РК1

Вариант Д.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Класс 1	Класс 2
Компьютер	Дисплейный класс

Вариант Д.

1. «Дисплейный класс» и «Компьютер» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название заканчивается на «n», и названия их классов.
2. «Дисплейный класс» и «Компьютер» связаны соотношением один-ко-многим. Выведите список классов со средней стоимостью компьютеров в каждом классе, отсортированный по средней стоимости (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Дисплейный класс» и «Компьютер» связаны соотношением многие-ко-многим. Выведите список всех классов, у которых название начинается с буквы «N», и список компьютеров в них.

```
class Computer:
    def __init__(self, id: int, name: str, cost: int,
display_class_id: int = None):
        self.id = id
        self.name = name
        self.threads = cost
        self.display_class_id = display_class_id


class DisplayClass:
    def __init__(self, id: int, name: str):
        self.id = id
        self.name = name


class ComputerAndDisplayClass:
    def __init__(self, display_class_id: int, computer_id:
int):
        self.display_class_id = display_class_id
        self.computer_id = computer_id


display_classes = [
    DisplayClass(id=1, name="Quantum"),
    DisplayClass(id=2, name="Cyber"),
    DisplayClass(id=3, name="Nano"),
    DisplayClass(id=4, name="Titan"),
    DisplayClass(id=5, name="Nexus")
]


computers = [
    Computer(id=1, name="Alpha", cost=50000),
```

```
    Computer(id=2, name="Beta", cost=20000),
    Computer(id=3, name="Gamma", cost=70000),
    Computer(id=4, name="Delta", cost=90000),
    Computer(id=5, name="Epsilon", cost=30000)
]

display_classes_to_computers = [
    ComputerAndDisplayClass(display_class_id=1,
computer_id=1),
    ComputerAndDisplayClass(display_class_id=1,
computer_id=3),
    ComputerAndDisplayClass(display_class_id=2,
computer_id=2),
    ComputerAndDisplayClass(display_class_id=3,
computer_id=3),
    ComputerAndDisplayClass(display_class_id=4,
computer_id=4),
    ComputerAndDisplayClass(display_class_id=5,
computer_id=1),
    ComputerAndDisplayClass(display_class_id=5,
computer_id=5),
    ComputerAndDisplayClass(display_class_id=2,
computer_id=4)
]

def list_computers_with_name_ending_ov(computers,
display_classes, links):
    for link in links:
        computer = next(c for c in computers if c.id ==
link.computer_id)
        if computer.name.endswith("n"):
```

```
            display_class = next(d for d in display_classes
if d.id == link.display_class_id)
            print(f"Computer: {computer.name}, Display
Class: {display_class.name}")


def list_average_threads_per_display(display_classes,
computers, links):
    avg_threads_per_class = []
    for display_class in display_classes:
        related_computers = [c for c in computers if
any(l.display_class_id == display_class.id and l.computer_id
== c.id for l in links)]
        if related_computers:
            total_threads = sum(c.threads for c in
related_computers)
            avg_threads = total_threads /
len(related_computers)
            avg_threads_per_class.append((display_class.name
, avg_threads))
    avg_threads_per_class.sort(key=lambda x: x[1])
    for display_name, avg in avg_threads_per_class:
        print(f"Display Class: {display_name}, Average Cost:
{avg:.2f}")


def list_display_classes_starting_with_a(display_classes,
computers, links):
    for display_class in display_classes:
        if display_class.name.startswith("N"):
            related_computers = [c for c in computers if
any(l.display_class_id == display_class.id and l.computer_id
== c.id for l in links)]
```

```
            computer_names = [c.name for c in
related_computers]
            print(f"Display Class: {display_class.name},
Computers: {', '.join(computer_names)}")


def main():
    list_computers_with_name_ending_ov(computers,
display_classes, display_classes_to_computers)


    list_average_threads_per_display(display_classes,
computers, display_classes_to_computers)


    list_display_classes_starting_with_a(display_classes,
computers, display_classes_to_computers)


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

Задание №1

```
Computer: Epsilon, Display Class: Nexus
```

Задание №2

```
Display Class: Nexus, Average Cost: 40000.00
Display Class: Cyber, Average Cost: 55000.00
Display Class: Quantum, Average Cost: 60000.00
Display Class: Nano, Average Cost: 70000.00
Display Class: Titan, Average Cost: 90000.00
```

Задание №3

```
Display Class: Nano, Computers: Gamma
Display Class: Nexus, Computers: Alpha, Epsilon
```