:  
 place\_map[row\_place.id] = (row\_place.institution, row\_place.location)  
 staff\_map = {}  
 rows\_staff = session.execute("select \* from medical\_staff")  
 for row\_staff in rows\_staff:  
 staff\_map[row\_staff.id] = (row\_staff.address)  
 result = []  
 rows = session.execute("SELECT staff\_id, operation\_id, workplace\_id FROM work\_activity")  
 for row in rows:  
 staff\_info = staff\_map.get(row.staff\_id)  
 operation\_info = operation\_map.get(row.operation\_id)  
 place\_info = place\_map.get(row.workplace\_id)  
 if ("Вознесенское" in staff\_info or "Выкса" in staff\_info) and ("больница" in place\_info[0].lower()) and (  
 operation\_info not in result):  
 result.append((operation\_info))  
 for r in result:  
 print(r)

**Результат запроса:**

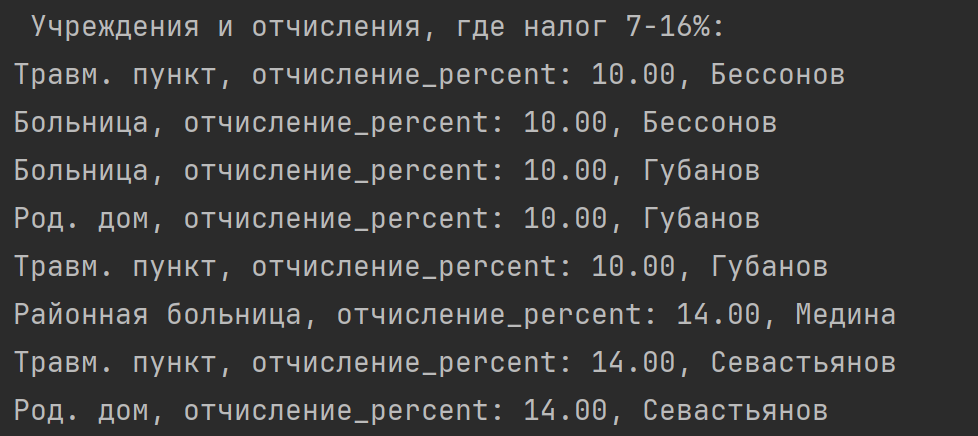


1. названия и размер отчислений в местный бюджет для тех учреждений, где проводили операции те, у кого налог не менее 7%, но не более 16%. Включить в вывод фамилии таких людей и отсортировать по размеру отчислений и налогу;

**Реализация:**

# 7.c) Учреждения и отчисления, где налог 7-16%, фамилии, сортировка по отчислениям и налогу.  
def task\_7c(session):  
 print("Ответ на задание 7с.\n Учреждения и отчисления, где налог 7-16%:")  
  
 staff\_rows = session.execute("""  
 SELECT id, surname, tax\_percent FROM medical\_staff   
 WHERE tax\_percent >= 7 AND tax\_percent <= 16 ALLOW FILTERING;  
 """)  
  
 staff\_dict = {row.id: (row.surname, float(row.tax\_percent)) for row in staff\_rows}  
 if not staff\_dict:  
 print("Нет сотрудников с налогом в диапазоне 7-16%.")  
 return  
 wp\_rows = session.execute("SELECT id, institution FROM workplaces;")  
 wp\_dict = {row.id: row.institution for row in wp\_rows}  
 work\_rows = session.execute("SELECT staff\_id, workplace\_id FROM work\_activity;")  
 tax\_data = set()  
 for work in work\_rows:  
 if work.staff\_id in staff\_dict and work.workplace\_id in wp\_dict:  
 surname, tax\_percent = staff\_dict[work.staff\_id]  
 institution = wp\_dict[work.workplace\_id]  
 tax\_data.add((institution, tax\_percent, surname))  
  
 if not tax\_data:  
 print("Нет данных для отображения.")  
 return  
  
 sorted\_data = sorted(tax\_data, key=lambda x: (x[1], x[2]))  
  
 for institution, deduction\_percent, surname in sorted\_data:  
 print(f"{institution}, отчисление\_percent: {deduction\_percent:.2f}, {surname}")

**Результат запроса:**

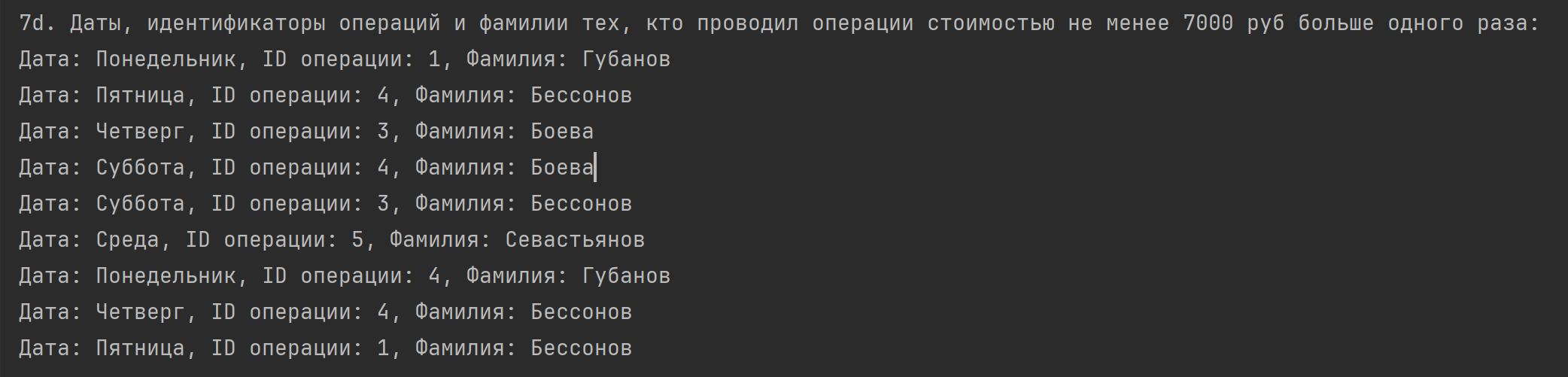
****

1. даты, идентификаторы операций и фамилии тех, кто проводил операции стоимостью не менее 7000руб больше одного раза.

**Реализация:**

def task\_7d(session):  
 # 7d. Определить даты, идентификаторы операций и фамилии тех, кто проводил операции стоимостью не менее 7000 руб больше одного раза.  
  
 work\_activity\_rows = session.execute("""  
 SELECT date, operation\_id, staff\_id, quantity  
 FROM work\_activity;  
 """)  
  
 operation\_types\_rows = session.execute("""  
 SELECT id, price  
 FROM operation\_types;  
 """)  
  
 medical\_staff\_rows = session.execute("""  
 SELECT id, surname  
 FROM medical\_staff;  
 """)  
  
 operation\_prices = {row.id: row.price for row in operation\_types\_rows}  
 staff\_surnames = {row.id: row.surname for row in medical\_staff\_rows}  
  
 filtered\_results = []  
 for row in work\_activity\_rows:  
 operation\_price = operation\_prices.get(row.operation\_id, 0)  
 if operation\_price >= 7000 and row.quantity > 1:  
 staff\_surname = staff\_surnames.get(row.staff\_id, "Неизвестный сотрудник")  
 filtered\_results.append((row.date, row.operation\_id, staff\_surname))  
  
 print("\n7d. Даты, идентификаторы операций и фамилии тех, кто проводил операции стоимостью не менее 7000 руб больше одного раза:")  
 for date, operation\_id, surname in filtered\_results:  
 print(f"Дата: {date}, ID операции: {operation\_id}, Фамилия: {surname}")

**Результат запроса:**

****

**Уровень 2:**

**Задание 10:**

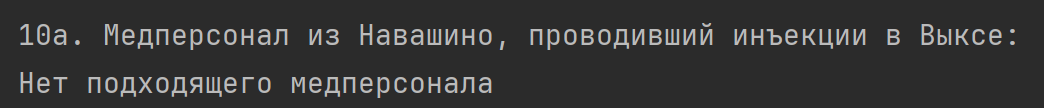
Используя операцию IN (NOT IN)  реализовать следующие запросы:

1. найти фамилии медперсонала из Навашино, проводивших инъекции в Выксе;

**Реализация:**

def task\_10a(session):  
 # 10a найти фамилии медперсонала из Навашино, проводивших инъекции в Выксе  
 print("\n10a. Медперсонал из Навашино, проводивший инъекции в Выксе:")  
 operation\_ids = set()  
 rows\_5 = session.execute("""  
 select id from operation\_types  
 where strong\_point = 'Выкса' and name in ('Инъекция поливитаминов', 'Инъекция алоэ')  
 allow filtering;  
 """)  
 for row in rows\_5:  
 operation\_ids.add(row.id)  
 staff\_ids = set()  
 for operation\_id in operation\_ids:  
 rows = session.execute("""  
 select staff\_id from work\_activity  
 where operation\_id = %s allow filtering;  
 """, (operation\_id,))  
 for row in rows:  
 staff\_ids.add(row.staff\_id)  
 if staff\_ids:  
 rows\_6 = session.execute("""  
 select surname from medical\_staff  
 where id in %s and address = 'Навашино' allow filtering  
 """, (tuple(staff\_ids),))  
 for row in rows\_6:  
 print(row.surname)  
 else:  
 print("Нет подходящего медперсонала")

**Результат запроса:**

****

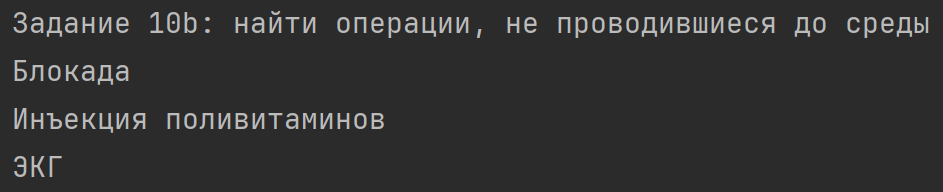
Проверили вручную: таких данных действительно нет.

1. найти те операции, которые не проводились до среды;

**Реализация:**

def task\_10b(session):  
 print("Задание 10b: найти операции, не проводившиеся до среды")  
 """Пояснение: не используется not in тк такая операция очень сильно влияет на производительность"""  
 rows = session.execute("""  
 select operation\_id from work\_activity where date in ('Среда', 'Четверг', 'Пятница', 'Суббота', 'Воскресенье')  
 allow filtering;  
 """)  
 operations\_after = {row\_after.operation\_id for row\_after in rows}  
 rows\_before = session.execute("""  
 select operation\_id from work\_activity   
 where date in ('Понедельник', 'Вторник')  
 allow filtering;  
 """)  
 operations\_before = {row.operation\_id for row in rows\_before} # Множество operation\_id  
  
 new\_operations = operations\_after - operations\_before  
 operation\_map = {}  
 rows\_operation = session.execute("select id, name from operation\_types")  
 for row\_operation in rows\_operation:  
 operation\_map[row\_operation.id] = (row\_operation.name)  
 for row\_result in new\_operations:  
 print(operation\_map.get(row\_result))

**Результат запроса:**

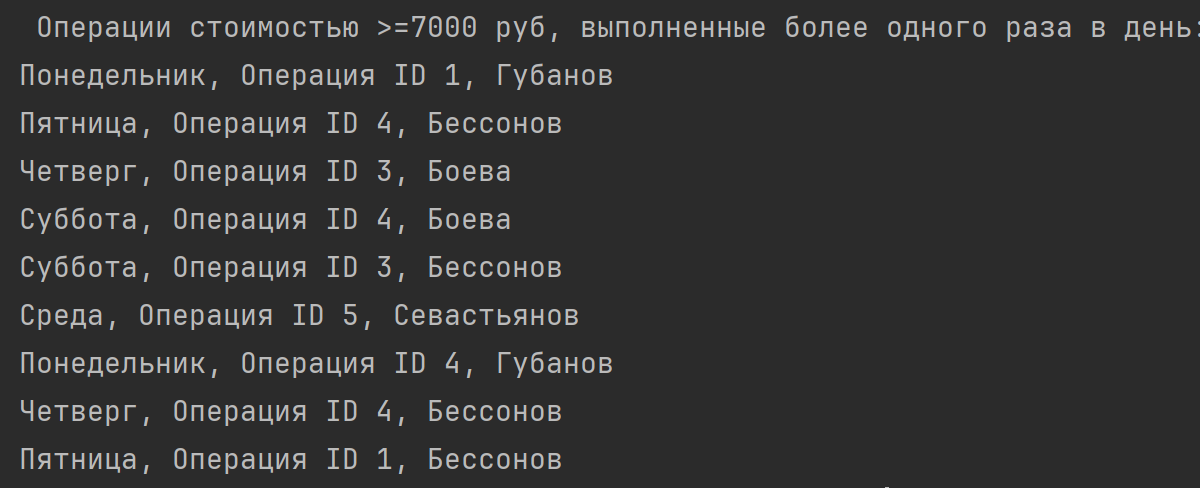
****

1. запросы задания 7.с и 7.d.

**Реализация:**

def task\_10c(session):  
 print("Ответ на задание 10с. \n Операции стоимостью >=7000 руб, выполненные более одного раза в день:")  
  
 high\_cost\_ops = session.execute("""  
 SELECT id FROM operation\_types WHERE price >= 7000 ALLOW FILTERING;  
 """)  
 high\_cost\_ids = {row.id for row in high\_cost\_ops}  
  
 if not high\_cost\_ids:  
 print("Нет операций со стоимостью >=7000 руб.")  
 return  
  
 work\_rows = session.execute("""  
 SELECT date, staff\_id, operation\_id, quantity FROM work\_activity;  
 """)  
  
 result\_rows = []  
 for row in work\_rows:  
 if row.operation\_id in high\_cost\_ids and row.quantity > 1:  
 result\_rows.append((row.date, row.staff\_id, row.operation\_id))  
  
 staff\_map = {}  
 staff\_rows = session.execute("SELECT id, surname FROM medical\_staff;")  
 for row in staff\_rows:  
 staff\_map[row.id] = row.surname  
  
 for date, staff\_id, operation\_id in result\_rows:  
 surname = staff\_map.get(staff\_id, "Неизвестно")  
 print(f"{date}, Операция ID {operation\_id}, {surname}")

**Результат запроса:**



**Задание 11:**

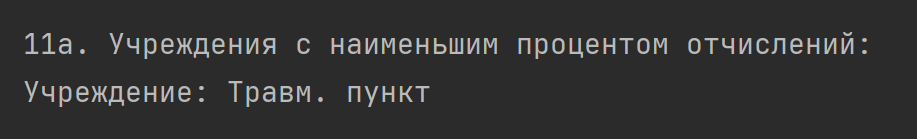
Используя операции ALL-ANY реализовать следующие запросы:

1. найти среди больниц ту, которая имеет наименьший процент отчислений;

**Реализация:**

def task\_11a(session):  
 # 11a. Найти учреждение с наименьшим процентом отчислений.  
  
 min\_tax\_row = session.execute("""  
 SELECT MIN(local\_tax\_percent) AS min\_tax  
 FROM workplaces;  
 """).one()  
  
 if not min\_tax\_row:  
 print("\n11a. Учреждения не найдены.")  
 return  
  
 min\_tax\_percent = min\_tax\_row.min\_tax  
  
 min\_tax\_institutions = session.execute(f"""  
 SELECT institution  
 FROM workplaces  
 WHERE local\_tax\_percent = {min\_tax\_percent}  
 ALLOW FILTERING;  
 """)  
  
 unique\_institutions = set(row.institution for row in min\_tax\_institutions)  
  
 if unique\_institutions:  
 print("\n11a. Учреждения с наименьшим процентом отчислений:")  
 for institution in unique\_institutions:  
 print(f"Учреждение: {institution}")  
 else:  
 print("\n11a. Учреждения не найдены.")

**Результат:**

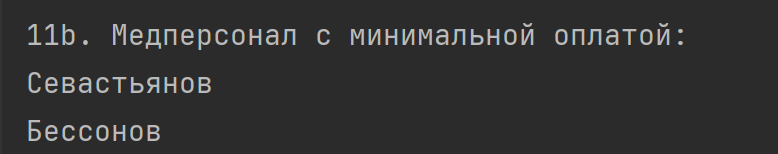
****

1. найти медперсонал, проводивший операции с самой малой суммой оплаты;

**Реализация:**

def task\_11b(session):  
 # 11b найти медперсонал, проводивший операции с самой малой суммой оплаты  
 min\_payment = session.execute("""  
 select min(payment) from work\_activity;  
 """).one()[0]  
  
 med\_ids = session.execute("""  
 select staff\_id from work\_activity where payment = %s allow filtering  
 """, (min\_payment,))  
 staff\_ids = [row.staff\_id for row in med\_ids]  
  
 surnames = session.execute("""  
 select surname from medical\_staff  
 where id in (%s)  
 """ % ','.join([str(id) for id in staff\_ids]))  
 print("\n11b. Медперсонал с минимальной оплатой:")  
 for surname in surnames:  
 print(surname.surname)

**Результат:**

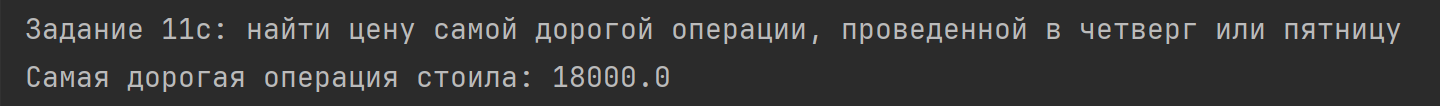
****

1. найти цену самой дорогой операции, проведенной в четверг или пятницу;

**Реализация:**

def task\_11c(session):  
 print("Задание 11c: найти цену самой дорогой операции, проведенной в четверг или пятницу")  
 rows = session.execute("select \* from work\_activity where date in ('Четверг', 'Пятница') allow filtering")  
 rows\_operation = session.execute("select \* from operation\_types")  
 operation\_map = {}  
 for row\_operation in rows\_operation:  
 operation\_map[row\_operation.id] = (row\_operation.price)  
 max\_operation = 0  
 for row in rows:  
 operation\_info = operation\_map.get(row.operation\_id)  
 if operation\_info > max\_operation:  
 max\_operation = operation\_info  
 print("Самая дорогая операция стоила:", max\_operation)

**Результат:**

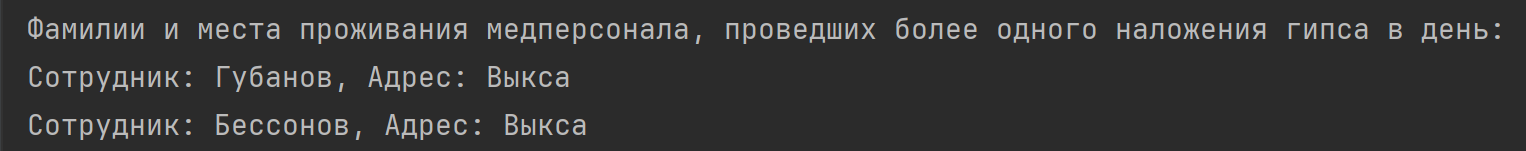


1. запрос задания 7.а.

**Реализация:**

def task\_11d(session):  
 try:  
 session.execute("CREATE INDEX IF NOT EXISTS idx\_work\_activity\_operation\_id ON work\_activity (operation\_id);")  
 session.execute("CREATE INDEX IF NOT EXISTS idx\_work\_activity\_quantity ON work\_activity (quantity);")  
 operation\_id = None  
 operations = session.execute("""  
 SELECT id FROM operation\_types WHERE name = 'Наложение гипса' ALLOW FILTERING;  
 """)  
  
 print("Ответ на задание 11d. \nФамилии и места проживания медперсонала, проведших более одного наложения гипса в день:")  
 for operation in operations:  
 operation\_id = operation.id  
 break  
  
 if operation\_id is not None:  
 rows\_3 = session.execute("""  
 SELECT staff\_id FROM work\_activity  
 WHERE operation\_id = %s AND quantity > 1 ALLOW FILTERING;  
 """, (operation\_id,))  
  
 staff\_ids = {row.staff\_id for row in rows\_3}  
 rows\_4 = session.execute("""  
 SELECT id, surname, address FROM medical\_staff;  
 """)  
 found = False  
 for row in rows\_4:  
 if row.id in staff\_ids:  
 print(f"Сотрудник: {row.surname}, Адрес: {row.address}")  
 found = True  
 if not found:  
 print("Нет данных.")  
 else:  
 print("Операция 'Наложение гипса' не найдена в таблице.")  
 finally:  
 session.execute("DROP INDEX IF EXISTS idx\_work\_activity\_operation\_id;")  
 session.execute("DROP INDEX IF EXISTS idx\_work\_activity\_quantity;")

**Результат:**

****

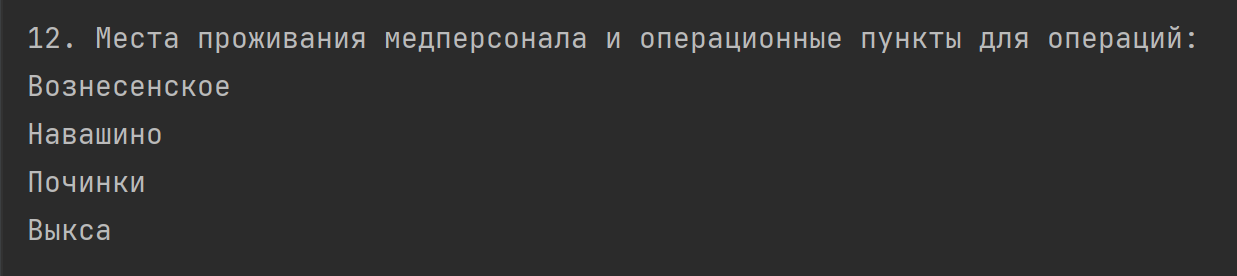
**Задание 12:**

Используя операцию UNION получить места проживания медпероснала и опероные пункты для операций

**Реализация:**

def task\_12(session):  
 # 12. Используя операцию UNION, получить места проживания медперсонала и операционные пункты для операций.  
  
 staff\_locations = session.execute("SELECT address FROM medical\_staff;")  
 operation\_locations = session.execute("SELECT strong\_point FROM operation\_types;")  
  
 unique\_locations = set(row.address for row in staff\_locations).union(  
 set(row.strong\_point for row in operation\_locations)  
 )  
  
 print("\n12. Места проживания медперсонала и операционные пункты для операций:")  
 for location in unique\_locations:  
 print(location)

**Результат:**

****

**Задание 13:**

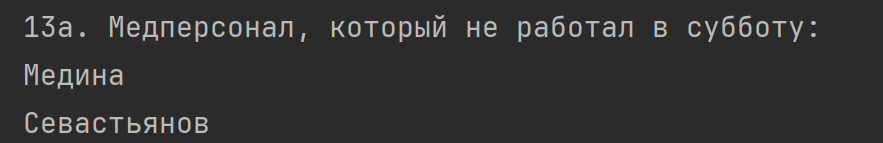
Используя операцию EXISTS ( NOT EXISTS ) реализовать нижеследующие запросы. В случае, если для текущего состояния БД запрос будет выдавать пустое множество строк, требуется указать какие добавления в БД необходимо провести.

1. определить тот медперсонал, который не работал в субботу;

**Реализация:**

def task\_13a(session):  
 # 13a медперсонал, который не работал в субботу  
 med\_ids\_2 = session.execute("""  
 select staff\_id from work\_activity  
 where date = 'Суббота' allow filtering;  
 """)  
 staff\_ids\_saturday = {row.staff\_id for row in med\_ids\_2}  
 surnames\_2 = session.execute("""  
 select surname, id from medical\_staff;  
 """)  
 print("\n13a. Медперсонал, который не работал в субботу:")  
 for surname in surnames\_2:  
 if surname.id not in staff\_ids\_saturday:  
 print(surname.surname)

**Результат:**

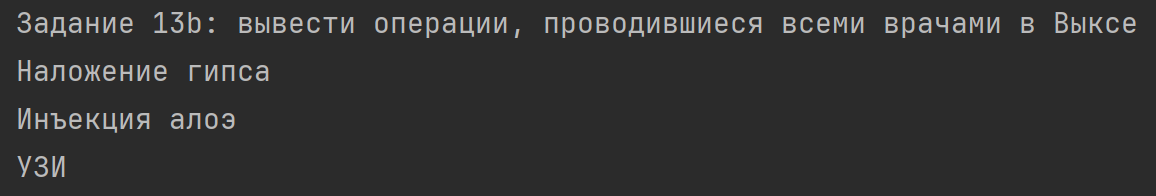
****

1. найти такие операции, которые проводились всеми врачами в Выксе;

**Реализация:**

def task\_13b(session):  
 print("Задание 13b: вывести операции, проводившиеся всеми врачами в Выксе")  
 rows = session.execute("select id from medical\_staff where address='Выкса' allow filtering")  
 medical\_staff = {row.id for row in rows}  
 if not medical\_staff:  
 print("Нет врачей в Выксе.")  
 else:  
 staff\_ids = ', '.join(map(str, medical\_staff))  
 query = f"select staff\_id, operation\_id from work\_activity where staff\_id in ({staff\_ids}) allow filtering"  
 rows\_activity = session.execute(query)  
 staff\_operations = {staff\_id: set() for staff\_id in medical\_staff}  
 for row in rows\_activity:  
 staff\_operations[row.staff\_id].add(row.operation\_id)  
 non\_empty\_operations = [ops for ops in staff\_operations.values() if ops]  
 if non\_empty\_operations:  
 common\_operations = set.intersection(\*non\_empty\_operations)  
 else:  
 common\_operations = set()  
 rows\_operations = session.execute("select id, name from operation\_types")  
 operation\_map = {row.id: row.name for row in rows\_operations}  
 if common\_operations:  
 for common in common\_operations:  
 print(operation\_map.get(common, f"Неизвестная операция (ID: {common})"))  
 else:  
 print("Нет операций, которые выполняли все врачи в Выксе.")

**Результат:**

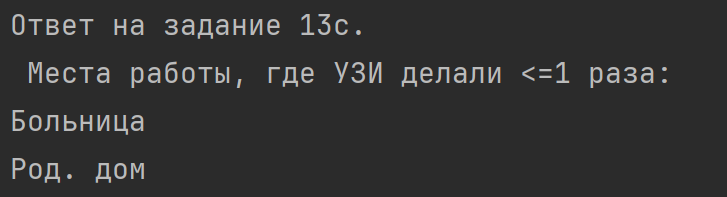
****

1. определить те места работы, где не делали УЗИ более раза;

**Реализация:**

def task\_13c(session):  
 rows = session.execute("""  
 SELECT workplace\_id FROM work\_activity  
 WHERE operation\_id = 6 ALLOW FILTERING;  
 """)  
 print("Ответ на задание 13с.\n Места работы, где УЗИ делали <=1 раза:")  
 from collections import Counter  
 wp\_ids = [row.workplace\_id for row in rows]  
 counter = Counter(wp\_ids)  
 limited\_wp\_ids = [wp\_id for wp\_id, count in counter.items() if count <= 1]  
 if not limited\_wp\_ids:  
 print("Нет таких мест работы. Нужно добавить данные.")  
 return  
 wp\_rows = session.execute(f"""  
 SELECT institution FROM workplaces  
 WHERE id IN ({', '.join(map(str, limited\_wp\_ids))});  
 """)  
 for row in wp\_rows:  
 print(row.institution)

**Результат:**

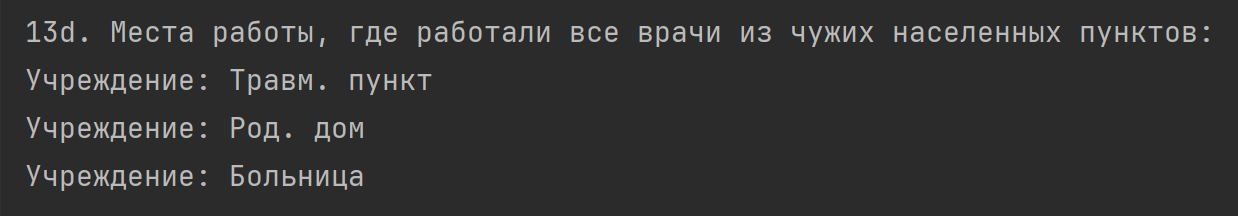
****

1. определить места работы, где работали все врачи из чужих населенных пунктов.

**Реализация:**

def task\_13d(session):  
 # 13d. Определить места работы, где работали все врачи из чужих населенных пунктов.  
 staff\_rows = session.execute("SELECT id, address FROM medical\_staff;")  
 staff\_data = {row.id: row.address for row in staff\_rows}  
  
 workplace\_rows = session.execute("SELECT id, institution, location FROM workplaces;")  
 workplace\_data = {row.id: (row.institution, row.location) for row in workplace\_rows}  
  
 activity\_rows = session.execute("SELECT staff\_id, workplace\_id FROM work\_activity;")  
  
 workplace\_info = {}  
 for row in activity\_rows:  
 staff\_id = row.staff\_id  
 workplace\_id = row.workplace\_id  
  
 staff\_address = staff\_data.get(staff\_id, "")  
 workplace\_location = workplace\_data.get(workplace\_id, ("", ""))[1]  
  
 is\_foreign = staff\_address != workplace\_location  
  
 if workplace\_id not in workplace\_info:  
 workplace\_info[workplace\_id] = {"institution": workplace\_data.get(workplace\_id, ("", ""))[0],  
 "all\_foreign": True}  
  
 if not is\_foreign:  
 workplace\_info[workplace\_id]["all\_foreign"] = False  
  
 foreign\_workplaces = [info for info in workplace\_info.values() if info["all\_foreign"]]  
  
 if foreign\_workplaces:  
 print("\n13d. Места работы, где работали все врачи из чужих населенных пунктов:")  
 for workplace in foreign\_workplaces:  
 print(f"Учреждение: {workplace['institution']}")  
 else:  
 print("\n13d. Нет мест работы, где работали все врачи из чужих населенных пунктов.")  
 print("Для выполнения запроса добавьте данные о врачах, которые работают только в чужих населенных пунктах.")

**Результат:**

****

**Задание 14:**

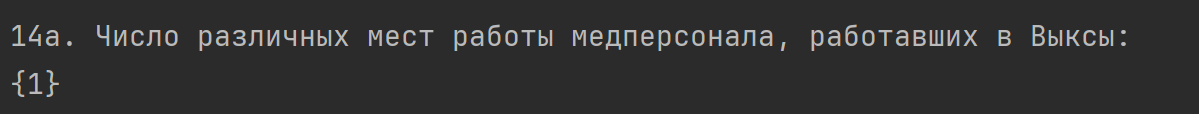
Реализовать запросы с использованием аггрегатных функций:

1. найти число различных мест работы для медперсонала, работавшего в мед.учреждениях Выксы;

**Реализация:**

def task\_14a(session):  
 # 14a найти число различных мест работы для медперсонала,  
 # работавшего в Выксы  
 addresses = session.execute("""  
 select id from workplaces  
 where location = 'Выкса' allow filtering;  
 """)  
 workplace\_ids = {row.id for row in addresses}  
 workplace\_ids\_2 = session.execute("""  
 select workplace\_id from work\_activity  
 where workplace\_id in (%s) allow filtering;  
 """ % ','.join([str(id) for id in workplace\_ids]))  
  
 unique\_workplaces = {row.workplace\_id for row in workplace\_ids\_2}  
 print("\n14а. Число различных мест работы медперсонала, работавших в Выксы:")  
 print({len(unique\_workplaces)})

**Результат:**

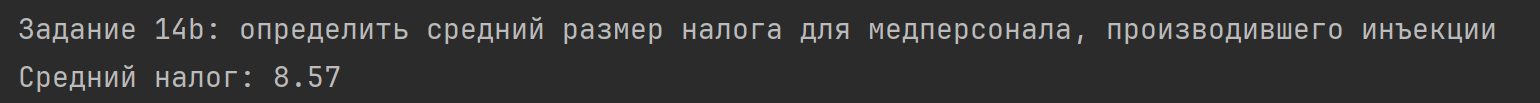
****

1. определить средний размер налога для медперсонала, производившего иньекции;

**Реализация:**

def task\_14b(session):  
 print("Задание 14b: определить средний размер налога для медперсонала, производившего инъекции")  
 operation\_rows = session.execute("select id, name from operation\_types")  
 operation\_map = {}  
 for operation\_row in operation\_rows:  
 operation\_map[operation\_row.id] = (operation\_row.name)  
 medical\_staff\_rows = session.execute("select id, tax\_percent from medical\_staff")  
 medical\_staff\_map = {}  
 for staff\_row in medical\_staff\_rows:  
 medical\_staff\_map[staff\_row.id] = (staff\_row.tax\_percent)  
 result = []  
 rows = session.execute("select staff\_id, operation\_id from work\_activity")  
 for row in rows:  
 staff\_info = medical\_staff\_map.get(row.staff\_id)  
 operation\_info = operation\_map.get(row.operation\_id)  
 if 'инъекция' in operation\_info.lower():  
 result.append((staff\_info))  
 if result:  
 print("Средний налог:", round(sum(result) / len(result), 2))  
 else:  
 print("Нет данных дляя вычисления среднего налога")

**Результат:**

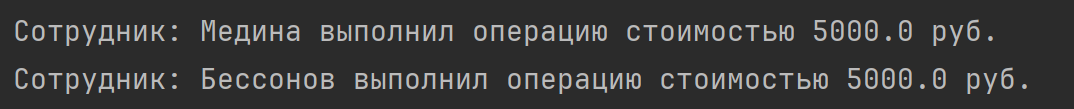
****

1. кто из медперсонала делал операцию с минимальной стоимостью;

**Реализация:**

def task\_14c(session):  
 print("Ответ на задание 14с.\n Медперсонал, сделавший операцию с минимальной стоимостью:")  
  
 operation\_rows = session.execute("SELECT id, name, price FROM operation\_types;")  
 operations = list(operation\_rows)  
  
 if not operations:  
 print("Нет данных об операциях.")  
 return  
  
 min\_price = min(op.price for op in operations)  
 min\_price\_operations = [op.id for op in operations if op.price == min\_price]  
  
 placeholders = ', '.join(str(op\_id) for op\_id in min\_price\_operations)  
 query = f"""  
 SELECT staff\_id FROM work\_activity   
 WHERE operation\_id IN ({placeholders}) ALLOW FILTERING;  
 """  
 rows = session.execute(query)  
 staff\_ids = set(row.staff\_id for row in rows)  
  
 if not staff\_ids:  
 print("Нет сотрудников, выполнивших операцию с минимальной стоимостью.")  
 return  
  
 for staff\_id in staff\_ids:  
 staff\_row = session.execute("SELECT surname FROM medical\_staff WHERE id=%s;", (staff\_id,))  
 surname\_row = staff\_row.one()  
 if surname\_row:  
 print(f"Сотрудник: {surname\_row.surname} выполнил операцию стоимостью {min\_price} руб.")

**Результат:**

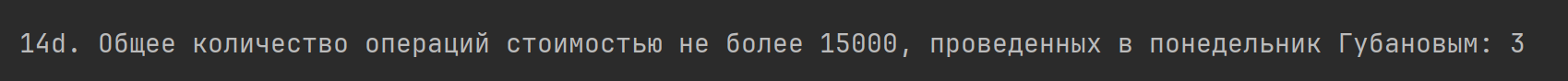
****

1. определить количество операций стоимостью не более 15000, проведенных в понедельник Губановым .

**Реализация:**

def task\_14d(session):  
 # 14d. Определить количество операций стоимостью не более 15000, проведенных в понедельник Губановым.  
  
 gubanov\_row = session.execute("""  
 SELECT id FROM medical\_staff WHERE surname = 'Губанов' ALLOW FILTERING;  
 """).one()  
  
 if not gubanov\_row:  
 print("\n14d. Сотрудник с фамилией 'Губанов' не найден.")  
 return  
  
 gubanov\_id = gubanov\_row.id  
 operations = session.execute(f"""  
 SELECT operation\_id, quantity FROM work\_activity  
 WHERE staff\_id = {gubanov\_id} AND date = 'Понедельник'  
 ALLOW FILTERING;  
 """)  
  
 operation\_prices = {}  
 operation\_rows = session.execute("SELECT id, price FROM operation\_types;")  
 for row in operation\_rows:  
 operation\_prices[row.id] = row.price  
  
 total\_quantity = 0  
 for row in operations:  
 operation\_id = row.operation\_id  
 quantity = row.quantity  
 operation\_price = operation\_prices.get(operation\_id, 0)  
  
 if operation\_price <= 15000:  
 total\_quantity += quantity  
  
 print(  
 f"\n14d. Общее количество операций стоимостью не более 15000, проведенных в понедельник Губановым: {total\_quantity}")

**Результат:**

****

**Задание 15:**

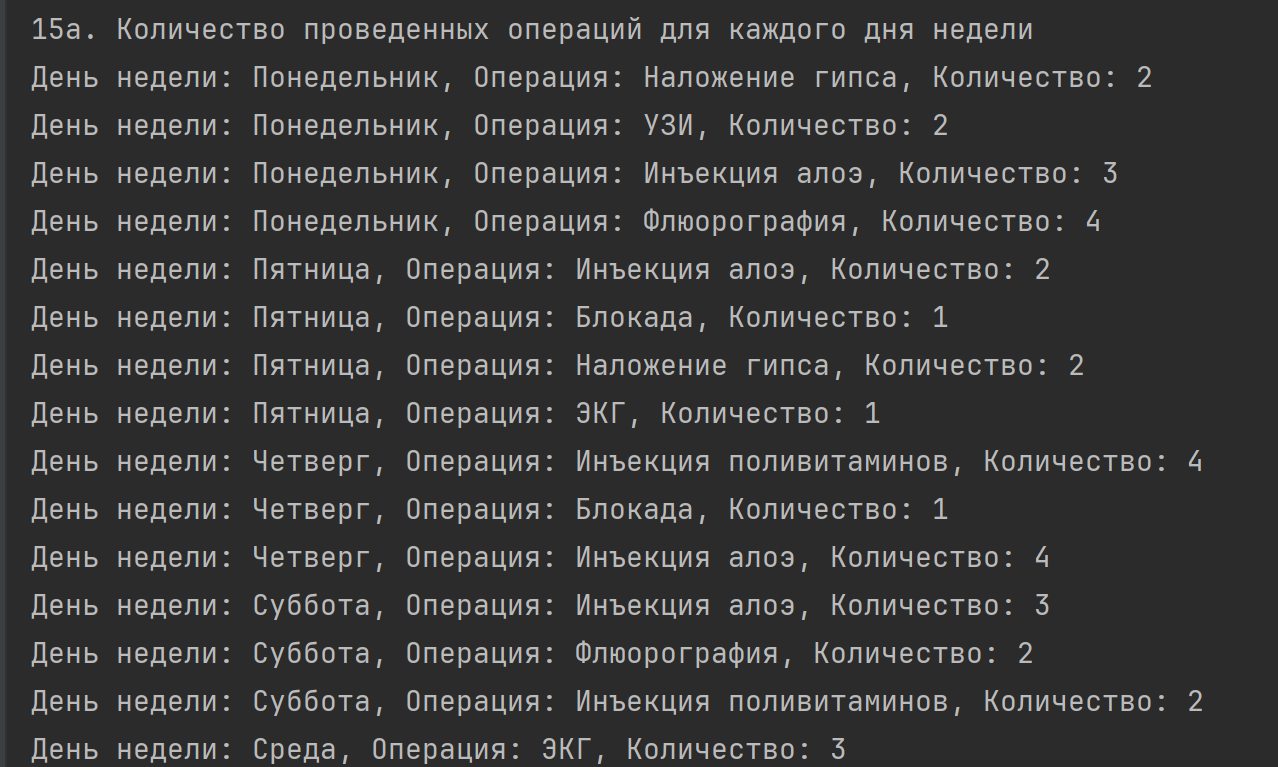
Используя средства группировки реализовать следующие запросы:

1. определить для каждого дня недели и каждой операции сколько раз ее проводили;

**Реализация:**

def task\_15a(session):  
 # 15a определить для каждого дня недели и каждой операции  
 # сколько раз ее проводили  
 print("\n15a. Количество проведенных операций для каждого дня недели")  
 operation\_counts = session.execute("""  
 select date, operation\_id, quantity from work\_activity;  
 """)  
 operation\_count\_by\_day = {}  
 for row in operation\_counts:  
 date = row.date  
 operation\_id = row.operation\_id  
 count = row.quantity  
 if date not in operation\_count\_by\_day:  
 operation\_count\_by\_day[date] = {}  
 if operation\_id not in operation\_count\_by\_day[date]:  
 operation\_count\_by\_day[date][operation\_id] = 0  
 operation\_count\_by\_day[date][operation\_id] += count  
  
 operation\_names = session.execute("""  
 select id, name from operation\_types;  
 """)  
 operation\_name\_map = {row.id: row.name for row in operation\_names}  
  
 for date, operations in operation\_count\_by\_day.items():  
 for operation\_id, count in operations.items():  
 operation\_name = operation\_name\_map.get(operation\_id, "Unknown operation")  
 print(f"День недели: {date}, Операция: {operation\_name}, Количество: {count}")

**Результат:**

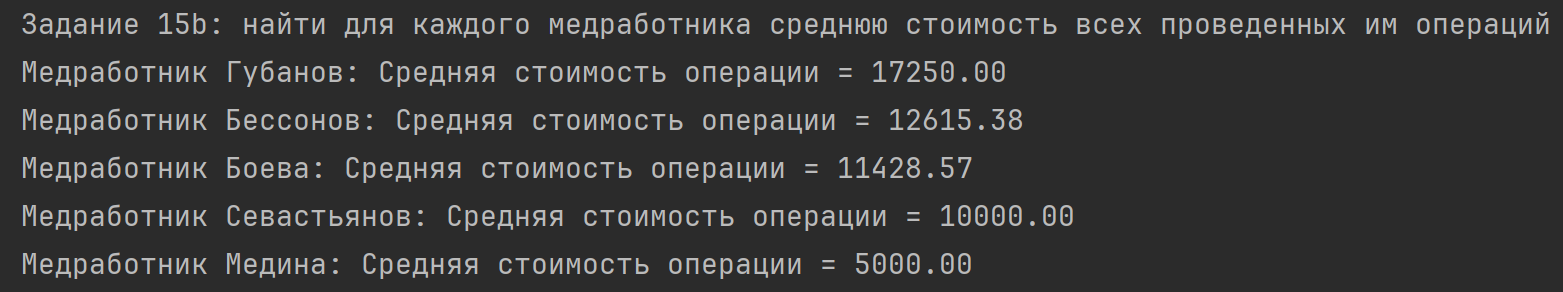
****

1. найти для каждого медработника среднюю стоимость всех проведенных им операций;

**Реализация:**

def task\_15b(session):  
 print("Задание 15b: найти для каждого медработника среднюю стоимость всех проведенных им операций")  
 medical\_staff\_rows = session.execute("select id, surname from medical\_staff")  
 medical\_staff\_map = {}  
 for staff\_row in medical\_staff\_rows:  
 medical\_staff\_map[staff\_row.id] = (staff\_row.surname)  
 result = {}  
 rows = session.execute("select staff\_id, payment, quantity from work\_activity")  
 for row in rows:  
 staff\_info = medical\_staff\_map.get(row.staff\_id)  
 if staff\_info in result:  
 total\_payment, count = result[staff\_info]  
 result[staff\_info] = (total\_payment + row.payment, row.quantity + count)  
 else:  
 result[staff\_info] = (row.payment, row.quantity)  
 for staff\_name, (total\_payment, count) in result.items():  
 avg\_payment = total\_payment / count  
 print(f"Медработник {staff\_name}: Средняя стоимость операции = {avg\_payment:.2f}")

**Результат:**

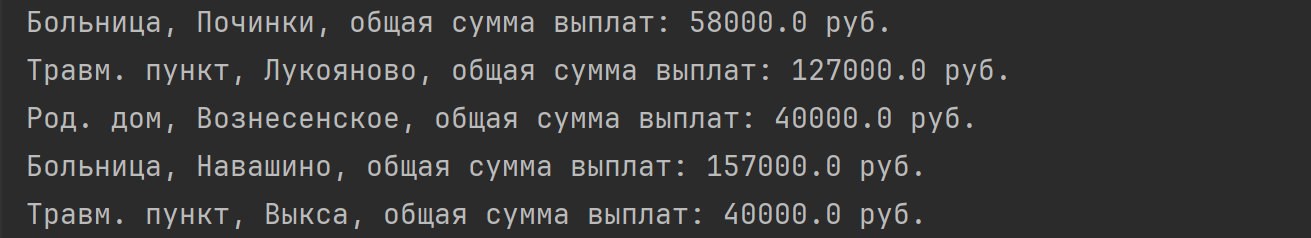
****

1. определить те мед.учреждения, где суммарная величина стоимости всех проведенных в них операций была более 30000;

**Реализация:**

def task\_15c(session):  
 print("Ответ на задание 15с.\n Медучреждения с суммой операций >30000 руб.")  
  
 work\_rows = session.execute("SELECT workplace\_id, payment FROM work\_activity;")  
  
 payments\_by\_workplace = {}  
 for row in work\_rows:  
 payments\_by\_workplace.setdefault(row.workplace\_id, Decimal(0))  
 payments\_by\_workplace[row.workplace\_id] += row.payment  
  
 high\_payment\_workplaces = [wid for wid, total in payments\_by\_workplace.items() if total > 30000]  
  
 if not high\_payment\_workplaces:  
 print("Нет мест работы с общей суммой выплат более 30000 руб.")  
 return  
  
 for workplace\_id in high\_payment\_workplaces:  
 workplace\_row = session.execute("SELECT institution, location FROM workplaces WHERE id=%s;", (workplace\_id,))  
 result = workplace\_row.one()  
 if result:  
 total\_payment = payments\_by\_workplace[workplace\_id]  
 print(f"{result.institution}, {result.location}, общая сумма выплат: {total\_payment} руб.")

**Результат:**

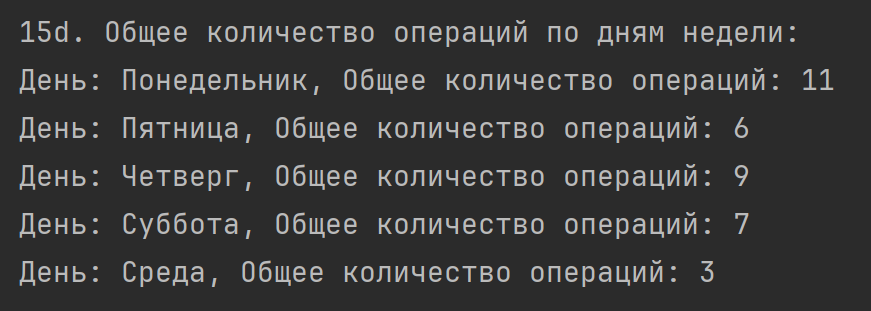


1. для каждого дня недели найти число проведенных в этот день операций.

**Реализация:**

def task\_15d(session):  
 # 15d. Для каждого дня недели найти общее количество проведенных операций.  
  
 rows = session.execute("""  
 SELECT date, quantity FROM work\_activity;  
 """)  
  
 operations\_by\_day = {}  
  
 for row in rows:  
 day = row.date  
 quantity = row.quantity  
  
 if day in operations\_by\_day:  
 operations\_by\_day[day] += quantity  
 else:  
 operations\_by\_day[day] = quantity  
  
 print("\n15d. Общее количество операций по дням недели:")  
 for day, total\_quantity in operations\_by\_day.items():  
 print(f"День: {day}, Общее количество операций: {total\_quantity}")

**Результат:**

****