Файл rk2.py

```
from collections import Counter, defaultdict

class Course:
    def __init__(self, course_id, name):
        self.course_id = course_id
        self.name = name

class Teacher:
    def __init__(self, teacher_id, fio, sal, course_id):
        self.teacher_id = teacher_id
        self.fio = fio
        self.sal = sal
        self.course_id = course_id

class TeacherCourse:
    def __init__(self, teacher_id, course_id):
        self.teacher_id = teacher_id
        self.course_id = course_id

def get_sorted_teachers(teachers_data):
    return sorted(teachers_data, key=lambda x: x.fio)

def get_course_dict(courses_data):
    return {course.course_id: course.name for course in courses_data}

def query_1(teachers_data, courses_data, teacher_and_course_data):
    sorted_teachers = get_sorted_teachers(teachers_data)
    course_dict = get_course_dict(courses_data)
    result = []
    for teacher in sorted_teachers:
        course_name = course_dict.get(teacher.course_id, 'Неизвестный учебный
курс')
        result.append(f"Преподаватель: {teacher.fio}, Учебный курс:
{course_name}")
    return result

def query_2(teachers_data, courses_data):
    num_teachers = Counter(d.course_id for d in teachers_data)
    sorted_courses = sorted(courses_data, key=lambda x: x.name)
    sorted_courses = sorted(sorted_courses, key=lambda x:
num_teachers[x.course_id], reverse=True)
    result = []
    for course in sorted_courses:
        result.append(f"Учебный курс: {course.name}, Количество преподавателей:
{num_teachers[course.course_id]}")
    return result
```

```
def query_3(teachers_data, courses_data, teacher_and_course_data):
    teacher_course_dict = defaultdict(list)
    course_dict = get_course_dict(courses_data)
    for cd in teacher_and_course_data:
        teacher_course_dict[cd.teacher_id].append(cd.course_id)
    filtered_teachers = [teacher for teacher in teachers_data if
teacher.fio.endswith("ов")]
    result = []
    for teacher in filtered_teachers:
        courses = [course_dict[course_id] for course_id in
teacher_course_dict.get(teacher.teacher_id, [])]
        result.append(f"Преподаватель: {teacher.fio}, Учебный курс(ы): {',
'.join(courses)}")
    return result
```

Файл test.py

```
import unittest
from rk2 import Teacher, Course, TeacherCourse, query_1, query_2, query_3

teachers_data = [
            Teacher(1, 'Куликов', 75000, 1),
            Teacher(2, 'Мальцев', 105000, 1),
            Teacher(3, 'Борисенко', 93000, 2),
            Teacher(4, 'Кузнецов', 78000, 3),
            Teacher(5, 'Аксенов', 81000, 3)
        ]
courses_data = [
            Course(1, "Теория вероятностей"),
            Course(2, "Модели данных"),
            Course(3, "Правоведение"),
        ]
teacher_and_course_data = [
            TeacherCourse(1, 1),
            TeacherCourse(2, 1),
            TeacherCourse(3, 2),
            TeacherCourse(4, 3),
            TeacherCourse(5, 3)
        ]

class TestTeacherCourseQueries(unittest.TestCase):
    def test_query_1(self):
        result = query_1(teachers_data, courses_data, teacher_and_course_data)
        expected = [
            "Преподаватель: Аксенов, Учебный курс: Правоведение",
            "Преподаватель: Борисенко, Учебный курс: Модели данных",
            "Преподаватель: Кузнецов, Учебный курс: Правоведение",
            "Преподаватель: Куликов, Учебный курс: Теория вероятностей",
            "Преподаватель: Мальцев, Учебный курс: Теория вероятностей"
        ]
        self.assertEqual(result, expected)
```

```
    def test_query_2(self):
        result = query_2(teachers_data, courses_data)
        expected = [
            "Учебный курс: Правоведение, Количество преподавателей: 2",
            "Учебный курс: Теория вероятностей, Количество преподавателей: 2",
            "Учебный курс: Модели данных, Количество преподавателей: 1"
        ]
        self.assertEqual(result, expected)

    def test_query_3(self):
        result = query_3(teachers_data, courses_data, teacher_and_course_data)
        expected = [
        "Преподаватель: Куликов, Учебный курс(ы): Теория вероятностей",
        "Преподаватель: Кузнецов, Учебный курс(ы): Правоведение",
        "Преподаватель: Аксенов, Учебный курс(ы): Правоведение"
    ]

        self.assertEqual(result, expected)

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

Результат тестирования:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```