Постановка задачи

Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс ≪Сотрудник≫, содержащий поля:
   - ID записи о сотруднике;
   - Фамилия сотрудника;
   - Зарплата (количественный признак);
   - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс ≪Отдел≫, содержащий поля:
   - ID записи об отделе;
   - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс ≪Сотрудники отдела≫, содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов ≪Сотрудник≫ и ≪Отдел≫, которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи ≪много≫, произвольный количественный признак, например, ≪зарплата сотрудника≫.

17	Дирижер	Оркестр

```python
from typing import Self, Any, Callable


class Conductor:
    def __init__(self, conductorID, name, work_experience, orchestraID) -> Self:
        self._conductorID = conductorID
        self._name = name
        self._work_experience = work_experience
        self._orchestraID = orchestraID

    def __str__(self) -> str:
        return f"Conductor: {self._name} from orchestra\n"

    @property
    def get_conductorID(self): return self._conductorID

    @property
    def get_name(self): return self._name

    @property
    def get_work_experience(self): return self._work_experience


class Orchestra:
    def __init__(self, orchestraID, name, date):
        self.ID = orchestraID
        self._name = name
        self._establishment = date

    def __str__(self):
        return f"Orchestra: {self._name}\n"

    @property
    def get_name(self): return self._name

    @property
    def get_establishment(self): return self._establishment


class Ensemble:
    def __init__(self, conductorID, orchestraID):
        self._conductor_ID = conductorID
        self._orchestra_ID = orchestraID


orchestras = [
    Orchestra(1, "Gewandhausorchester Leipzig", 1781),
    Orchestra(2, "New York Philharmonic", 1842),
    Orchestra(3, "Munich Philharmonic", 1893),
    Orchestra(4, "Boston Symphony Orchestra", 1881),
    Orchestra(5, "London Symphony Orchestra", 1904)
]
```

```python

conductors = [
    Conductor(1, "Mario", 11, 1),
    Conductor(2, "Zuck", 7, 2),
    Conductor(3, "Enzo", 17, 3),
    Conductor(4, "Edward", 9, 4),
    Conductor(5, "Joshua", 8, 5)
]

ensembles = [
    Ensemble(1, 4),
    Ensemble(2, 1),
    Ensemble(3, 4),
    Ensemble(5, 2),
    Ensemble(2, 2),
    Ensemble(1, 3),
    Ensemble(1, 5),
    Ensemble(4, 4),
    Ensemble(5, 4)
]

one_to_many = [(conductor._name, conductor._work_experience, orchestra._name, orchestra._establishment)
               for conductor in conductors
               for orchestra in orchestras
               if conductor._orchestraID == orchestra.ID]

many_to_many_temp = [(orchestra._name, ensemble._conductor_ID, ensemble._orchestra_ID)
                     for orchestra in orchestras
                     for ensemble in ensembles
                     if orchestra.ID == ensemble._orchestra_ID]

many_to_many = [(conductor._name, conductor._work_experience, orchestra_name)
                for orchestra_name, _, conductorID in many_to_many_temp
                for conductor in conductors if conductor._conductorID == conductorID]

#Task1: Выведите список всех всязанных дирижеров и оркестров, отсортированных по дирижерам:

result1 = [(conductor_name, orchestra_name)
           for conductor_name, _, orchestra_name, _ in one_to_many]

result1 = sorted(result1, key=lambda x: x[0])

for i in result1:
    print(i)

```

```
 ~/Developer/Python Projects/3SEM_LAB  master ?4
 python3 RK1/main.py
('Edward', 'Boston Symphony Orchestra')
('Enzo', 'Munich Philharmonic')
('Joshua', 'London Symphony Orchestra')
('Mario', 'Gewandhausorchester Leipzig')
('Zuck', 'New York Philharmonic')

('New York Philharmonic', 'Zuck', 7)
('London Symphony Orchestra', 'Joshua', 8)
('Boston Symphony Orchestra', 'Edward', 9)

('Gewandhausorchester Leipzig', 'Mario')
('London Symphony Orchestra', 'Joshua')
('Boston Symphony Orchestra', 'Edward')
('New York Philharmonic', 'Zuck')
('Munich Philharmonic', 'Enzo')
```