Файл RK2.py:

```
from collections import Counter, defaultdict

class CarPark:
    def __init__(self, carpark_id, name):
        self.carpark_id = carpark_id
        self.name = name

class Driver:
    def __init__(self, driver_id, surname, salary, carpark_id):
        self.driver_id = driver_id
        self.surname = surname
        self.salary = salary
        self.carpark_id = carpark_id

class DriverAndCarPark:
    def __init__(self, driver_id, carpark_id):
        self.driver_id = driver_id
        self.carpark_id = carpark_id

def get_sorted_drivers(drivers_data):
    return sorted(drivers_data, key=lambda x: x.surname)

def get_carpark_dict(carparks_data):
    return {carpark.carpark_id: carpark.name for carpark in carparks_data}

def query_1(drivers_data, carparks_data, driver_and_carpark_data):
    sorted_drivers = get_sorted_drivers(drivers_data)
    carpark_dict = get_carpark_dict(carparks_data)
    result = []
    for driver in sorted_drivers:
        carpark_name = carpark_dict.get(driver.carpark_id, 'Неизвестный автопарк')
        result.append(f"Водитель: {driver.surname}, Автопарк: {carpark_name}")
    return result

def query_2(drivers_data, carparks_data):
    num_drivers = Counter(d.carpark_id for d in drivers_data)
    sorted_carparks = sorted(carparks_data, key=lambda x: x.name)
    sorted_carparks = sorted(sorted_carparks, key=lambda x: num_drivers[x.carpark_id],
reverse=True)
    result = []
    for carpark in sorted_carparks:
        result.append(f"Автопарк: {carpark.name}, Количество водителей:
{num_drivers[carpark.carpark_id]}")
    return result

def query_3(drivers_data, carparks_data, driver_and_carpark_data):
    driver_carpark_dict = defaultdict(list)
    carpark_dict = get_carpark_dict(carparks_data)
    for cd in driver_and_carpark_data:
        driver_carpark_dict[cd.driver_id].append(cd.carpark_id)
    filtered_drivers = [driver for driver in drivers_data if driver.surname.endswith("ов")]
    result = []
    for driver in filtered_drivers:
        carparks = [carpark_dict[carpark_id] for carpark_id in
driver_carpark_dict.get(driver.driver_id, [])]
        result.append(f"Водитель: {driver.surname}, Автопарк(и): {', '.join(carparks)}")
    return result
```

Файл tests.py:

```
import unittest
from RK2 import Driver, CarPark, DriverAndCarPark, query_1, query_2, query_3

drivers_data = [
            Driver(1, "Иванов", 20000, 4),
            Driver(2, "Петров", 15000, 3),
            Driver(3, "Сидоров", 17000, 5),
```

```
            Driver(4, "Михалёв", 21000, 1),
            Driver(5, "Радченко", 23000, 2),
            Driver(6, "Сидоренко", 16000, 3),
            Driver(7, "Абрамов", 20000, 3),
            Driver(8, "Тюльпанов", 18000, 5)
        ]
carparks_data = [
            CarPark(1, "ЭкоМобиль"),
            CarPark(2, "АвтоОазис"),
            CarPark(3, "Городской Флот"),
            CarPark(4, "Звездный"),
            CarPark(5, "АвтоСпектр")
        ]
driver_and_carpark_data = [
            DriverAndCarPark(1, 1),
            DriverAndCarPark(1, 3),
            DriverAndCarPark(1, 5),
            DriverAndCarPark(2, 2),
            DriverAndCarPark(3, 1),
            DriverAndCarPark(3, 4),
            DriverAndCarPark(4, 2),
            DriverAndCarPark(4, 5),
            DriverAndCarPark(5, 1),
            DriverAndCarPark(5, 2),
            DriverAndCarPark(5, 4),
            DriverAndCarPark(6, 1),
            DriverAndCarPark(7, 3),
            DriverAndCarPark(7, 4),
            DriverAndCarPark(8, 2),
        ]

class TestCarParkQueries(unittest.TestCase):
    def test_query_1(self):
        result = query_1(drivers_data, carparks_data, driver_and_carpark_data)
        expected = [
            "Водитель: Абрамов, Автопарк: Городской Флот",
            "Водитель: Иванов, Автопарк: Звездный",
            "Водитель: Михалёв, Автопарк: ЭкоМобиль",
            "Водитель: Петров, Автопарк: Городской Флот",
            "Водитель: Радченко, Автопарк: АвтоОазис",
            "Водитель: Сидоренко, Автопарк: Городской Флот",
            "Водитель: Сидоров, Автопарк: АвтоСпектр",
            "Водитель: Тюльпанов, Автопарк: АвтоСпектр"
        ]
        self.assertEqual(result, expected)

    def test_query_2(self):
        result = query_2(drivers_data, carparks_data)
        expected = [
            "Автопарк: Городской Флот, Количество водителей: 3",
            "Автопарк: АвтоСпектр, Количество водителей: 2",
            "Автопарк: АвтоОазис, Количество водителей: 1",
            "Автопарк: Звездный, Количество водителей: 1",
            "Автопарк: ЭкоМобиль, Количество водителей: 1"
        ]
        self.assertEqual(result, expected)

    def test_query_3(self):
        result = query_3(drivers_data, carparks_data, driver_and_carpark_data)
        expected = [
            "Водитель: Иванов, Автопарк(и): ЭкоМобиль, Городской Флот, АвтоСпектр",
            "Водитель: Петров, Автопарк(и): АвтоОазис",
            "Водитель: Сидоров, Автопарк(и): ЭкоМобиль, Звездный",
            "Водитель: Абрамов, Автопарк(и): Городской Флот, Звездный",
            "Водитель: Тюльпанов, Автопарк(и): АвтоОазис"
        ]
        self.assertEqual(result, expected)
if __name__ == '__main__':
    unittest.main()
```

Результат работы: