```python
# Classes for One-to-Many relationship
class OneToManyOperator:
    def __init__(self, operator_id, symbol, description, salary, language_id):
        self.operator_id = operator_id
        self.symbol = symbol
        self.description = description
        self.salary = salary
        self.language_id = language_id

class OneToManyProgrammingLanguage:
    def __init__(self, language_id, language_name, version):
        self.language_id = language_id
        self.language_name = language_name
        self.version = version

# Classes for Many-to-Many relationship
class Operator:
    def __init__(self, operator_id, symbol, description, salary):
        self.operator_id = operator_id
        self.symbol = symbol
        self.description = description
        self.salary = salary

class ProgrammingLanguage:
    def __init__(self, language_id, language_name, version):
        self.language_id = language_id
        self.language_name = language_name
        self.version = version

class OperatorsLanguages:
    def __init__(self, operator_id, language_id):
        self.operator_id = operator_id
        self.language_id = language_id

# Function to create examples
def create_examples():
    languages = [
        OneToManyProgrammingLanguage(1, "Python", "3.10"),
        OneToManyProgrammingLanguage(2, "Java", "17"),
        OneToManyProgrammingLanguage(3, "C++", "20"),
        OneToManyProgrammingLanguage(4, "JavaScript", "ES2021"),
    ]

    operators = [
        OneToManyOperator(1, "+", "Addition", 50000, 1),
        OneToManyOperator(2, "-", "Subtraction", 60000, 1),
        OneToManyOperator(3, "*", "Multiplication", 70000, 1),
        OneToManyOperator(4, "/", "Division", 55000, 1),
        OneToManyOperator(5, "++", "Increment (Unary)", 52000, 2),
        OneToManyOperator(6, "--", "Decrement (Unary)", 52000, 2),
        OneToManyOperator(7, "!", "Logical NOT (Unary)", 53000, 3),
        OneToManyOperator(8, "~", "Bitwise NOT (Unary)", 54000, 3),
        OneToManyOperator(9, "&&", "Logical AND", 60000, 4),
        OneToManyOperator(10, "||", "Logical OR", 60000, 4),
    ]

```

```
    many_to_many_operators = [
        Operator(1, "+", "Addition", 50000),
        Operator(2, "-", "Subtraction", 60000),
        Operator(3, "*", "Multiplication", 70000),
        Operator(4, "/", "Division", 55000),
        Operator(5, "++", "Increment (Unary)", 52000),
        Operator(6, "--", "Decrement (Unary)", 52000),
        Operator(7, "!", "Logical NOT (Unary)", 53000),
        Operator(8, "~", "Bitwise NOT (Unary)", 54000),
    ]

    many_to_many_languages = [
        ProgrammingLanguage(1, "Python", "3.10"),
        ProgrammingLanguage(2, "Java", "17"),
        ProgrammingLanguage(3, "C++", "20"),
        ProgrammingLanguage(4, "JavaScript", "ES2021"),
    ]

    operators_languages = [
        OperatorsLanguages(1, 1),  # + in Python
        OperatorsLanguages(2, 1),  # - in Python
        OperatorsLanguages(3, 1),  # * in Python
        OperatorsLanguages(4, 1),  # / in Python
        OperatorsLanguages(5, 2),  # ++ in Java
        OperatorsLanguages(6, 2),  # -- in Java
        OperatorsLanguages(1, 3),  # + in C++
        OperatorsLanguages(7, 3),  # ! in C++
        OperatorsLanguages(9, 4),  # && in JavaScript
        OperatorsLanguages(10, 4),  # || in JavaScript
        OperatorsLanguages(8, 3),  # ~ in C++
    ]

    return languages, operators, many_to_many_operators, many_to_many_languages, operators_languages
```

```
# Functions for One-to-Many queries
def one_to_many_query(languages, operators):
    # Query for languages starting with 'J'
    print("Languages starting with 'J' and their operators:")
    for lang in languages:
        if lang.language_name.startswith("J"):
            print(f"Language: {lang.language_name}")
            for op in operators:
                if op.language_id == lang.language_id:
                    print(f"  Operator: {op.symbol} ({op.description})")

def languages_with_max_operators(languages, operators):
    # Count operators for each language
    operator_count = {lang.language_id: 0 for lang in languages}

    for op in operators:
        operator_count[op.language_id] += 1

    # Create a sorted list of languages by operator count
    sorted_languages = sorted(languages, key=lambda lang: operator_count[lang.language_id], reverse=True)

    print("\nLanguages sorted by the number of operators (descending):")
    for lang in sorted_languages:
        print(f"Language: {lang.language_name}, Operators Count: {operator_count[lang.language_id]}")
```

```python

# Functions for Many-to-Many queries
def many_to_many_query(languages, operators, operators_languages):
    # Create a sorted list of languages
    sorted_languages = sorted(languages, key=lambda lang: lang.language_name)

    print("\nOperators sorted by programming languages:")
    for language in sorted_languages:
        print(f"Language: {language.language_name}")
        for ol in operators_languages:
            if ol.language_id == language.language_id:
                operator = next(op for op in operators if op.operator_id == ol.operator_id)
                print(f"  Operator: {operator.symbol} ({operator.description})")

def main():
    # Create example instances
    languages, operators, many_to_many_operators, many_to_many_languages, operators_languages = create_examples()

    print("One-to-Many Example:")
    one_to_many_query(languages, operators)

    languages_with_max_operators(languages, operators)

    print("\nMany-to-Many Example:")
    many_to_many_query(many_to_many_languages, operators, operators_languages)

if __name__ == "__main__":
    main()

```

```
PS C:\Users\vladu\projects\university\sem3\PCPL_course_sem3> & C:/Users/vladu/App
se_sem3/rk1/main.py
One-to-Many Example:
Languages starting with 'J' and their operators:
Language: Java
  Operator: ++ (Increment (Unary))
  Operator: -- (Decrement (Unary))
Language: JavaScript
  Operator: && (Logical AND)
  Operator: || (Logical OR)

Languages sorted by the number of operators (descending):
Language: Python, Operators Count: 4
Language: Java, Operators Count: 2
Language: C++, Operators Count: 2
Language: JavaScript, Operators Count: 2

Many-to-Many Example:

Operators sorted by programming languages:
Language: C++
  Operator: + (Addition)
  Operator: ! (Logical NOT (Unary))
  Operator: ~ (Bitwise NOT (Unary))
Language: Java
  Operator: ++ (Increment (Unary))
  Operator: -- (Decrement (Unary))
Language: JavaScript
  Operator: && (Logical AND)
  Operator: || (Logical OR)
Language: Python
  Operator: + (Addition)
  Operator: - (Subtraction)
  Operator: * (Multiplication)
  Operator: / (Division)
PS C:\Users\vladu\projects\university\sem3\PCPL_course_sem3>
```