



## Ondokuz Mayıs Üniversitesi, Web Programlama Labı

Ad - Soyad: Alper Karaca

Öğrenci Numarası: 21060619

Ödev Adı: Föy - 3

[GitHub Linki](#)

## Özet Bilgi

---

Bu föy içerisinde NodeJs'e giriş yapılmıştır. MySQL kütüphanesi ile CRUD işlemleri, NodeJs kullanarak dosya ve matematiksel işlemler istenmiştir.

# Program Çıktıları

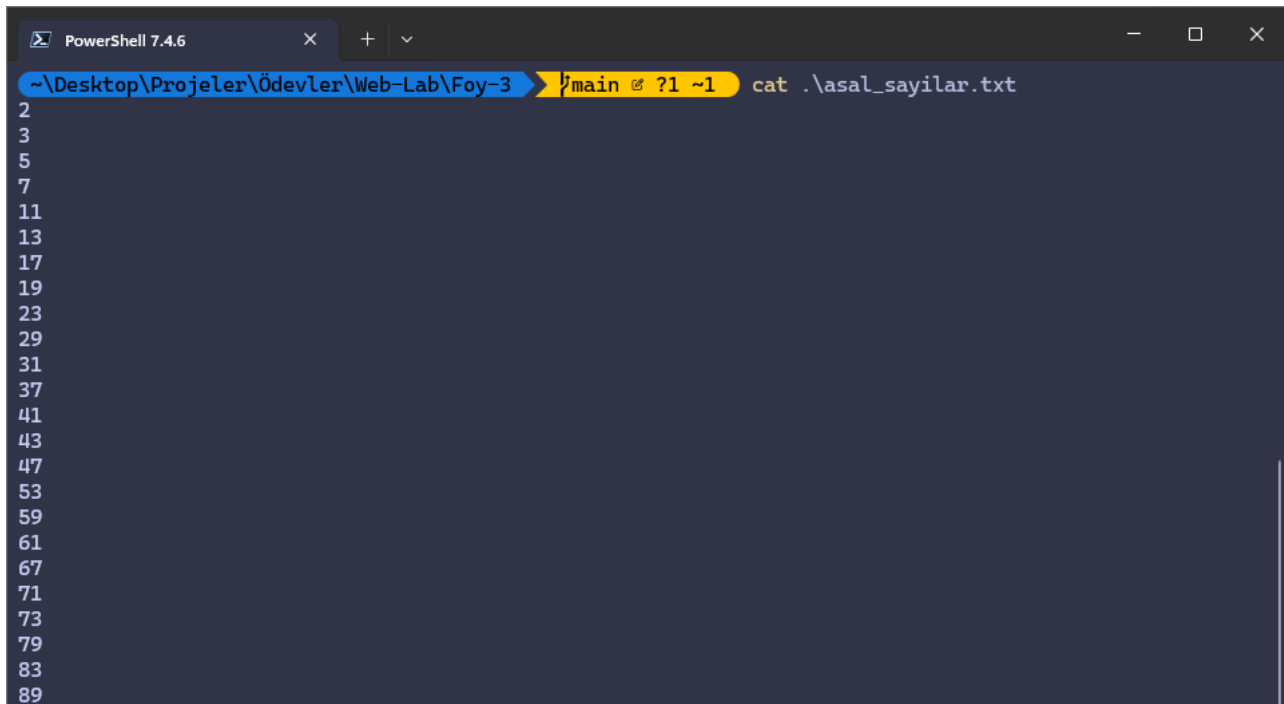
## 1. CRUD İşlemleri

```
PowerShell 7.4.6
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > ./main ?1 ~1 node .\MySQL.js
Veritabanı oluşturuldu
employees Veritabanı seçildi
employee Tablosu oluşturuldu
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Kullanıcı eklendi
Mühendislik bölümündeki çalışanlar sıralanıyor ...
[
  {
    EmployeeID: 3,
    FirstName: 'Roberto',
    LastName: 'Tamburello',
    DepartmentName: 'Engineering'
  },
  {
    EmployeeID: 4,
    FirstName: 'Rob',
    LastName: 'Walters',
    DepartmentName: 'Engineering'
  },
  {
    EmployeeID: 5,
    FirstName: 'Gail',
    LastName: 'Erickson',
    DepartmentName: 'Engineering'
  },
  {
    EmployeeID: 6,
    FirstName: 'Jossef',
    LastName: 'Goldberg',
    DepartmentName: 'Engineering'
  }
]
Terri'nin departmanı Executive olarak güncellendi
Terri'nin bilgileri yazdırılıyor ...
[
  {
    EmployeeID: 2,
    FirstName: 'Terri',
    LastName: 'Duffy',
    DepartmentName: 'Executive'
  }
]
```

## 2. Dosyadan Okunan Sayılarla Polinom İşlemi

```
PowerShell 7.4.6
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > cat .\quiz6.txt
3,5,4,7
4,4,4,4
5,2,1,0
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > node .\quiz6.js
64
84
55
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > |
```

### 3. 90'dan Küçük Asal Olmayan Sayıları Dosyaya Yazma



A PowerShell terminal window titled "PowerShell 7.4.6" with a dark background. The command prompt shows the path `~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3` and the command `cat .\asal_sayilar.txt`. The output of the command is a list of prime numbers from 2 to 89, displayed one per line.

```
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > cat .\asal_sayilar.txt
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
```

```
PowerShell 7.4.6
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 /main of 71 ~1 node .\FillBlankNumbers.js
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 /main of 71 ~1 cat .\tum_sayilar.txt
0
1
4
6
8
9
10
12
14
15
16
18
20
21
22
24
25
26
27
28
30
32
33
34
35
36
38
39
40
42
44
45
46
48
49
50
51
52
54
55
56
57
58
60
62
63
64
65
66
68
69
70
72
74
75
76
77
78
80
81
82
84
85
86
87
88
90
```

## 4. 1 – 100 Aralığındaki Asal Sayıları Dosyaya Yazma

```
PowerShell 7.4.6
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > cat asal.txt
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > node .\prime.js
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > cat .\asal.txt
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
~\Desktop\Projeler\Ödevler\Web-Lab\Foy-3 > |
```

# Kaynak Kodları ve Açıklamalar

## 1. CRUD İşlemleri

```
const sql = require('mysql2'); /*MySQL modülü yerine mysql2 kullanma sebebim güvenlik sebebiyle bağlantıma izin vermemesi örnek hata:
  sqlMessage: 'Client does not support authentication protocol requested by server; consider upgrading MySQL client',
  Çözüm: https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server */
const env = require('dotenv').config();

let connection = sql.createConnection({
  host: process.env.HOST,
  user: process.env.USER,
  password: process.env.PASSWORD,
  port: process.env.PORT
});

const TABLE_NAME = process.env.TABLE_NAME;
const DB_NAME = process.env.DB_NAME;

// HOST, USER, PASSWORD, PORT, TABLE_NAME, DB_NAME gibi bilgileri güvenlik amaçlı .env dosyası içerisinden alıyorum.

const USERS = [
  { FirstName: 'Ken', LastName: 'Sanchez', DepartmentName: 'Executive' },
  { FirstName: 'Terri', LastName: 'Duffy', DepartmentName: 'Engineering' },
  { FirstName: 'Roberto', LastName: 'Tamburello', DepartmentName: 'Engineering' },
  { FirstName: 'Rob', LastName: 'Walters', DepartmentName: 'Engineering' },
  { FirstName: 'Gail', LastName: 'Erickson', DepartmentName: 'Engineering' },
  { FirstName: 'Jossef', LastName: 'Goldberg', DepartmentName: 'Engineering' },
  { FirstName: 'Dylan', LastName: 'Miller', DepartmentName: 'Support' },
  { FirstName: 'Diane', LastName: 'Margheim', DepartmentName: 'Support' },
  { FirstName: 'Gigi', LastName: 'Matthew', DepartmentName: 'Support' },
  { FirstName: 'Michael', LastName: 'Raheem', DepartmentName: 'Support' }
];

connection.query('CREATE DATABASE IF NOT EXISTS ' + DB_NAME, (err) => {
  if (err) {
    console.log(err);
  } else {
    console.log('Veritabanı oluşturuldu');
  }
});
```

Öncelikle föyde belirtilen ‘MySQL’ kütüphanesinin desteği kesildiği için ‘MySQL2’ kütüphanesini kullandım.

Bu föyleri GitHub üzerinde public olarak yayınlamam sebebiyle parola bilgilerinin sızmaması için ortam değişkenlerini kullanmak için ‘dotenv’ kullandım bunların hepsi ‘process.env.\*\*\*’ içeren başlıklarda yer almaktadır.

Bağlanılacak veritabanının adresi, kullanıcı adı, parolası ve port bilgileriyle bağlantı gerçekleştiriyorum.

Burada kullanıcıları teker teker eklememek için bir USERS dizisi içerisinde her bir kullanıcıyı obje olarak tutuyorum böylece döngü içerisine alıp hızlı bir şekilde hepsini veritabanına ekleyebileceğim.

Veritabanına bir QUERY göndererek eğer ki ismi bu veritabanını oluşturmasını sağlıyorum.



```

connection.query('USE ' + DB_NAME, (err) => {
  if (err) {
    console.log(err);
  } else {
    console.log(process.env.DB_NAME + ' Veritabanı seçildi');
  }
});

connection.query('CREATE TABLE IF NOT EXISTS ' + TABLE_NAME +
  ' (EmployeeID INT AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(255), ' +
  'LastName VARCHAR(255) UNIQUE, DepartmentName VARCHAR(255))', (err) => {
  if (err) {
    console.log(err);
  } else {
    console.log(TABLE_NAME + ' Tablosu oluşturuldu');
  }
});

USERS.forEach((user) => {
  connection.query('INSERT IGNORE INTO ' + TABLE_NAME + '(FirstName, LastName, DepartmentName) VALUES ' + `(
    '${user.FirstName}', '${user.LastName}', '${user.DepartmentName}')`, (err) => {
    //Burada IGNORE kullanmanın sebebi UNIQUE Constraintini kullandığım için replika verilerde hata veriyordu bu
    //sebeple hata oluşturan verileri atlamayı sağladım.
    //https://stackoverflow.com/questions/1361340/how-can-i-do-insert-if-not-exists-in-mysql
    if (err) {
      console.log(err);
    } else {
      console.log('Kullanıcı eklendi');
    }
  });
});

```

Veritabanına yeniden bir istek göndererek kullanılacak veritabanını seçiyorum.

Bir sonraki adımda ‘employee’ adına bir table oluşturuyorum burada EmployeeID Primary Key olarak geçiyor. Çalışan isim, soy isim ve departman adını alıyorum. Script her çalıştığı zaman kullanıcıların tekrar eklenip yer kaplamasını önlemek içinse soy isimlerine UNIQUE değerini ekledim bu sayede aynı soy isime sahip kişiler listeye tekrardan eklenmeyecek.

Kullanıcıları veritabanına eklemek içinse oluşturduğum USERS dizisindeki her bir obje için INSERT IGNORE INTO kullanıyorum. Burada IGNORE eklememin sebebi veritabanına aynı soy isime sahip kişiler eklenirken hata verdiği için o kişileri atlamasını sağlıyorum.

```

connection.query('SELECT * FROM ' + TABLE_NAME + ' WHERE DepartmentName = "Engineering"', (err, result) => {
  if (err) {
    console.log(err);
  } else {
    console.log("Mühendislik bölümündeki çalışanlar sıralanıyor ...");
    console.log(result);
  }
});

connection.query('UPDATE ' + TABLE_NAME + ' SET DepartmentName = "Executive" WHERE FirstName = "Terri"', (err) => {
  if(err){
    console.log(err);
  }
  else{
    console.log('Terri\'nin departmanı Executive olarak güncellendi');
  }
});

connection.query('SELECT * FROM ' + TABLE_NAME + ' WHERE FirstName = "Terri"', (err, result) => {
  if (err) {
    console.log(err);
  } else {
    console.log("Terri'nin bilgileri yazdırılıyor ...");
    console.log(result);
  }
});

connection.close = () => {
  connection.end((err) => {
    if (err) {
      console.log(err);
    } else {
      console.log('Bağlantı kapatıldı');
    }
  });
}

```

Sadece mühendislik departmanında çalışanları görmek için employee tablosunda gelecek olan verilere WHERE ile bir şart koyuyorum ve departman isminin 'Engineering' olması şartı ekliyorum.

Tablodaki Terri isimli kişinin departmanını güncellemek için UPDATE – SET komutlarını kullanıyorum böylece sadece ilgili kısım güncelleniyor.

Terri'nin departman adı değişikliğini göstermek için sadece Terri özelinde bir listeleme yapıyorum ve ardından veritabanı bağlantımı close() fonksiyonunu çağırarak kapatıyorum.

## 2. Dosyadan Okunan Sayılarla Polinom İşlemi

```
let math = require('mathjs');
let file = require('fs');

let txt = file.readFileSync('quiz6.txt', 'utf8');

txt = txt.split('\n');

txt.forEach(line => {
  line = line.split(',');
  let x = line[0];
  let a = line[1];
  let b = line[2];
  let c = Number(line[3].replace('\r', ''));
  let res = a * math.pow(x, 2) + b * x + c;
  console.log(res);
});
```

Öncelikle 'mathjs' ve 'fs' kütüphanelerini ekliyorum bu sayede matematiksel fonksiyonlar ve dosya işlemlerini kullanabileceğim.

Ardından okunacak quiz6.txt dosyasını okumak için readFileSync() fonksiyonunu kullanıyorum.

Okunan değeri satır satır ayırıp bir dizi haline getiriyorum.

Dizideki her bir satır için öncelikle virgülleri kullanarak dizilere ayırıyorum böylece ham sayısal değerlere ulaşabiliyoruz.

Devamında x, a, b ve c değerlerini tanımlıyorum – c değerinde ekstra işlem olma sebebi sonunda \r karakterinin yer alması bu sebeple onu temizleyip string olan değeri Number tipine çeviriyorum)

Son olarak ise math kütüphanesinden üs fonksiyonunu da kullanarak verilen polinomu çözüyor ve konsola yazdırıyorum.

### 3. 90'dan Küçük Asal Olmayan Sayıları Dosyaya Yazma

```
let file = require('fs');

let inputFile = file.readFileSync('asal_sayilar.txt', 'utf8');
let outputFile = file.createWriteStream('tum_sayilar.txt', 'utf8');

inputFile = inputFile.split('\n');

for (let i = 0; i ≤ 90; i++) {
  if (!inputFile.includes(i.toString())) {
    outputFile.write(i + '\n');
  }
}
```

Öncelikle dosya işlemleri için 'fs' kütüphanesini uygulamaya ekliyorum.

Burada bir girdi ve çıktı dosyamız olacağı için iki farklı değişken tanımlıyorum ve bunu belirtilen isimlere göre isimlendirilen dosyalar vasıtasıyla sağlıyorum.

Girdi dosyamdan okunan değeri satır satır böldükten sonra bir 90'dan küçük sayılar olması şartıyla for döngüsü başlatıyorum ve eğer şu anda bulunan değer girdi dizisinde yok ise bu sayıyı çıktı dosyasına yazmasını sağlıyorum.

## 1 – 100 Aralığındaki Asal Sayıları Dosyaya Yazma

```
let file = require('fs');
let math = require('mathjs');

let outputFile = file.createWriteStream('asal.txt');

for(i = 1; i ≤ 100; i++){
  if(math.isPrime(i)){
    outputFile.write(i + '\n');
  }
}

/* Eğer Math Kütüphanesini Kullanmamam Gerekiyorsa */

const isPrime = num => {
  for(let i = 2, s = Math.sqrt(num); i ≤ s; i++) {
    if(num % i ≡ 0) return false;
  }
  return num > 1;
}
```

İşlemlere öncelikle matematik ve dosya işlemleri kütüphanelerini ekleyerek başlıyorum.

Asal sayıların yazılacağı dosya için yazma özelliikle dosya akışı oluşturuyorum ve oluşturulacak dosya adını belirtiyorum.

Devamında for döngüsü içerisinde 1'den 100'e kadar olan sayıları teker teker gezip math kütüphanesindeki isPrime - asal mı – fonksiyonundan faydalanarak asallığını sorguluyorum eğer ki sayı asalsa da çıktı dosyasına satır satır eklenmesini sağlıyorum.

NOT: Bu kısımda mathjs kütüphanesine izin verilip verilmemesi durumunda bir bilgilendirilme yapılmadığı için ben direkt olarak tanımlı fonksiyonu kullanmayı tercih ettim ancak kullanmamam gerekliyse aynı işlemleri gören isPrime fonksiyonu da görsel içerisinde yer almaktadır.