Вариант запросов Д. Предметная область 5

1. **Запросы по варианту В**:
    - **Запрос 1**: Вывод музыкантов, у которых фамилия начинается с "А", и названий их оркестров (используется связь один-ко-многим)
    - **Запрос 2**: Вывод оркестров с минимальной зарплатой музыкантов, отсортированный по минимальной зарплате
    - **Запрос 3**: Вывод всех связанных музыкантов и оркестров, отсортированный по музыкантам (используется связь многие-ко-многим)

Код программы.

```
from operator import itemgetter


class Musician:
    """Музыкант"""
    def __init__(self, id, name, salary, orchestra_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.orchestra_id = orchestra_id


class Orchestra:
    """Оркестр"""
    def __init__(self, id, name):
        self.id = id
        self.name = name


class MusicianOrchestra:
```

```
    """'Музыканты в оркестре' для реализации связи многие-ко-многим"""
    def __init__(self, orchestra_id, musician_id):
        self.orchestra_id = orchestra_id
        self.musician_id = musician_id


orchestras = [
    Orchestra(1, 'Симфонический оркестр'),
    Orchestra(2, 'Камерный оркестр'),
    Orchestra(3, 'Джазовый оркестр'),
    Orchestra(11, 'Симфонический оркестр филармонии'),
    Orchestra(22, 'Камерный оркестр театра'),
    Orchestra(33, 'Джазовый оркестр клуба'),
]


musicians = [
    Musician(1, 'Андреев', 50000, 1),
    Musician(2, 'Петров', 45000, 2),
    Musician(3, 'Сидоров', 60000, 3),
    Musician(4, 'Алексеев', 55000, 3),
    Musician(5, 'Борисов', 48000, 3),
]
musicians_orchestras = [
    MusicianOrchestra(1, 1),
    MusicianOrchestra(2, 2),
    MusicianOrchestra(3, 3),
    MusicianOrchestra(3, 4),
```

```
    MusicianOrchestra(3, 5),
    MusicianOrchestra(11, 1),
    MusicianOrchestra(22, 2),
    MusicianOrchestra(33, 3),
    MusicianOrchestra(33, 4),
    MusicianOrchestra(33, 5),
]

def main():

    one_to_many = [(m.name, m.salary, o.name)
                   for o in orchestras
                   for m in musicians
                   if m.orchestra_id == o.id]

    many_to_many_temp = [(o.name, mo.orchestra_id, mo.musician_id)
                         for o in orchestras
                         for mo in musicians_orchestras
                         if o.id == mo.orchestra_id]

    many_to_many = [(m.name, m.salary, orchestra_name)
                    for orchestra_name, orchestra_id, musician_id in
many_to_many_temp
                    for m in musicians if m.id == musician_id]

    print('Задание В1 (для предметной области Музыкант-Оркестр)')
```

```
    print('Список всех музыкантов, у которых фамилия начинается с буквы "A", и названия их оркестров:')

    res1 = list(filter(lambda x: x[0].startswith('A'), one_to_many))

    print(res1)



    print('\nЗадание В2 (для предметной области Музыкант-Оркестр)')

    print('Список оркестров с минимальной зарплатой музыкантов в каждом оркестре, отсортированный по минимальной зарплате:')



    res2_unsorted = []

    for o in orchestras:

        # Список музыкантов оркестра

        o_musicians = list(filter(lambda i: i[2] == o.name, one_to_many))

        # Если оркестр не пустой

        if len(o_musicians) > 0:

            # Зарплаты музыкантов оркестра

            o_salaries = [salary for _, salary, _ in o_musicians]

            # Минимальная зарплата в оркестре

            min_salary = min(o_salaries)

            res2_unsorted.append((o.name, min_salary))



    res2 = sorted(res2_unsorted, key=itemgetter(1))

    print(res2)



    print('\nЗадание В3 (для предметной области Музыкант-Оркестр)')

    print('Список всех связанных музыкантов и оркестров, отсортированный по музыкантам, сортировка по оркестрам произвольная:')
```

```
    # Сортируем по имени музыканта (первый элемент кортежа)
    res3 = sorted(many_to_many, key=itemgetter(0))
    print(res3)


if __name__ == '__main__':
    main()
```