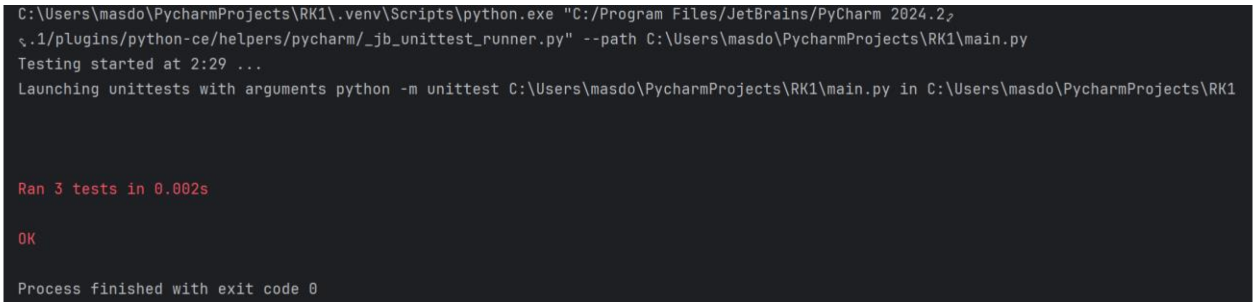
Рефакторинг кода:
```python
from operator import itemgetter
from typing import List, Tuple, Dict
import unittest

class DataRow:
    def __init__(self, emp_id: int, name: str, salary: int, dep_id: int):
        self.emp_id = emp_id
        self.name = name
        self.salary = salary
        self.dep_id = dep_id

class DataTable:
    def __init__(self, dep_id: int, name: str):
        self.dep_id = dep_id
        self.name = name

class EmpDep:
    def __init__(self, dep_id: int, emp_id: int):
        self.dep_id = dep_id
        self.emp_id = emp_id

def get_one_to_many(deps: List[DataTable], emps: List[DataRow]) ->
List[Tuple[str, int, str]]:
    return [(e.name, e.salary, d.name) for d in deps for e in emps if
e.dep_id == d.dep_id]

def get_sorted_one_to_many(one_to_many: List[Tuple[str, int, str]]) ->
List[Tuple[str, int, str]]:
    return sorted(one_to_many, key=itemgetter(0))

def get_department_counts(deps: List[DataTable], emps: List[DataRow]) ->
Dict[str, int]:
    return {d.name: sum(1 for e in emps if e.dep_id == d.dep_id) for d in
deps}

def get_sorted_department_counts(department_counts: Dict[str, int]) ->
List[Tuple[str, int]]:
    return sorted(department_counts.items(), key=lambda x: (-x[1], x[0]))

def get_many_to_many(deps: List[DataTable], emps: List[DataRow], emp_deps:
List[EmpDep]) -> List[Tuple[str, str]]:
    return [
        (e.name, d.name)
        for d in deps
        for ed in emp_deps
        if d.dep_id == ed.dep_id and d.name == 'Бухгалтерия'
        for e in emps
        if e.emp_id == ed.emp_id and e.name.endswith('ов')
    ]


def main():
    deps = [
        DataTable(1, 'Отдел кадров'),
        DataTable(2, 'Архивный отдел ресурсов'),
        DataTable(3, 'Бухгалтерия'),
    ]

    emps = [
        DataRow(1, 'Артамонов', 25000, 1),
        DataRow(2, 'Петров', 35000, 2),
        DataRow(3, 'Иваненко', 45000, 3),
        DataRow(4, 'Иванов', 35000, 3),
        DataRow(5, 'Иванин', 25000, 3),
    ]

    emp_deps = [
        EmpDep(1, 1),
        EmpDep(2, 2),
        EmpDep(3, 3),
        EmpDep(3, 4),
        EmpDep(3, 5),
    ]

    one_to_many = get_one_to_many(deps, emps)
    sorted_one_to_many = get_sorted_one_to_many(one_to_many)
    print('Запрос 1: Список сотрудников с отделами, отсортированный по 
фамилиям:')
    print(sorted_one_to_many)

    department_counts = get_department_counts(deps, emps)
    sorted_department_counts = 
get_sorted_department_counts(department_counts)
    print('\nЗапрос 2: Количество сотрудников в каждом отделе, 
отсортированное по количеству сотрудников:')
    print(sorted_department_counts)

    many_to_many = get_many_to_many(deps, emps, emp_deps)
    print('\nЗапрос 3: Сотрудники с фамилией на "ов" и названия их отделов:')
    print(many_to_many)

if __name__ == '__main__':
    main()

class TestEmployeeQueries(unittest.TestCase):
    def setUp(self):
        self.deps = [
            DataTable(1, 'Отдел кадров'),
            DataTable(2, 'Архивный отдел ресурсов'),
            DataTable(3, 'Бухгалтерия'),
        ]

        self.emps = [
            DataRow(1, 'Артамонов', 25000, 1),
            DataRow(2, 'Петров', 35000, 2),
            DataRow(3, 'Иваненко', 45000, 3),
            DataRow(4, 'Иванов', 35000, 3),
            DataRow(5, 'Иванин', 25000, 3),
        ]

        self.emp_deps = [
            EmpDep(1, 1),
            EmpDep(2, 2),
            EmpDep(3, 3),
            EmpDep(3, 4),
            EmpDep(3, 5),
        ]

    def test_get_one_to_many(self):
        expected = [
            ('Артамонов', 25000, 'Отдел кадров'),
            ('Петров', 35000, 'Архивный отдел ресурсов'),
            ('Иваненко', 45000, 'Бухгалтерия'),
            ('Иванов', 35000, 'Бухгалтерия'),
            ('Иванин', 25000, 'Бухгалтерия'),
        ]
        self.assertEqual(get_one_to_many(self.deps, self.emps), expected)
    def test_get_sorted_department_counts(self):
        counts = get_department_counts(self.deps, self.emps)
        expected = [('Бухгалтерия', 3), ('Архивный отдел ресурсов', 1),
('Отдел кадров', 1)]
        self.assertEqual(get_sorted_department_counts(counts), expected)

    def test_get_many_to_many(self):
        expected = [('Иванов', 'Бухгалтерия')]
        self.assertEqual(get_many_to_many(self.deps, self.emps,
self.emp_deps), expected)

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

Результат тестирования:



```
C:\Users\masdo\PycharmProjects\RK1\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm 2024.2
.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --path C:\Users\masdo\PycharmProjects\RK1\main.py
Testing started at 2:29 ...
Launching unittests with arguments python -m unittest C:\Users\masdo\PycharmProjects\RK1\main.py in C:\Users\masdo\PycharmProjects\RK1



Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```