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

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).

**Код программы.**

from operator import itemgetter  
  
class Table():  
 def \_\_init\_\_(self, id, name, base\_id, kol):  
 self.id = id  
 self.name = name  
 self.base\_id = base\_id  
 self.kol = kol  
  
class Base():  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
class TableBase():  
 def \_\_init\_\_(self, base\_id, table\_id):  
 self.base\_id = base\_id  
 self.table\_id = table\_id  
  
  
base: list[Base] = [  
 Base(1, "Учебное заведение"),  
 Base(2, "Магазин"),  
 Base(3, "Социальная сеть"),  
 Base(4, "Проект управления")  
]  
table: list[Table] = [  
 Table(1, "Студенты", 1, 5),  
 Table(2, "Курсы", 3, 4),  
 Table(3, "Преподаватели", 2, 6),  
 Table(4, "Товары", 2, 7),  
 Table(5, "Заказы", 1, 5),  
 Table(6, "Клиенты", 4, 3),  
 Table(7, "Пользователи", 1, 8),  
 Table(8, "Сообщения", 1, 4),  
 Table(9, "Друзья", 3, 2),  
 Table(10, "Проекты", 1, 6),  
 Table(11, "Задачи", 2, 5),  
 Table(12, "Сотрудники", 3, 4)  
]  
table\_b = []  
table\_base = [  
 TableBase(1, 3),  
 TableBase(1, 9),  
 TableBase(1, 5),  
 TableBase(1, 11),  
 TableBase(1, 7),  
 TableBase(2, 2),  
 TableBase(2, 8),  
 TableBase(2, 4),  
 TableBase(2, 6),  
 TableBase(2, 10),  
 TableBase(2, 12),  
 TableBase(3, 10),  
 TableBase(3, 1),  
 TableBase(3, 5),  
 TableBase(3, 7),  
 TableBase(3, 3),  
 TableBase(4, 12),  
 TableBase(4, 6),  
 TableBase(4, 9),  
 TableBase(4, 3)  
]  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(t.name, t.kol, b.name)  
 for b in base  
 for t in table  
 if t.base\_id == b.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(b.name, tb.table\_id)  
 for b in base  
 for tb in table\_base if tb.base\_id == b.id]  
  
 many\_to\_many = [(base\_name, t.name, t.kol)  
 for base\_name, table\_id in many\_to\_many\_temp  
 for t in table if t.id == table\_id]  
  
 print('Задание B1')  
 searched\_char = 'З'  
 res\_11 = [el for el in one\_to\_many if el[0][0] == searched\_char]  
 print(\*res\_11, sep='\n')  
  
 print('\nЗадание B2')  
 res\_12\_unsorted = [[el.name, 1000000] for el in base]  
 for t in res\_12\_unsorted:  
 t[1] = min(map(lambda i: i[1], list(filter(lambda i: i[2] == t[0], one\_to\_many))))  
  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 print(\*res\_12, sep='\n')  
  
 print('\nЗадание B3')  
 res\_13 = sorted(many\_to\_many, key=itemgetter(1), reverse=True)  
  
 print(\*res\_13, sep='\n')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Вывод программы.**

