```
from operator import itemgetter


class HardDrive:
    """Hard Drive"""
    def __init__(self, id, model, capacity_gb, computer_id):
        self.id = id
        self.model = model
        self.capacity_gb = capacity_gb
        self.computer_id = computer_id


class Computer:
    """Computer"""
    def __init__(self, id, name, type, hard_drive_id, cost):
        self.id = id
        self.name = name
        self.type = type
        self.hard_drive_id = hard_drive_id
        self.cost = cost


class ComputerHardDrive:
    """'Computers with Hard Drives' for implementing many-to-many relationship"""
    def __init__(self, computer_id, hard_drive_id):
        self.computer_id = computer_id
        self.hard_drive_id = hard_drive_id


# Hard Drives
hard_drives = [
    HardDrive(1, 'Seagate 1TB', 1000, 1),
    HardDrive(2, 'Western Digital 2TB', 2000, 2),
    HardDrive(3, 'Samsung 500GB', 500, 2),
]
```

```
# Computers
computers = [
    Computer(1, 'Computer 1', 'Desktop', 1, 800),
    Computer(2, 'Laptop 1', 'Laptop', 2, 1200),
    Computer(3, 'Computer 2', 'Desktop', 3, 700),
]

computer_hard_drives = [
    ComputerHardDrive(1, 1),
    ComputerHardDrive(2, 2),
    ComputerHardDrive(3, 3),
    ComputerHardDrive(3, 2),
]

def main():
    """Main function"""

    # One-to-many relationship
    one_to_many = [(h.model, h.capacity_gb, c.name)
        for c in computers
        for h in hard_drives
        if h.computer_id == c.id]

    # Many-to-many relationship
    many_to_many_temp = [(c.name, ch.computer_id, ch.hard_drive_id)
        for c in computers
        for ch in computer_hard_drives
        if c.id == ch.computer_id]

    many_to_many = [(h.model, h.capacity_gb, comp_name)
        for comp_name, comp_id, hd_id in many_to_many_temp
        for h in hard_drives if h.id == hd_id]
    print('Task A1')
    res_a1 = sorted(one_to_many, key=itemgetter(2))
    print(res_a1)

    print('\nTask A2')
    res_a2_unsorted = []
    for c in computers:
        c_hard_drives = list(filter(lambda i: i[2] == c.name, one_to_many))
        if len(c_hard_drives) > 0:
            c_capacities = [capacity for _, capacity, _ in c_hard_drives]
            c_capacity_sum = sum(c_capacities)
            res_a2_unsorted.append((c.name, c_capacity_sum))

    res_a2 = sorted(res_a2_unsorted, key=itemgetter(1), reverse=True)
    print(res_a2)

    print('\nTask A3')
    res_a3 = {}
    for c in computers:
        if 'computer' in c.name.lower():
            c_hard_drives = list(filter(lambda i: i[2] == c.name, many_to_many))
            c_hard_drive_models = [x for x, _, _ in c_hard_drives]
            res_a3[c.name] = c_hard_drive_models

    print(res_a3)

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

Результаты выполнения:

Task A1

```
[('Seagate 1TB', 1000, 'Computer 1'), ('Western Digital 2TB', 2000, 'Laptop 1'), ('Samsung 500GB', 500,
'Laptop 1')]
```

Task A2

```
[('Laptop 1', 2500), ('Computer 1', 1000)]
```

Task A3

```
{'Computer 1': ['Seagate 1TB'], 'Computer 2': ['Samsung 500GB', 'Western Digital 2TB']}
```