```python
# -*- coding: utf-8 -*-
from dataclasses import dataclass
from typing import List, Tuple

# ---------- модели ----------

@dataclass
class Supplier: # Поставщик
id: int
name: str

@dataclass
class Detail: # Деталь
id: int
name: str
price: int
supplier_id: int # основной поставщик (1-ко-многим)

@dataclass
class DetailSupplier: # связь многие-ко-многим
detail_id: int
supplier_id: int

# ---------- подготовка данных ----------

def create_test_data() -> tuple[
List[Supplier], List[Detail], List[DetailSupplier]
]:
"""Возвращает тестовые данные для задач."""
suppliers = [
Supplier(1, "Иванов ИП"),
Supplier(2, "МеталлПром"),
Supplier(3, "Детали+"),
Supplier(4, "Технолов"),
Supplier(5, "СтройСнаб"),
]

details = [
Detail(1, "Гайка", 7, 1),
Detail(2, "Шайба", 5, 1),
Detail(3, "Болт", 10, 2),
Detail(4, "Втулка", 12, 2),
Detail(5, "Крышка", 15, 3),
Detail(6, "Ручка", 20, 4),
Detail(7, "Прокладка", 8, 5),
]
```

```
detail_suppliers = [
DetailSupplier(1, 1),
DetailSupplier(2, 1),
DetailSupplier(3, 2),
DetailSupplier(4, 2),
DetailSupplier(5, 3),
DetailSupplier(6, 4),
DetailSupplier(7, 5),
DetailSupplier(2, 3),
DetailSupplier(4, 5),
DetailSupplier(5, 2),
]

return suppliers, details, detail_suppliers


# ---------- построение связей ----------

def make_one_to_many(
suppliers: List[Supplier],
details: List[Detail],
) -> List[tuple]:
"""Связь один-ко-многим: (деталь, цена, поставщик)."""
return [
(d.name, d.price, s.name)
for s in suppliers
for d in details
if d.supplier_id == s.id
]


def make_many_to_many(
suppliers: List[Supplier],
details: List[Detail],
link_table: List[DetailSupplier],
) -> List[tuple]:
"""Связь многие-ко-многим: (деталь, цена, поставщик)."""
temp = [
(s.name, ds.detail_id)
for s in suppliers
for ds in link_table
if s.id == ds.supplier_id
]
return [
(d.name, d.price, s_name)
for s_name, detail_id in temp
for d in details
if d.id == detail_id
]
```

```python
# ---------- функции для трёх запросов варианта Б ----------

def task1(one_to_many: List[tuple]) -> List[tuple]:
"""1) Все пары деталь–поставщик (1-ко-многим), сортировка по деталям."""
return sorted(one_to_many, key=lambda x: x[0])


def task2(
suppliers: List[Supplier],
details: List[Detail],
) -> List[tuple]:
"""
2) Список поставщиков с количеством их деталей,
отсортированный по количеству (по убыванию).
"""
res = [
(s.name, len([d for d in details if d.supplier_id == s.id]))
for s in suppliers
]
res = [item for item in res if item[1] > 0]
return sorted(res, key=lambda x: x[1], reverse=True)


def task3(many_to_many: List[tuple]) -> List[tuple]:
"""
3) Многие-ко-многим. Все детали, название которых
оканчивается на «ка», и названия их поставщиков.
"""
return [
(detail_name, supplier_name)
for detail_name, _, supplier_name in many_to_many
if detail_name.endswith("ка")
]


# ---------- демонстрация работы (не используется в тестах) ----------

def main():
suppliers, details, link_table = create_test_data()
one_to_many = make_one_to_many(suppliers, details)
many_to_many = make_many_to_many(suppliers, details, link_table)

print("Задание 1:")
for row in task1(one_to_many):
print(row)

print("\nЗадание 2:")
for row in task2(suppliers, details):
print(row)

print("\nЗадание 3:")
```

```
for row in task3(many_to_many):
print(row)

name = "main"
if name == "main":
main()
```

```
oem@UbuntuArm:~/labs/Andreev-Sem03/labs/RK2$ python3 RK2.py
Задание 1:
('Болт', 10, 'МеталлПром')
('Втулка', 12, 'МеталлПром')
('Гайка', 7, 'Иванов ИП')
('Крышка', 15, 'Детали+')
('Прокладка', 8, 'СтройСнаб')
('Ручка', 20, 'Технолов')
('Шайба', 5, 'Иванов ИП')

Задание 2:
('Иванов ИП', 2)
('МеталлПром', 2)
('Детали+', 1)
('Технолов', 1)
('СтройСнаб', 1)

Задание 3:
('Гайка', 'Иванов ИП')
('Втулка', 'МеталлПром')
('Крышка', 'МеталлПром')
('Крышка', 'Детали+')
('Ручка', 'Технолов')
('Прокладка', 'СтройСнаб')
('Втулка', 'СтройСнаб')
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```