Текст программы (решение варианта Б для рассмотренной предметной области варианта 1)

```
# используется для сортировки
from operator import itemgetter


class student:
    """Студент"""

    def __init__(self, id, fio, group_id):
        self.id = id
        self.fio = fio
        self.group_id = group_id


class Group:
    """Группа"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class StudGr:
    """ 'Студенты группы' для реализации
    связи многие-ко-многим"""

    def __init__(self, group_id, student_id):
        self.group_id = group_id
        self.student_id = student_id


# Группы
grou = [
    Group(1, 'G1'),
    Group(2, 'G2'),
    Group(3, 'G3'),
    Group(4, 'G4'),

]

# Студенты
Stud = [
    student(1, 'Петров', 1),
    student(2, 'Смирнов', 1),
    student(3, 'Краев', 2),
    student(4, 'Чернов', 4),
    student(5, 'Айвазовский', 3),
    student(6, 'Брюллов', 3),
    student(7, 'Ленин', 1),
    student(8, 'Васнецов', 2),
    student(9, 'Бауман', 4),
    student(10, 'Королев', 2),
    student(11, 'Бунин', 1),
]

stud_gr = [
    StudGr(1, 1),
    StudGr(1, 2),
    StudGr(1, 7),
    StudGr(1, 11),
    StudGr(2, 3),
```

```
    StudGr(2, 8),
    StudGr(2, 10),
    StudGr(3, 5),
    StudGr(3, 6),
    StudGr(4, 4),
    StudGr(4, 9),

    StudGr(11, 1),
    StudGr(11, 2),
    StudGr(11, 7),
    StudGr(11, 11),
    StudGr(22, 3),
    StudGr(22, 8),
    StudGr(22, 10),
    StudGr(33, 5),
    StudGr(33, 6),
    StudGr(44, 4),
    StudGr(44, 9),
]


def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(s.fio, g.name)
                   for g in grou
                   for s in Stud
                   if s.group_id == g.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(g.name, sg.group_id, sg.student_id)
                         for g in grou
                         for sg in stud_gr
                         if g.id == sg.group_id]

    many_to_many = [(s.fio, group_name)
                    for group_name, group_id, student_id in many_to_many_temp
                    for s in Stud if s.id == student_id]

    print('Задание Б1 \n')
    res_11 = sorted(one_to_many, key=itemgetter(1))
    print(*res_11, sep='\n')

    print('\nЗадание Б2')
    res_12_unsorted = []
    for g in grou:
        # Список сотрудников отдела
        g_stud = list(filter(lambda i: i[1] == g.name, one_to_many))
        if len(g_stud) > 0:
            # Суммарная зарплата сотрудников отдела
            g_sum = len(g_stud)
            res_12_unsorted.append((g.name, g_sum))

        # Сортировка по суммарной зарплате
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(*res_12, sep='\n')

    print('\nЗадание Б3')
    res_13 = {}
    # Перебираем все отделы
    for g in grou:
        # Список сотрудников отдела
        g_stud = list(filter(lambda i: i[1] == g.name, many_to_many))
        # Только ФИО сотрудников
```

```
        g_stud_names = [x for x, _ in g_stud]
        # Добавляем результат в словарь
        sss = list(filter(lambda i: i[-1] == 'в' and i[-2] == 'о',
g_stud_names))
        res_13[g.name] = sss
    print(res_13)


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

Результаты выполнения:

Задание Б1

('Петров', 'G1')

('Смирнов', 'G1')

('Ленин', 'G1')

('Бунин', 'G1')

('Краев', 'G2')

('Васнецов', 'G2')

('Королев', 'G2')

('Айвазовский', 'G3')

('Брюллов', 'G3')

('Чернов', 'G4')

('Бауман', 'G4')

Задание Б2

('G1', 4)

('G2', 3)

('G3', 2)

('G4', 2)

Задание Б3

{'G1': ['Петров', 'Смирнов'], 'G2': ['Васнецов'], 'G3': ['Брюллов'], 'G4': ['Чернов']}