## Текст программы

### main.py

```
from operator import itemgetter

       class DataTable:
           def __init__(self, id, name, size_mb, db_id):
               self.id = id
               self.name = name
               self.size_mb = size_mb
               self.db_id = db_id


       class Database:
           def __init__(self, id, name):
               self.id = id
               self.name = name


       class TableDatabase:
           def __init__(self, db_id, dt_id):
               self.db_id = db_id
               self.dt_id = dt_id


       def get_one_to_many(databases, data_tables):
           return [(dt.name, dt.size_mb, db.name)
                   for db in databases
                   for dt in data_tables
                   if dt.db_id == db.id]


       def get_many_to_many(databases, data_tables, table_dbs):
           many_to_many_temp = [(db.name, tdb.db_id, tdb.dt_id)
                                for db in databases
                                for tdb in table_dbs
                                if db.id == tdb.db_id]

           return [(dt.name, dt.size_mb, db_name)
                   for db_name, db_id, dt_id in
many_to_many_temp
                   for dt in data_tables if dt.id == dt_id]



       def get_dbs_starting_with_a(one_to_many_result):
           dbs_starting_with_a = [db for db in
one_to_many_result if db[2][0] == "A"]
           result = {}
           for db in dbs_starting_with_a:
               db_name = db[2]
```

```
            if db_name not in result:
                result[db_name] = []
            result[db_name].append(db[0]) # only add table
names

        return result



    def get_db_max_sizes(one_to_many_result):
        db_max_sizes = {}
        for _, size, db_name in one_to_many_result:
            if db_name not in db_max_sizes:
                db_max_sizes[db_name] = 0
            db_max_sizes[db_name] =
max(db_max_sizes[db_name], size)
        return sorted(db_max_sizes.items(),
key=itemgetter(1), reverse=True)



    def get_sorted_many_to_many(many_to_many_result):
        return sorted(many_to_many_result, key=itemgetter(2))


    def main():
        databases = [
            Database(1, 'AnalyticsDB'),
            Database(2, 'ProductsDB'),
            Database(3, 'ArchiveDB'),
            Database(4, 'OrdersDB')
        ]

        data_tables = [
            DataTable(1, 'Users', 50, 1),
            DataTable(2, 'Products', 100, 2),
            DataTable(3, 'Orders', 200, 4),
            DataTable(4, 'Archive', 500, 3),
            DataTable(5, 'Addresses', 20, 4)
        ]

        table_dbs = [
            TableDatabase(1, 1),
            TableDatabase(2, 2),
            TableDatabase(3, 4),
            TableDatabase(4, 3),
            TableDatabase(5, 4),
            TableDatabase(2, 5),
            TableDatabase(3, 1),
            TableDatabase(4, 5)
        ]
```

```
            one_to_many = get_one_to_many(databases, data_tables)
            many_to_many = get_many_to_many(databases,
data_tables, table_dbs)

            print("\nЗадание Г1")
            dbs_starting_with_a =
get_dbs_starting_with_a(one_to_many)
            for db_name, tables in dbs_starting_with_a.items():
                print(f"База данных: {db_name}, Таблицы:
{tables}")



            print("\nЗадание Г2")
            db_max_sizes = get_db_max_sizes(one_to_many)
            print(db_max_sizes)

            print("\nЗадание Г3")
            sorted_many_to_many =
get_sorted_many_to_many(many_to_many)
            print(sorted_many_to_many)


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

**test.py**

```
import unittest
from main import get_one_to_many, get_many_to_many,
get_dbs_starting_with_a, get_db_max_sizes,
get_sorted_many_to_many, Database, DataTable, TableDatabase

class TestYourModule(unittest.TestCase):

    def test_get_one_to_many(self):
        databases = [Database(1, 'TestDB')]
        data_tables = [DataTable(1, 'TestTable', 10, 1)]
        expected_result = [('TestTable', 10, 'TestDB')]
        self.assertEqual(get_one_to_many(databases,
data_tables), expected_result)


    def test_get_dbs_starting_with_a(self):
        one_to_many_result = [('Users', 50, 'AnalyticsDB'),
('Archive', 500, 'ArchiveDB')]
        expected_result = {'AnalyticsDB': ['Users'],
'ArchiveDB': ['Archive']}
```

```
self.assertEqual(get_dbs_starting_with_a(one_to_many_result),
expected_result)


    def test_get_db_max_sizes(self):
        one_to_many_result = [('Users', 50, 'AnalyticsDB'),
('Products', 100, 'ProductsDB'), ('Orders', 20,
'OrdersDB'),('Orders2', 200, 'OrdersDB')]
        expected_result = [('OrdersDB', 200), ('ProductsDB',
100), ('AnalyticsDB', 50)]
        self.assertEqual(get_db_max_sizes(one_to_many_result),
expected_result)



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

## Пример работы программы