1. General Questions:
   a. What are the subjects offered in Program A, Program B, and Program C?
   b. Who are the faculty members for Program A, Program B, and Program C?
   c. What are the start and end dates for Program A, Program B, and Program C?

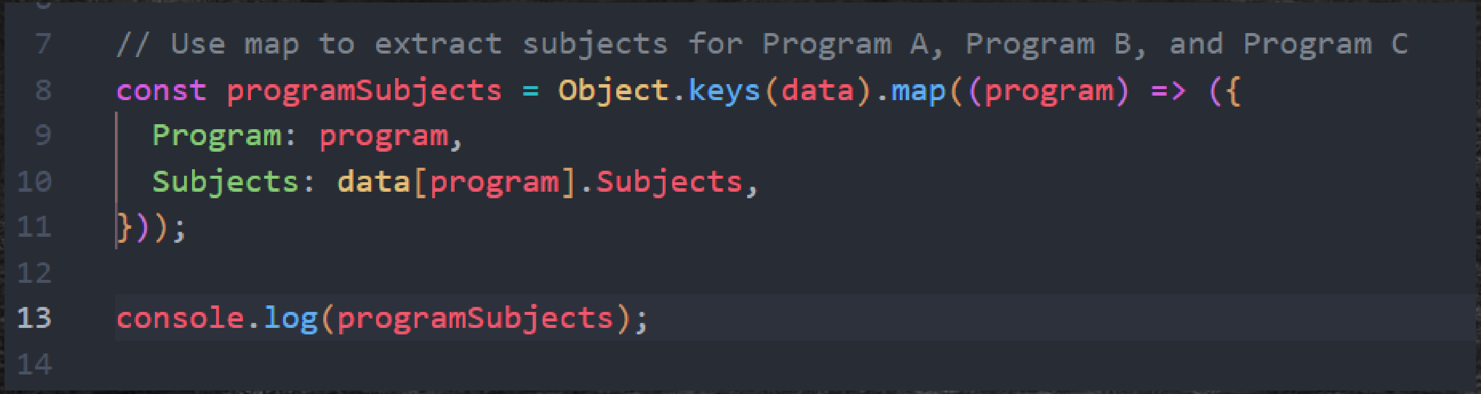
2. Branch-specific Questions:
   a. In Branch A of Program A, how many students have completed the program, joined new, and are currently ongoing?
   b. Who is the faculty member for Branch A in Program A?
   c. In Branch B of Program B, how many students have completed the program, joined new, and are currently ongoing?
   d. Who is the faculty member for Branch B in Program B?
   e. In Branch A of Program C, how many students have completed the program, joined new, and are currently ongoing?
   f. Who is the faculty member for Branch A in Program C?
   g. In Branch B of Program C, how many students have completed the program, joined new, and are currently ongoing?
   h. Who is the faculty member for Branch B in Program C?

3. Timeframe Questions:
   a. When does Program A start and end?
   b. When does Program B start and end?
   c. When does Program C start and end?
   d. Which program has the shortest duration, and which one has the longest duration?

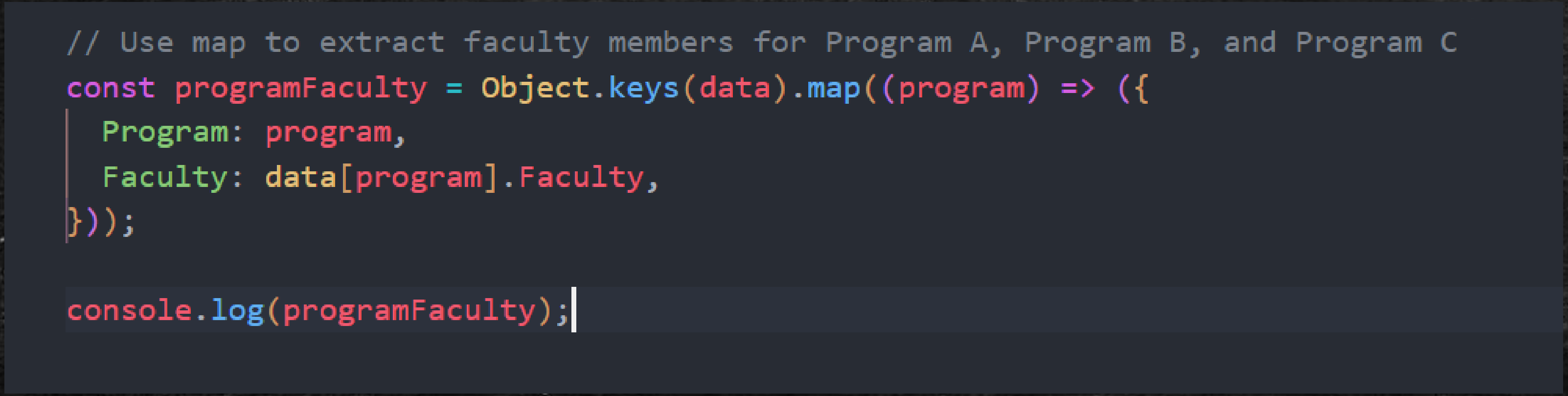
4. Faculty-related Questions:
   a. Who are the faculty members common to all three programs?
   b. Who is the faculty member with the most students in all branches combined?
   c. Which faculty member has the most students in a single branch?

5. Subject-related Questions:
   a. What are the subjects common to all three programs?
   b. Which program offers the most subjects, and how many subjects is that?

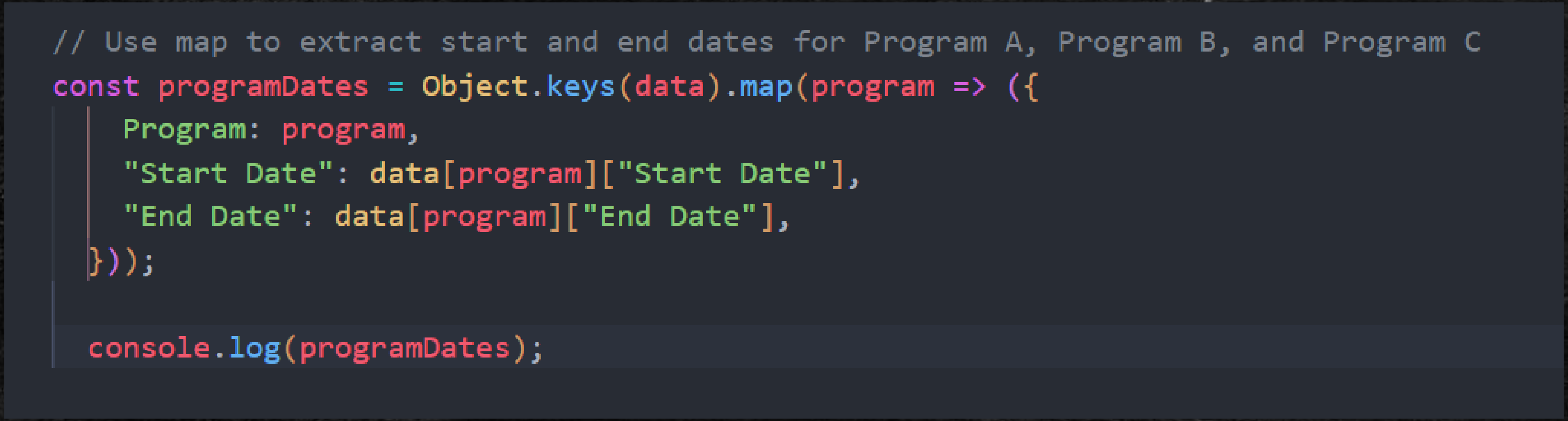
6. Enrollment and Completion Questions:
   a. Which program has the highest number of students who have completed their studies?
   b. In which branch is the highest number of students currently ongoing?
   c. Which program has the highest number of students who joined recently?



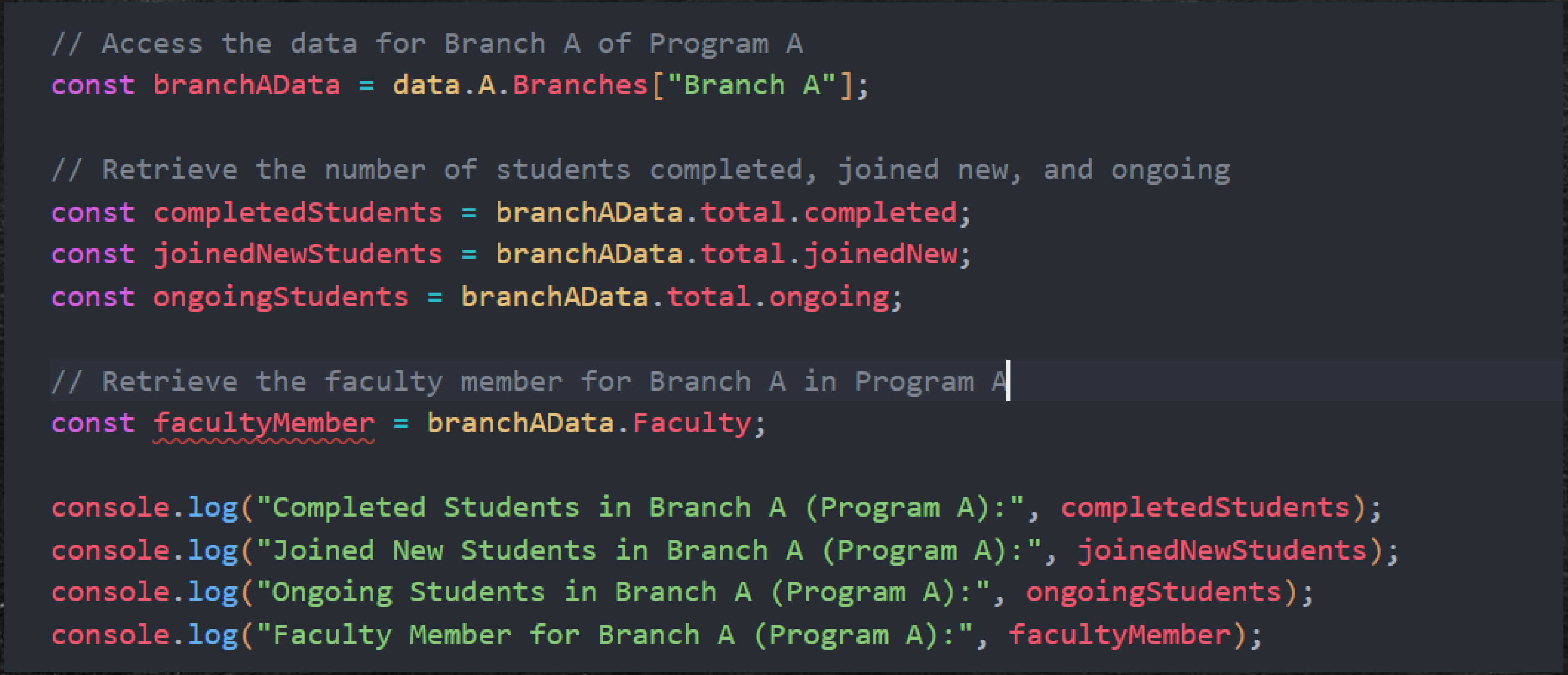
```javascript
// Use map to extract subjects for Program A, Program B, and Program C
const programSubjects = Object.keys(data).map((program) => ({
  Program: program,
  Subjects: data[program].Subjects,
}));

console.log(programSubjects);
```



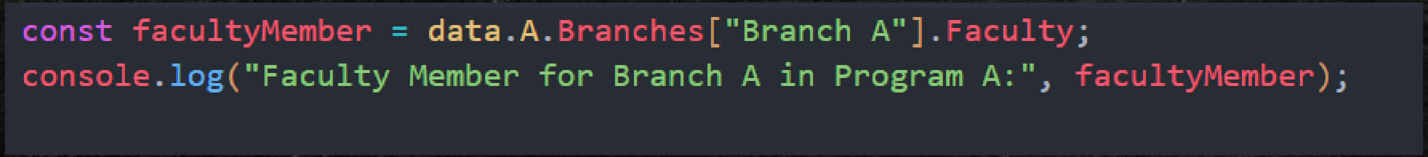
```javascript
// Use map to extract faculty members for Program A, Program B, and Program C
const programFaculty = Object.keys(data).map((program) => ({
  Program: program,
  Faculty: data[program].Faculty,
}));

console.log(programFaculty);
```



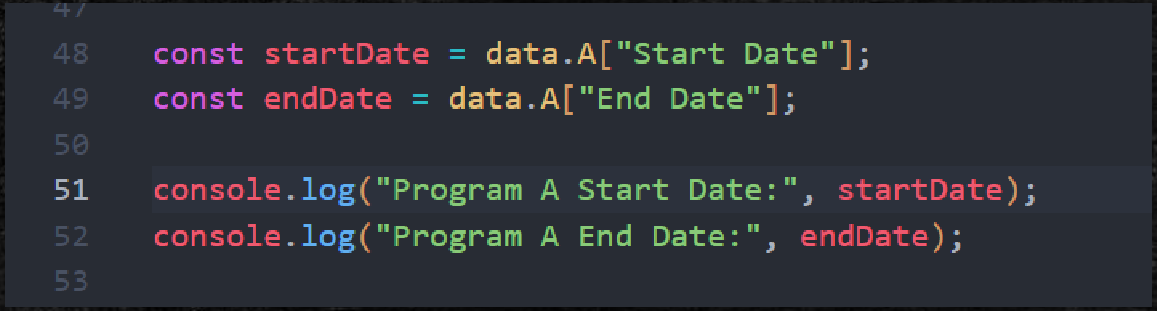
```javascript
// Use map to extract start and end dates for Program A, Program B, and Program C
const programDates = Object.keys(data).map(program => ({
    Program: program,
    "Start Date": data[program]["Start Date"],
    "End Date": data[program]["End Date"],
  }));

  console.log(programDates);
```



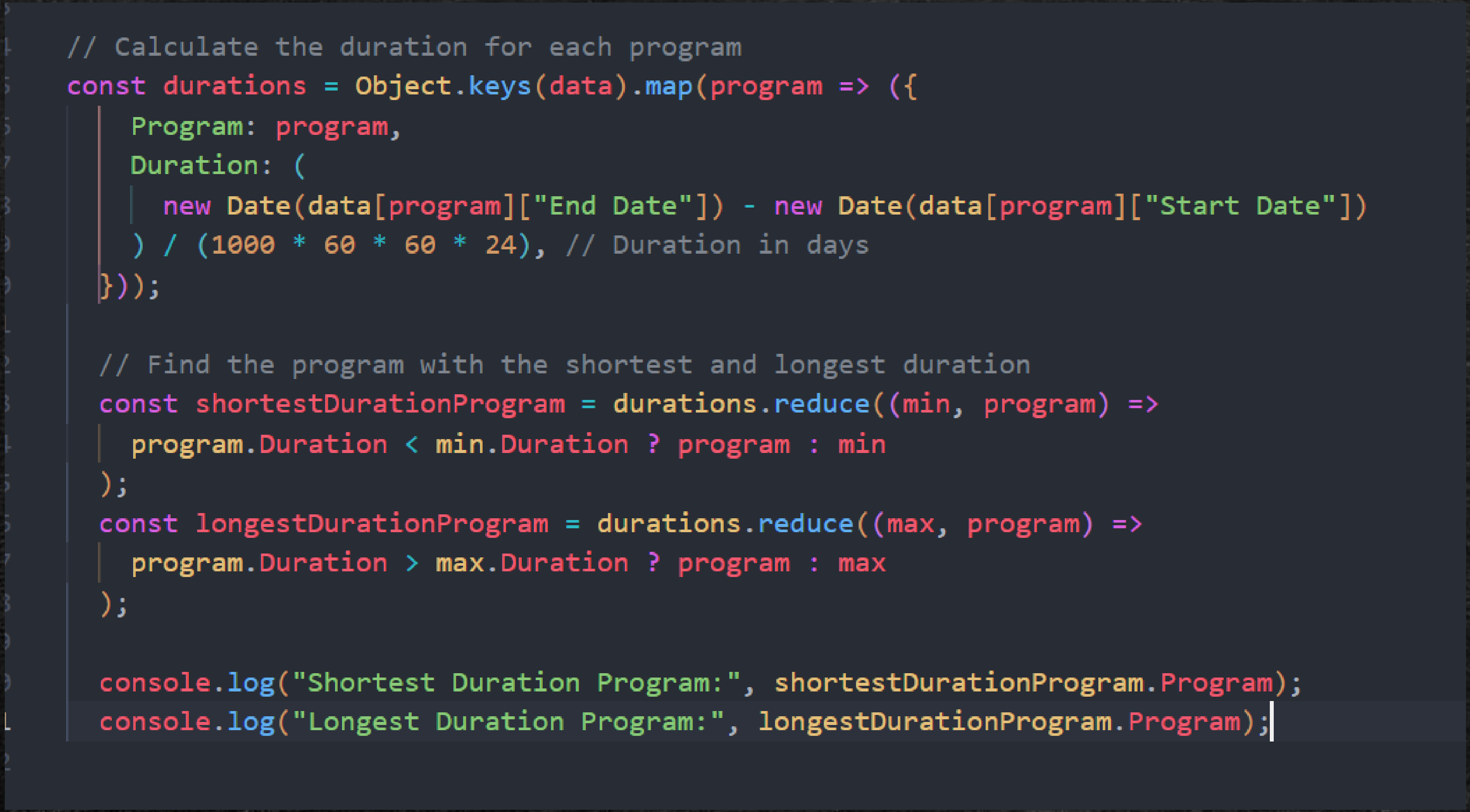
```javascript
// Access the data for Branch A of Program A
const branchAData = data.A.Branches["Branch A"];

// Retrieve the number of students completed, joined new, and ongoing
const completedStudents = branchAData.total.completed;
const joinedNewStudents = branchAData.total.joinedNew;
const ongoingStudents = branchAData.total.ongoing;

// Retrieve the faculty member for Branch A in Program A
const facultyMember = branchAData.Faculty;

console.log("Completed Students in Branch A (Program A):", completedStudents);
console.log("Joined New Students in Branch A (Program A):", joinedNewStudents);
console.log("Ongoing Students in Branch A (Program A):", ongoingStudents);
console.log("Faculty Member for Branch A (Program A):", facultyMember);
```



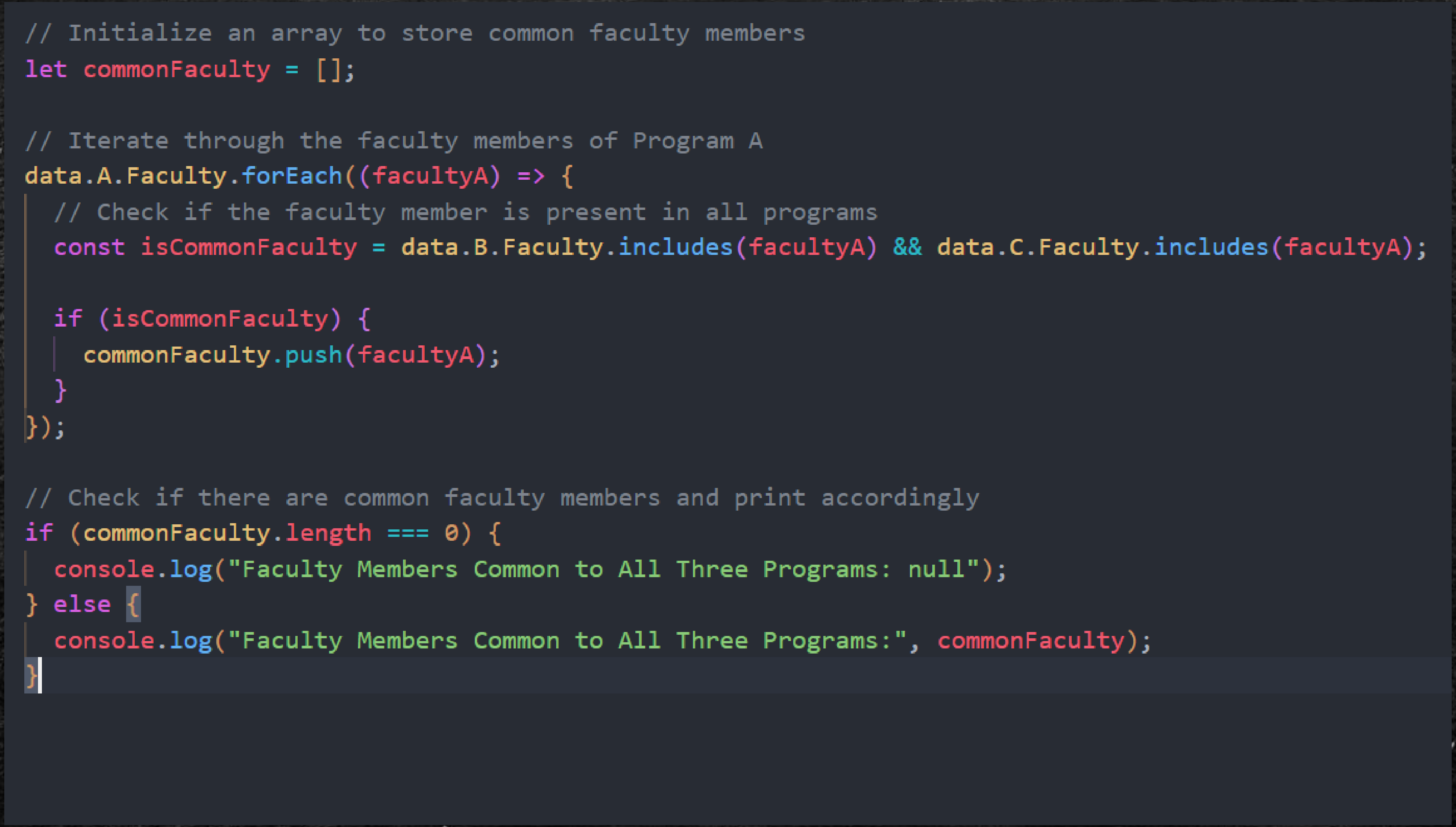
```javascript
const facultyMember = data.A.Branches["Branch A"].Faculty;
console.log("Faculty Member for Branch A in Program A:", facultyMember);
```



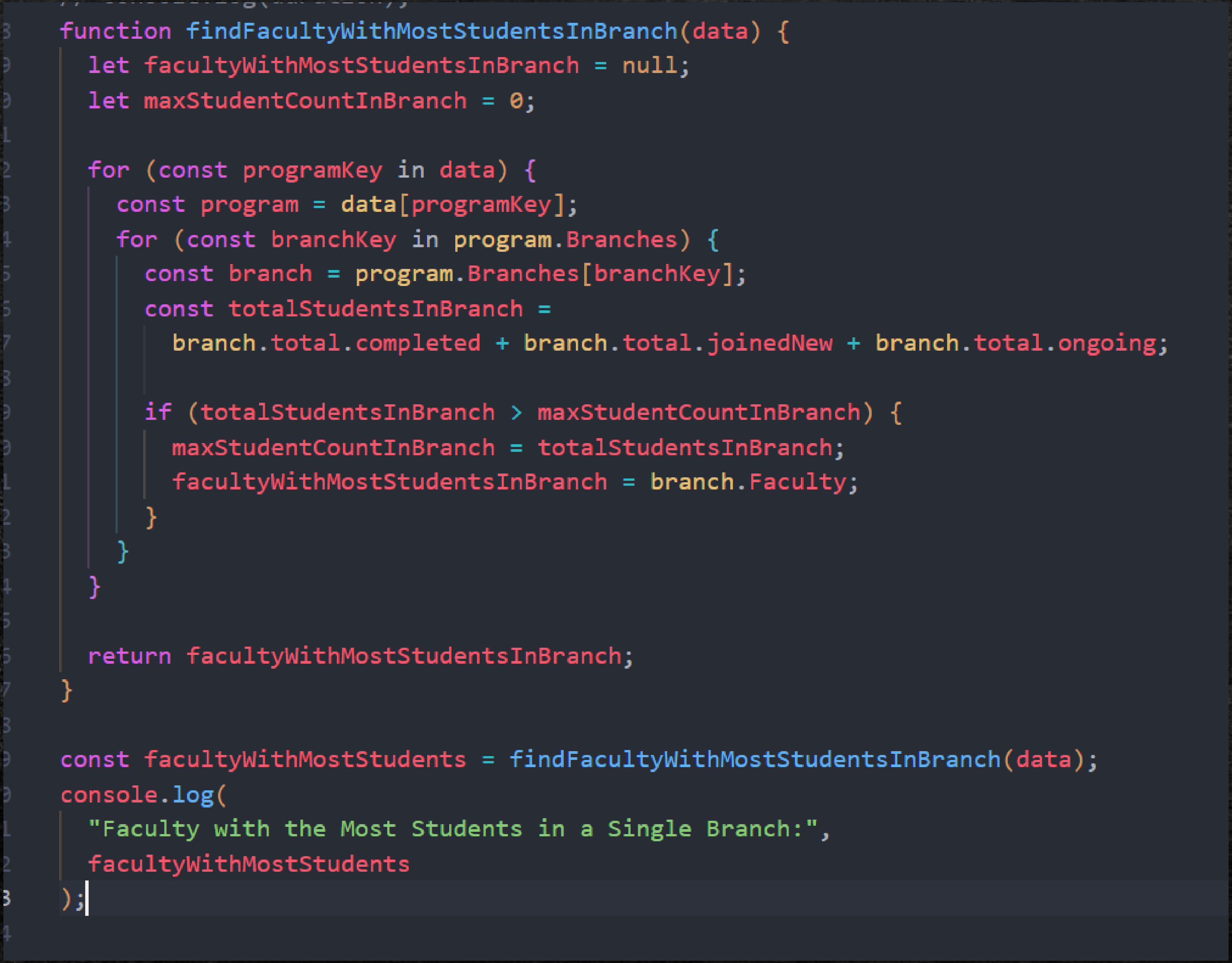
```javascript
const startDate = data.A["Start Date"];
const endDate = data.A["End Date"];

console.log("Program A Start Date:", startDate);
console.log("Program A End Date:", endDate);
```



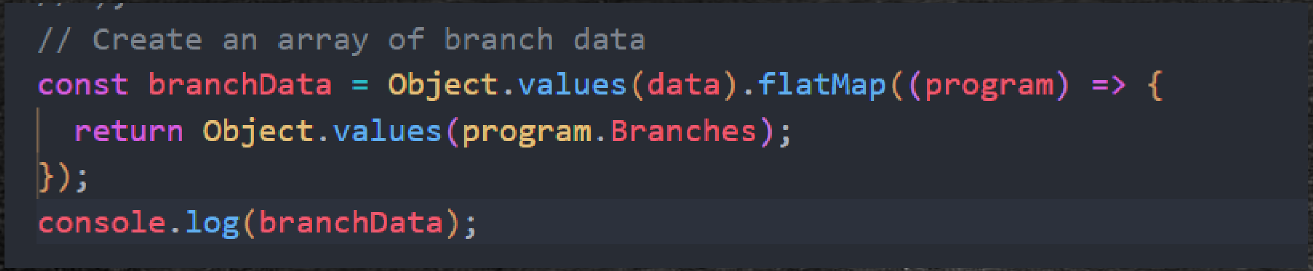
```
// Calculate the duration for each program
const durations = Object.keys(data).map(program => ({
    Program: program,
    Duration: (
      new Date(data[program]["End Date"]) - new Date(data[program]["Start Date"])
    ) / (1000 * 60 * 60 * 24), // Duration in days
  }));

  // Find the program with the shortest and longest duration
  const shortestDurationProgram = durations.reduce((min, program) =>
    program.Duration < min.Duration ? program : min
  );
  const longestDurationProgram = durations.reduce((max, program) =>
    program.Duration > max.Duration ? program : max
  );

  console.log("Shortest Duration Program:", shortestDurationProgram.Program);
  console.log("Longest Duration Program:", longestDurationProgram.Program);
```



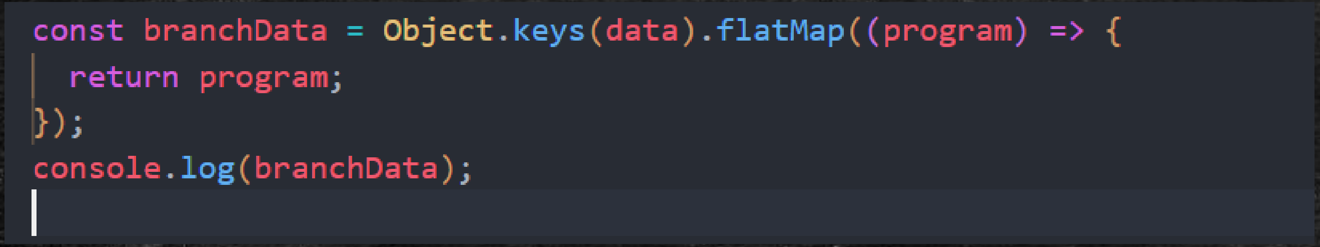
```javascript
// Initialize an array to store common faculty members
let commonFaculty = [];

// Iterate through the faculty members of Program A
data.A.Faculty.forEach((facultyA) => {
  // Check if the faculty member is present in all programs
  const isCommonFaculty = data.B.Faculty.includes(facultyA) && data.C.Faculty.includes(facultyA);

  if (isCommonFaculty) {
    commonFaculty.push(facultyA);
  }
});

// Check if there are common faculty members and print accordingly
if (commonFaculty.length === 0) {
  console.log("Faculty Members Common to All Three Programs: null");
} else {
  console.log("Faculty Members Common to All Three Programs:", commonFaculty);
}
```



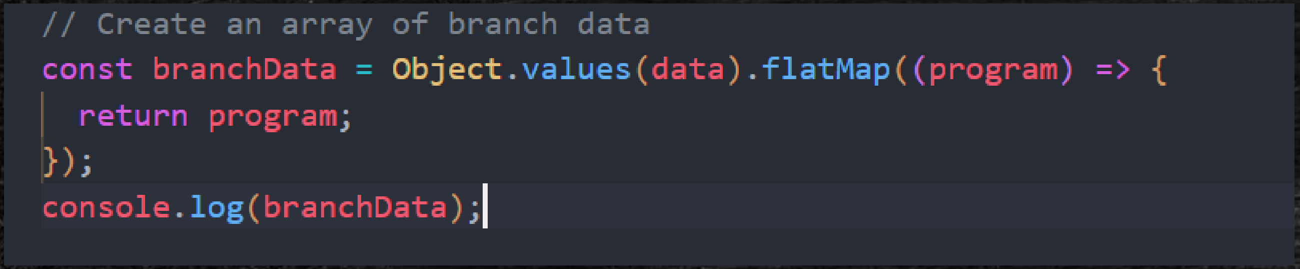
```javascript
function findFacultyWithMostStudentsInBranch(data) {
  let facultyWithMostStudentsInBranch = null;
  let maxStudentCountInBranch = 0;

  for (const programKey in data) {
    const program = data[programKey];
    for (const branchKey in program.Branches) {
      const branch = program.Branches[branchKey];
      const totalStudentsInBranch =
        branch.total.completed + branch.total.joinedNew + branch.total.ongoing;

      if (totalStudentsInBranch > maxStudentCountInBranch) {
        maxStudentCountInBranch = totalStudentsInBranch;
        facultyWithMostStudentsInBranch = branch.Faculty;
      }
    }
  }

  return facultyWithMostStudentsInBranch;
}

const facultyWithMostStudents = findFacultyWithMostStudentsInBranch(data);
console.log(
  "Faculty with the Most Students in a Single Branch:",
  facultyWithMostStudents
);
```



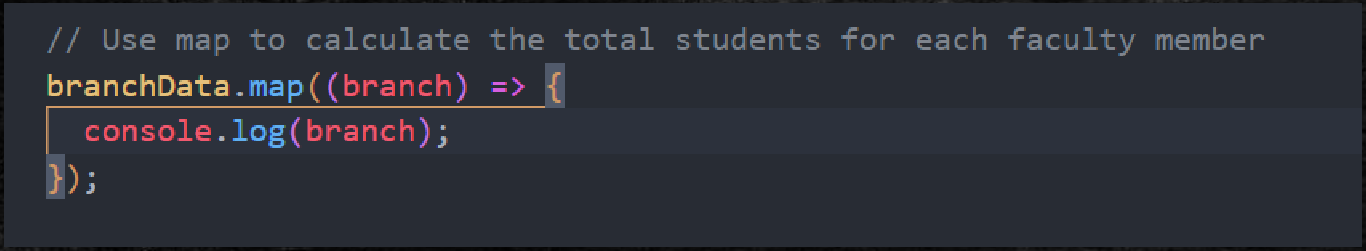
```
// Create an array of branch data
const branchData = Object.values(data).flatMap((program) => {
  return Object.values(program.Branches);
});
console.log(branchData);
```



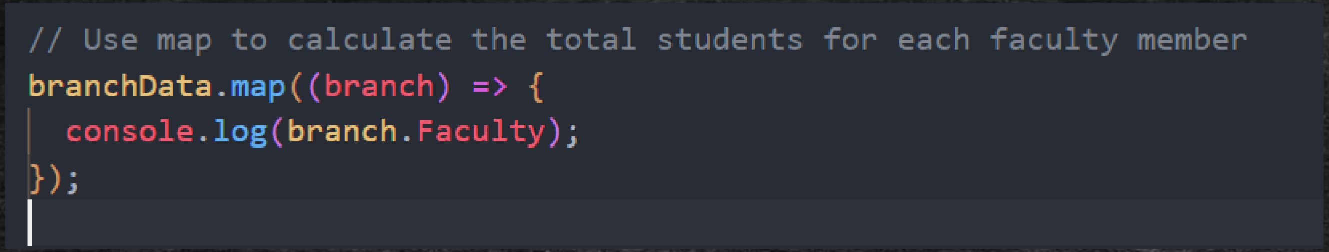
```
const branchData = Object.keys(data).flatMap((program) => {
  return program;
});
console.log(branchData);
```



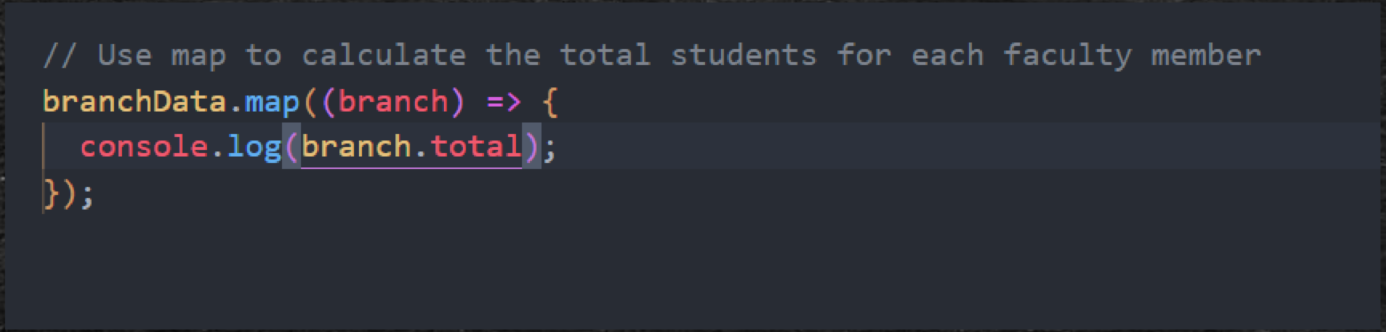
```
// Create an array of branch data
const branchData = Object.values(data).flatMap((program) => {
  return program;
});
console.log(branchData);
```



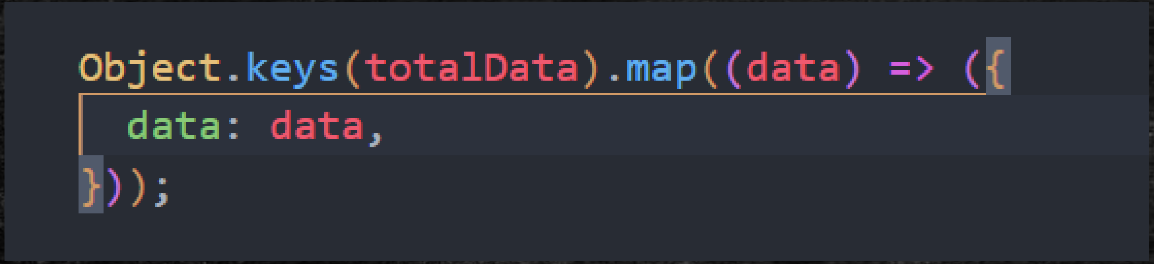
```
// Use map to calculate the total students for each faculty member
branchData.map((branch) => {
  console.log(branch);
});
```



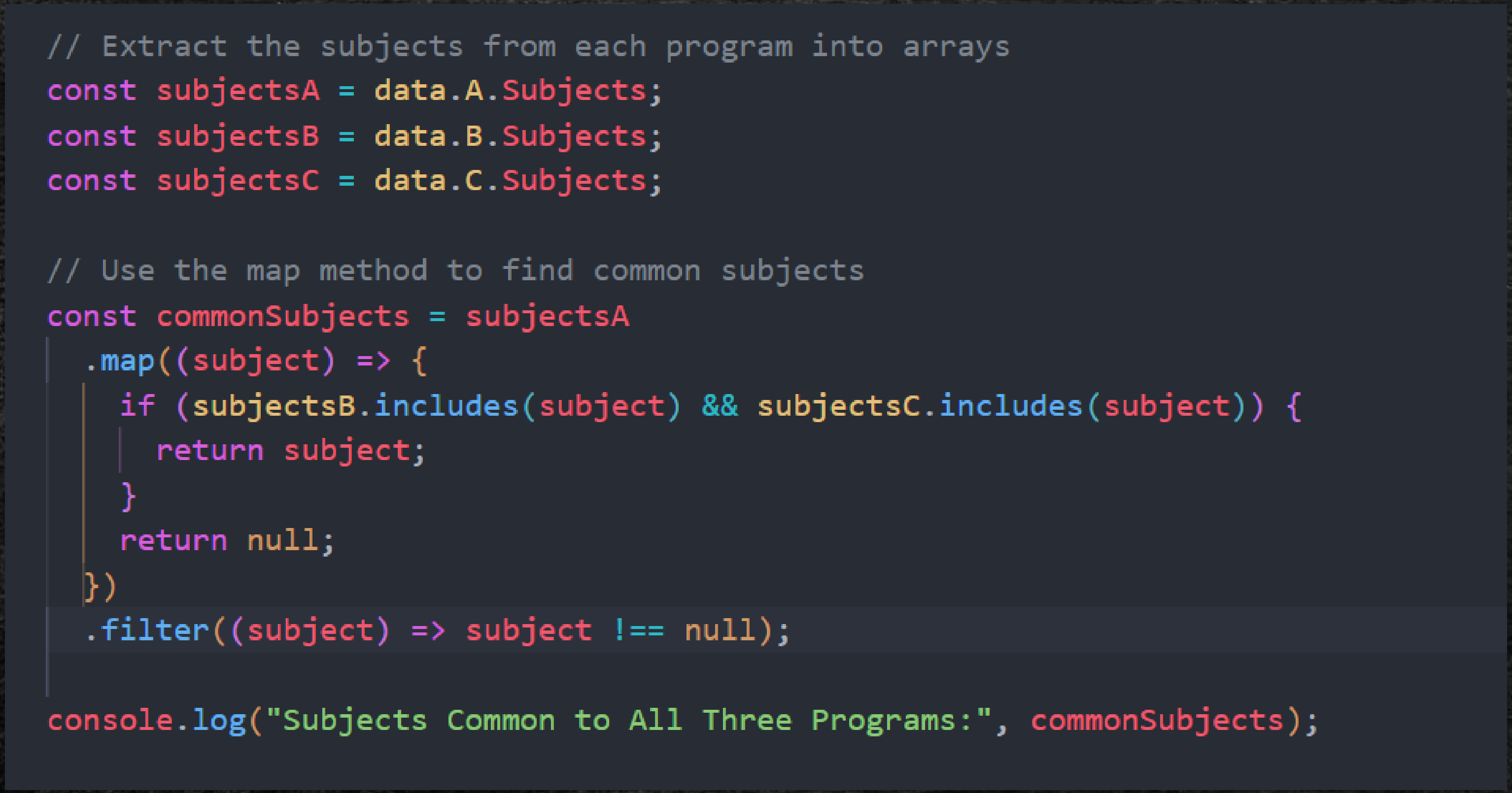
```javascript
// Use map to calculate the total students for each faculty member
branchData.map((branch) => {
  console.log(branch.Faculty);
});
```



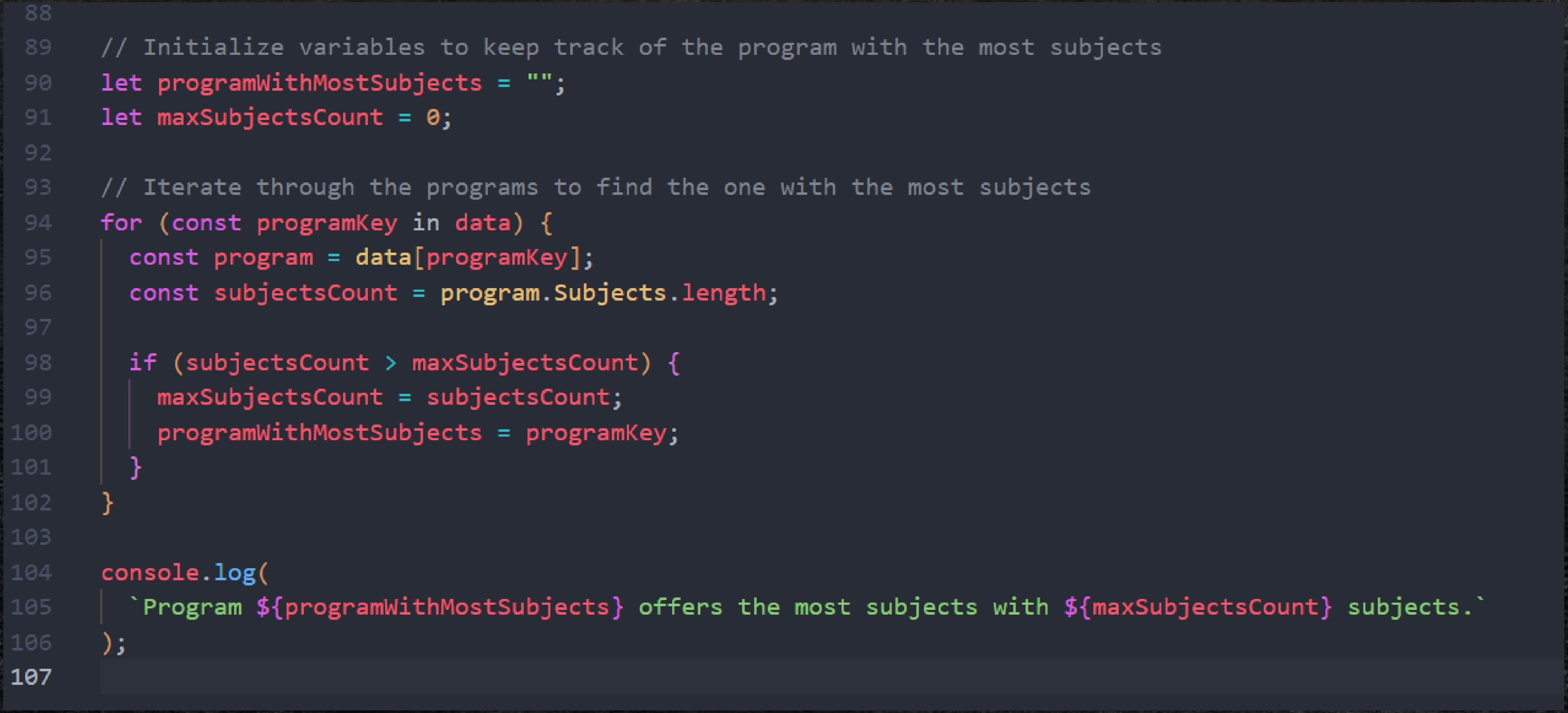
```javascript
// Use map to calculate the total students for each faculty member
branchData.map((branch) => {
  console.log(branch.total);
});
```



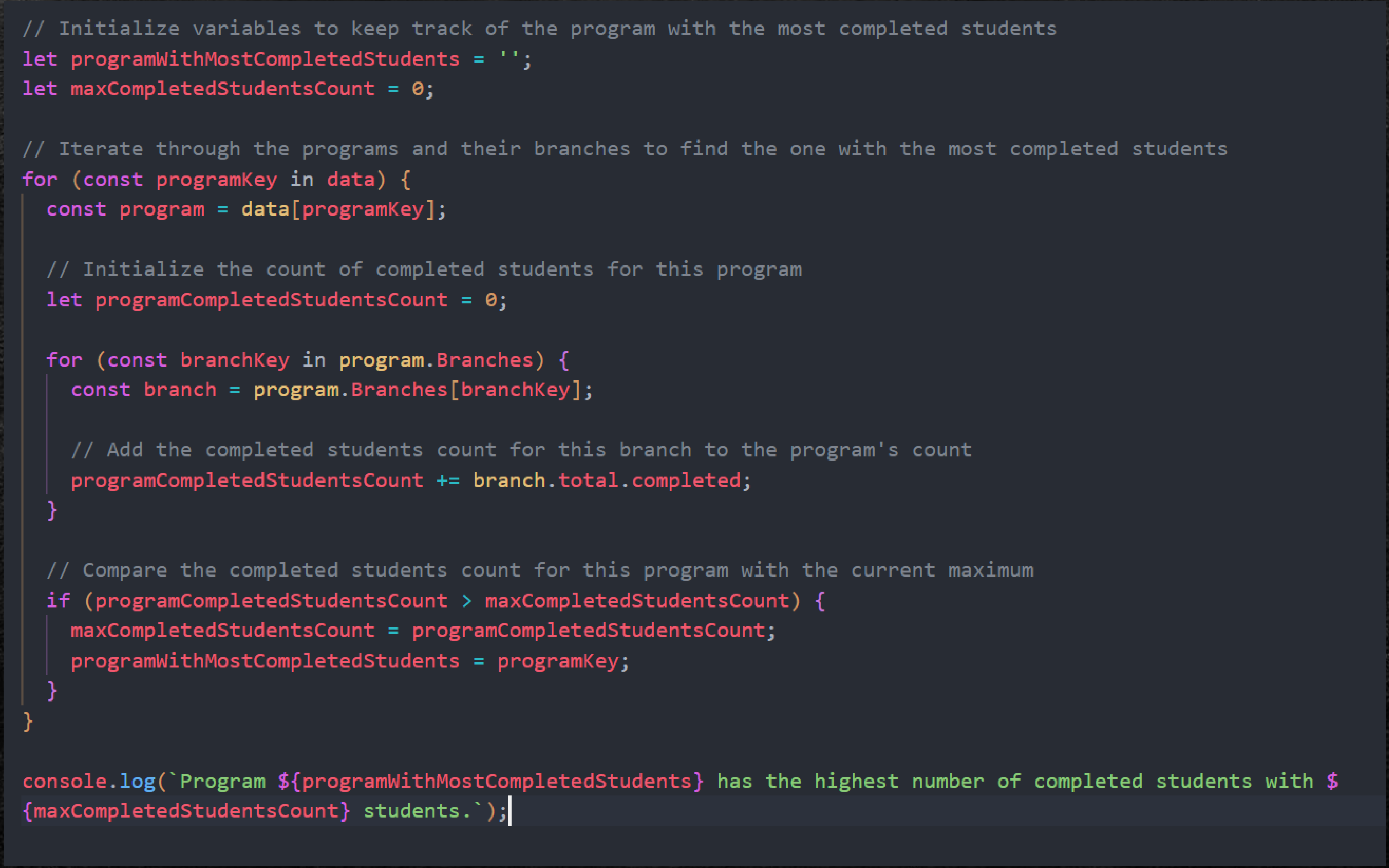
```javascript
Object.keys(totalData).map((data) => ({
  data: data,
}));
```



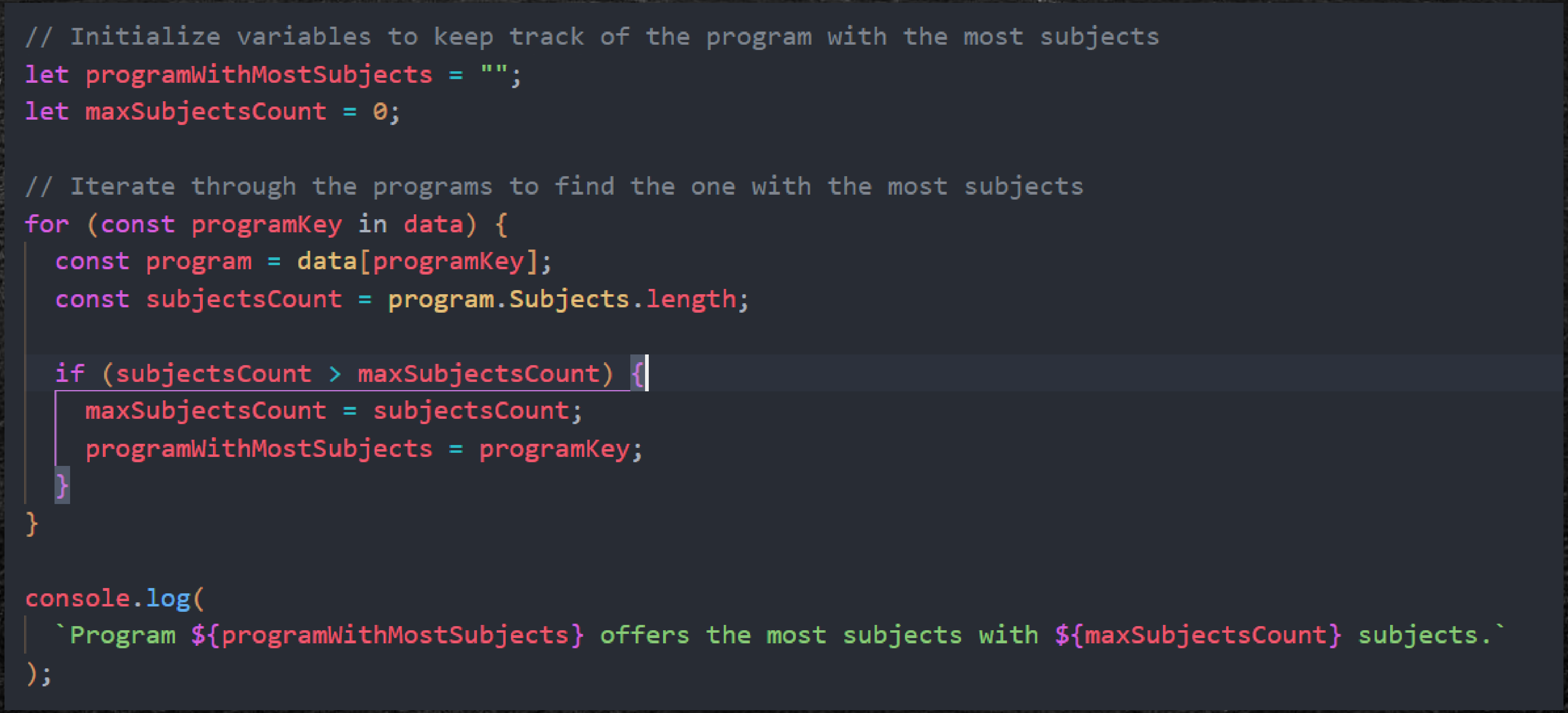
```javascript
// Extract the subjects from each program into arrays
const subjectsA = data.A.Subjects;
const subjectsB = data.B.Subjects;
const subjectsC = data.C.Subjects;

// Use the map method to find common subjects
const commonSubjects = subjectsA
  .map((subject) => {
    if (subjectsB.includes(subject) && subjectsC.includes(subject)) {
      return subject;
    }
    return null;
  })
  .filter((subject) => subject !== null);

console.log("Subjects Common to All Three Programs:", commonSubjects);
```



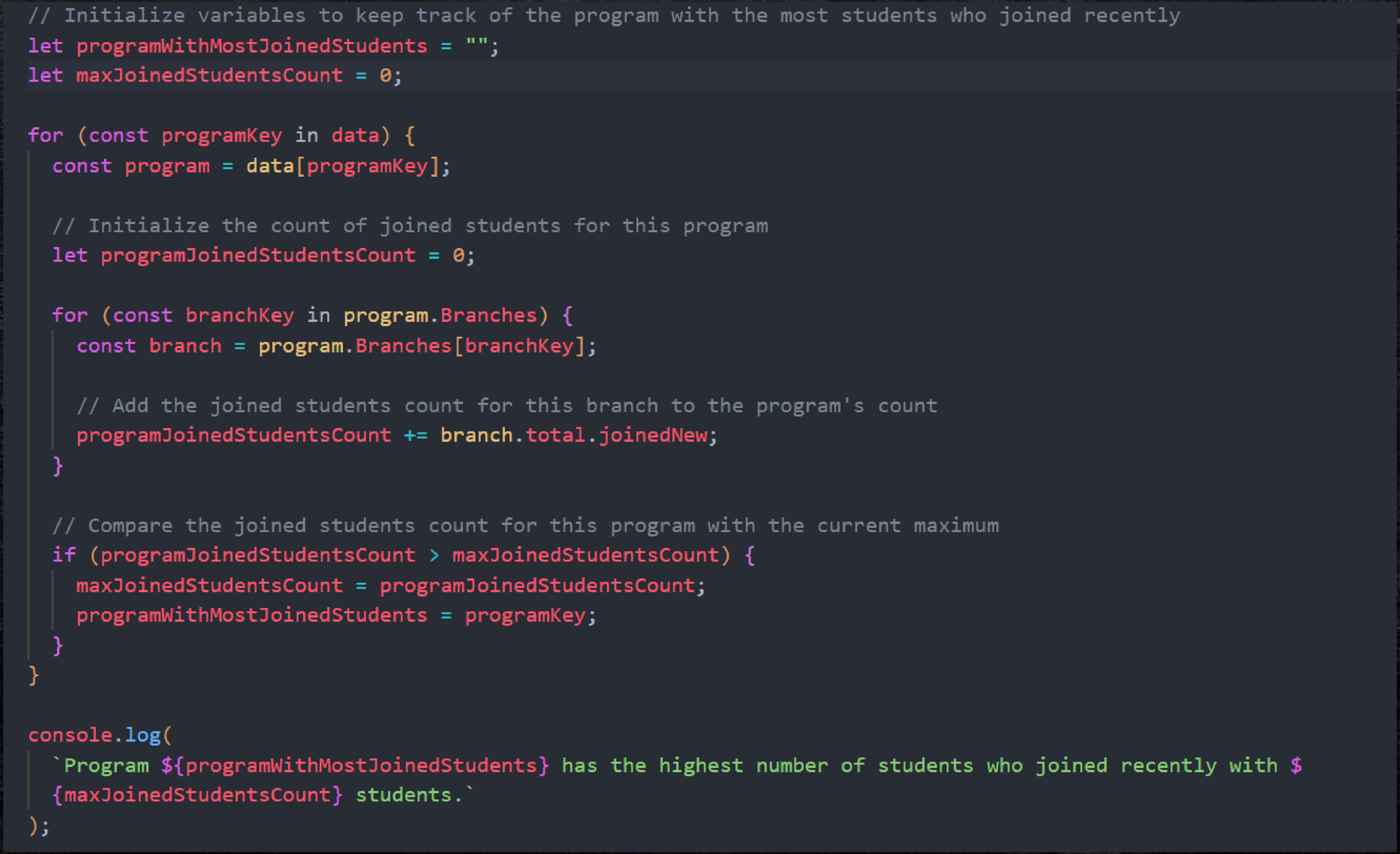
```javascript

// Initialize variables to keep track of the program with the most subjects
let programWithMostSubjects = "";
let maxSubjectsCount = 0;

// Iterate through the programs to find the one with the most subjects
for (const programKey in data) {
  const program = data[programKey];
  const subjectsCount = program.Subjects.length;

  if (subjectsCount > maxSubjectsCount) {
    maxSubjectsCount = subjectsCount;
    programWithMostSubjects = programKey;
  }
}

console.log(
  `Program ${programWithMostSubjects} offers the most subjects with ${maxSubjectsCount} subjects.`
);

```



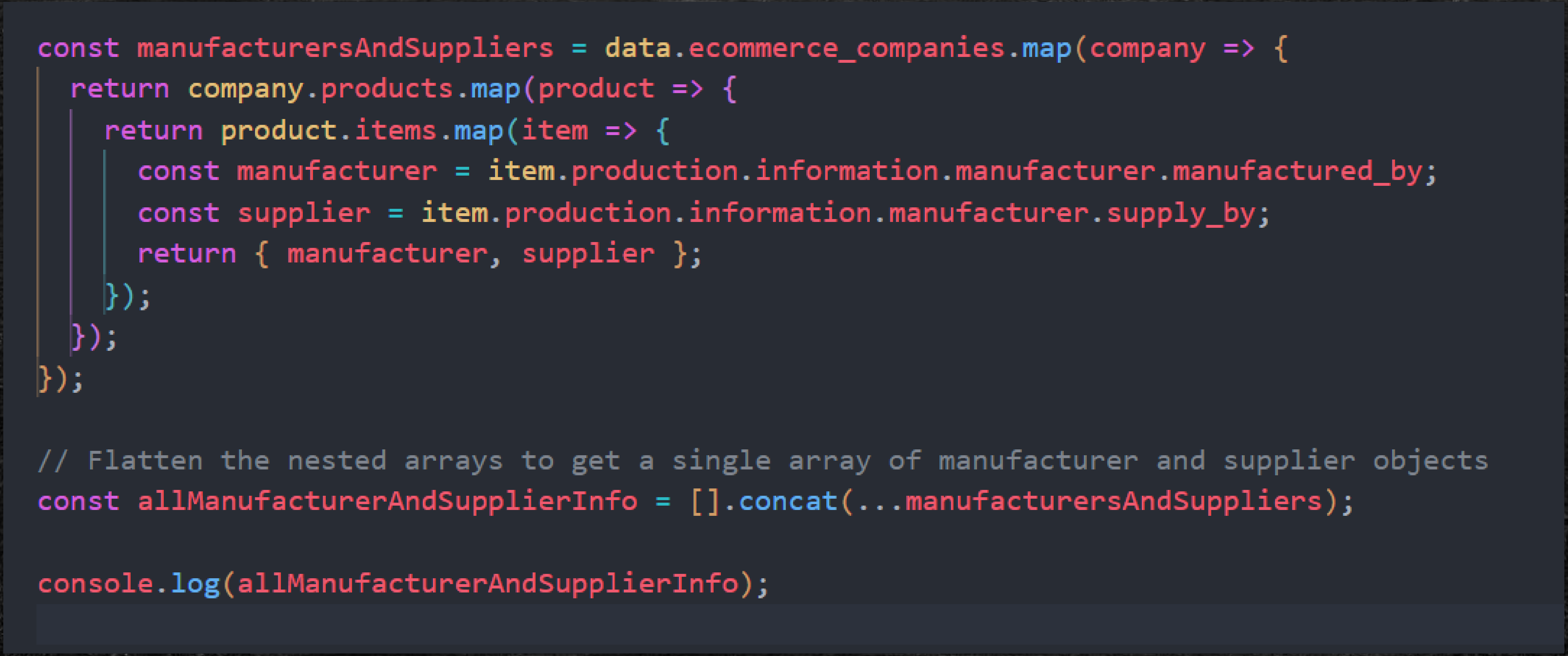
```javascript
// Initialize variables to keep track of the program with the most completed students
let programWithMostCompletedStudents = '';
let maxCompletedStudentsCount = 0;

// Iterate through the programs and their branches to find the one with the most completed students
for (const programKey in data) {
  const program = data[programKey];

  // Initialize the count of completed students for this program
  let programCompletedStudentsCount = 0;

  for (const branchKey in program.Branches) {
    const branch = program.Branches[branchKey];

    // Add the completed students count for this branch to the program's count
    programCompletedStudentsCount += branch.total.completed;
  }

  // Compare the completed students count for this program with the current maximum
  if (programCompletedStudentsCount > maxCompletedStudentsCount) {
    maxCompletedStudentsCount = programCompletedStudentsCount;
    programWithMostCompletedStudents = programKey;
  }
}

console.log(`Program ${programWithMostCompletedStudents} has the highest number of completed students with $
{maxCompletedStudentsCount} students.`);
```



```javascript
// Initialize variables to keep track of the program with the most subjects
let programWithMostSubjects = "";
let maxSubjectsCount = 0;

// Iterate through the programs to find the one with the most subjects
for (const programKey in data) {
  const program = data[programKey];
  const subjectsCount = program.Subjects.length;

  if (subjectsCount > maxSubjectsCount) {
    maxSubjectsCount = subjectsCount;
    programWithMostSubjects = programKey;
  }
}

console.log(
  `Program ${programWithMostSubjects} offers the most subjects with ${maxSubjectsCount} subjects.`
);
```



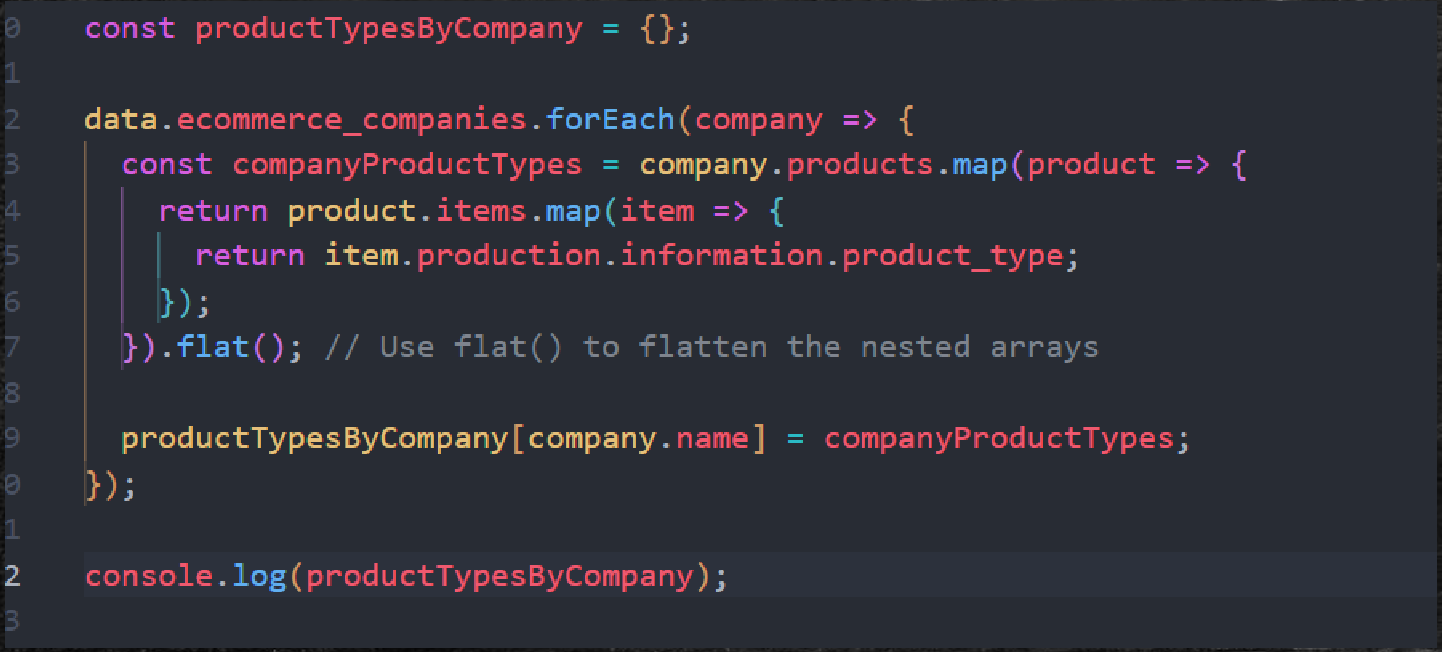
```javascript
// Initialize variables to keep track of the program with the most students who joined recently
let programWithMostJoinedStudents = "";
let maxJoinedStudentsCount = 0;

for (const programKey in data) {
  const program = data[programKey];

  // Initialize the count of joined students for this program
  let programJoinedStudentsCount = 0;

  for (const branchKey in program.Branches) {
    const branch = program.Branches[branchKey];

    // Add the joined students count for this branch to the program's count
    programJoinedStudentsCount += branch.total.joinedNew;
  }

  // Compare the joined students count for this program with the current maximum
  if (programJoinedStudentsCount > maxJoinedStudentsCount) {
    maxJoinedStudentsCount = programJoinedStudentsCount;
    programWithMostJoinedStudents = programKey;
  }
}

console.log(
  `Program ${programWithMostJoinedStudents} has the highest number of students who joined recently with $
  {maxJoinedStudentsCount} students.`
);
```



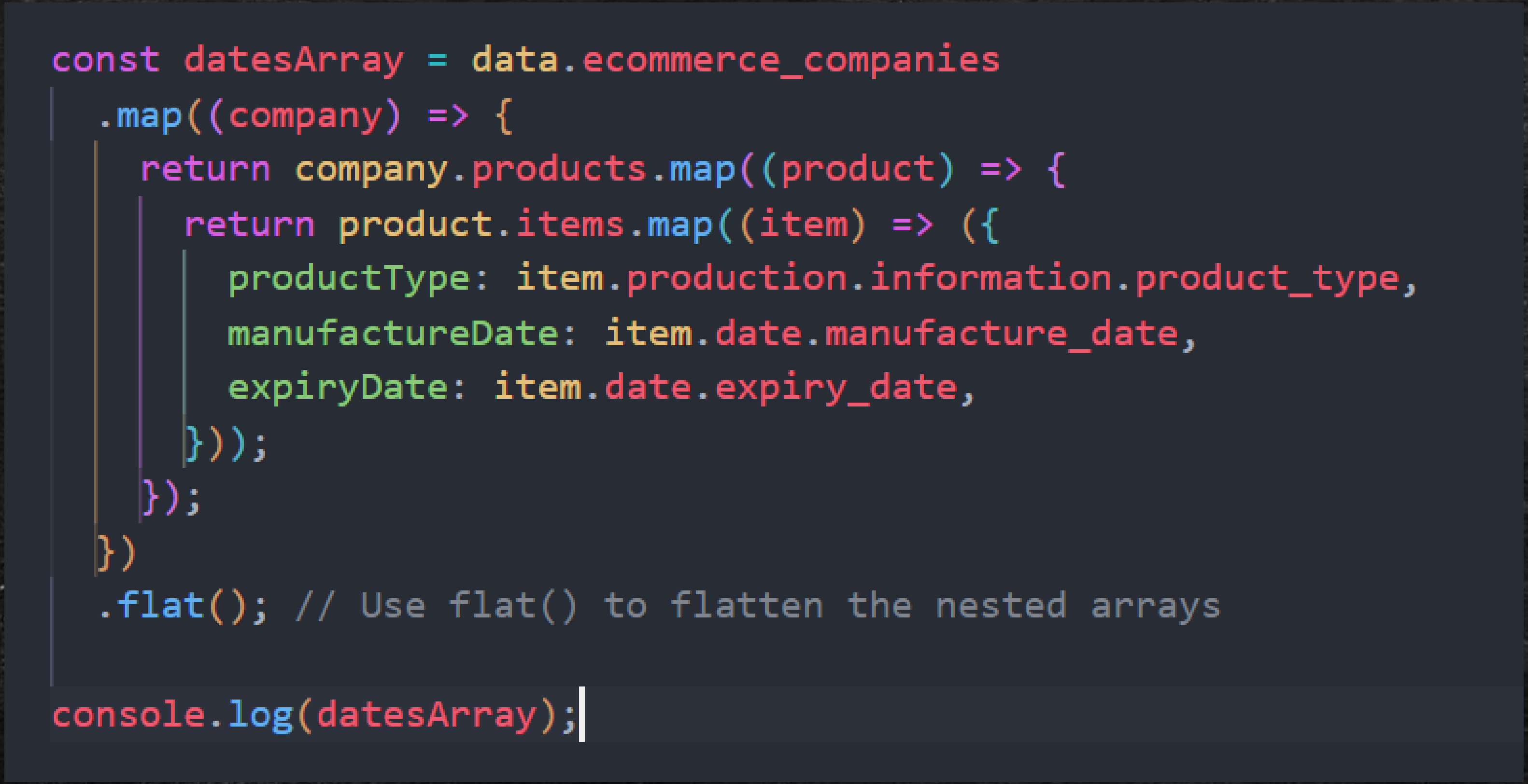
```
const manufacturersAndSuppliers = data.ecommerce_companies.map(company => {
  return company.products.map(product => {
    return product.items.map(item => {
      const manufacturer = item.production.information.manufacturer.manufactured_by;
      const supplier = item.production.information.manufacturer.supply_by;
      return { manufacturer, supplier };
    });
  });
});

// Flatten the nested arrays to get a single array of manufacturer and supplier objects
const allManufacturerAndSupplierInfo = [].concat(...manufacturersAndSuppliers);

console.log(allManufacturerAndSupplierInfo);
```

```
const productTypes = data.ecommerce_companies.map(company => {
  return company.products.map(product => {
    return product.items.map(item => {
      return item.production.information.product_type;
    });
  });
});

// Flatten the nested arrays to get a single array of product types
const allProductTypes = [].concat(...productTypes);

console.log(allProductTypes);
```

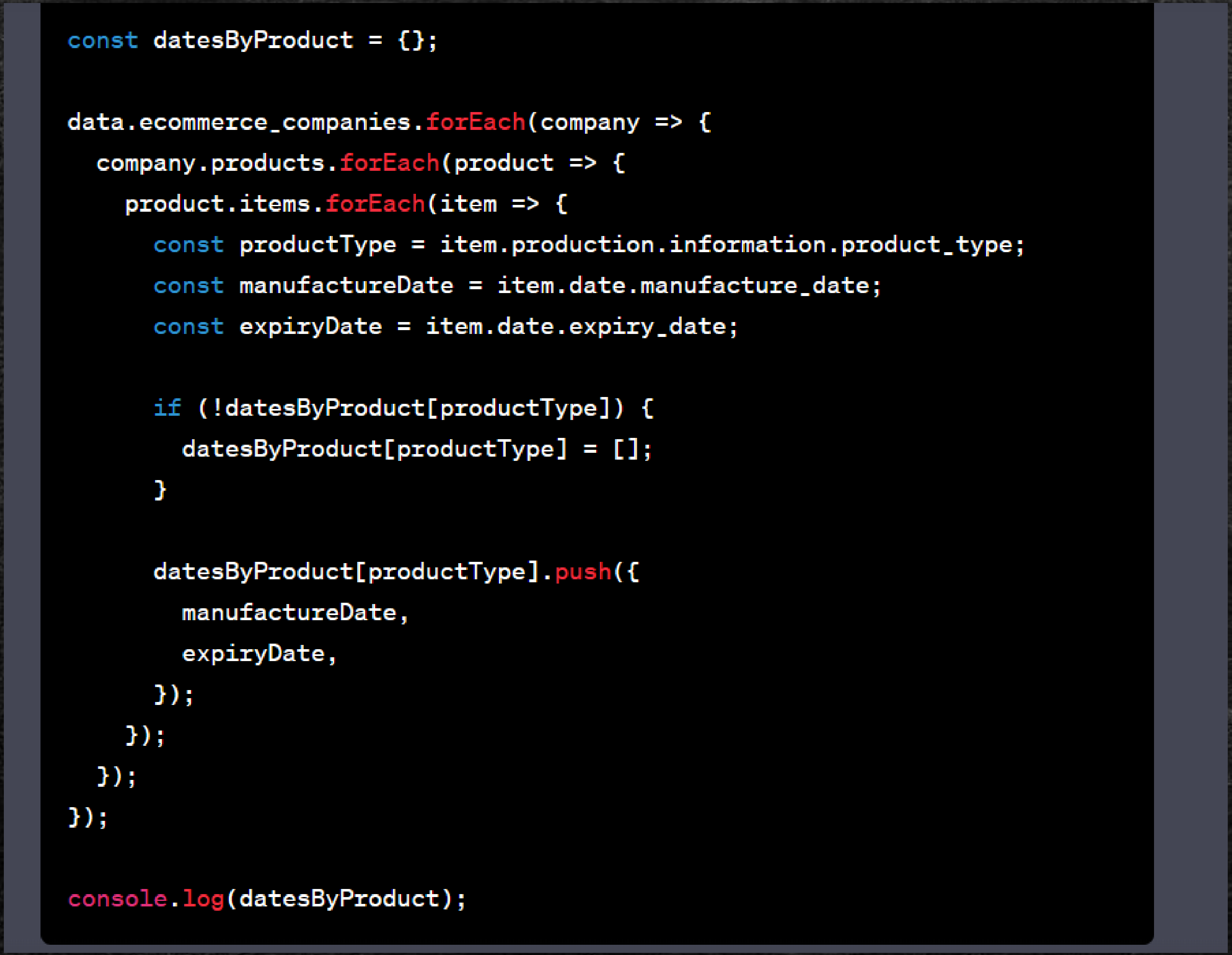
```
const companyInformation = data.ecommerce_companies.map(company => {
  return {
    name: company.name,
    start_date: company.start_date,
    no_of_employees: company.no_of_employees,
    ceo_name: company.ceo_name,
  };
});

console.log(companyInformation);
```



```javascript
const productTypesByCompany = {};

data.ecommerce_companies.forEach(company => {
  const companyProductTypes = company.products.map(product => {
    return product.items.map(item => {
      return item.production.information.product_type;
    });
  }).flat(); // Use flat() to flatten the nested arrays

  productTypesByCompany[company.name] = companyProductTypes;
});

console.log(productTypesByCompany);
```



```
const datesArray = data.ecommerce_companies
  .map((company) => {
    return company.products.map((product) => {
      return product.items.map((item) => ({
        productType: item.production.information.product_type,
        manufactureDate: item.date.manufacture_date,
        expiryDate: item.date.expiry_date,
      }));
    });
  })
  .flat(); // Use flat() to flatten the nested arrays

console.log(datesArray);
```



```
const datesByProduct = {};

data.ecommerce_companies.forEach(company => {
  company.products.forEach(product => {
    product.items.forEach(item => {
      const productType = item.production.information.product_type;
      const manufactureDate = item.date.manufacture_date;
      const expiryDate = item.date.expiry_date;

      if (!datesByProduct[productType]) {
        datesByProduct[productType] = [];
      }

      datesByProduct[productType].push({
        manufactureDate,
        expiryDate,
      });
    });
  });
});

console.log(datesByProduct);
```