

Tutorial – 1

1. Design html page to get below given output using bootstrap css.

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>UI</title>
  <link rel="stylesheet" href="../css/bootstrap.min.css">
  <script src="../js/bootstrap.bundle.min.js"></script>

  <style type="text/css">
    .primary{
      background: #1900ff; color: #1900ff;
    }
    .warning{
      background: #feff00; color: #feff00;
    }
    .success{
      background: #008100; color: #008100;
    }
    .cyan{
      background: #8CF08C; color: #8CF08C;
    }
    .oreng{
      background: #ffa700; color: #ffa700;
    }
    .gray{
      background: #808080; color: #808080;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="row mt-4">
      <div class="mr-2 col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6">
        <div class="primary text-center">6</div>
      </div>
      <div class="ml-2 col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6">
        <div class="primary text-center">6</div>
      </div>
    </div>
  </div>
```

```
</div>
<div class="row mt-2">
  <div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">
    <div class="warning p-2 text-center">4</div>
  </div>
  <div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">
    <div class="warning p-2 text-center">4</div>
  </div>
  <div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">
    <div class="warning p-2 text-center">4</div>
  </div>
</div>
<div class="row mt-2">
  <div class="mr-2 col-8 col-sm-8 col-md-8 col-lg-8 col-xl-8">
    <div class="success p-2 text-center">8</div>
  </div>
  <div class="mr-2 col-4 col-sm-4 col-md-4 col-lg-4 col-xl-4">
    <div class="cyan p-2 text-center">4</div>
  </div>
</div>
<div class="row mt-2">
  <div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">
    <div class="oreng p-2 text-center">3</div>
  </div>
  <div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">
    <div class="oreng p-2 text-center">3</div>
  </div>
  <div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">
    <div class="oreng p-2 text-center">3</div>
  </div>
  <div class="mr-2 col-3 col-sm-3 col-md-3 col-lg-3 col-xl-3">
    <div class="oreng p-2 text-center">3</div>
  </div>
</div>
<div class="row mt-2">
  <div class="mr-2 col-12 col-sm-12 col-md-12 col-lg-12 col-xl-12">
    <div class="gray text-center">12</div>
  </div>
</div>
</div>
</body>
</html>
```

- CODE :**

3

```
<th scope="col">Email</th>
</tr>
</thead>
<tbody>
<tr class="table-success">
<td>John</td>
<td>Doe</td>
<td>john@example.com</td>
</tr>
<tr class="table-danger">
<td>Mary</td>
<td>Moe</td>
<td>mary@example.cpm</td>
</tr>
<tr class="table-warning">
<td>July</td>
<td>Dooley</td>
<td>july@example.com</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```

::OUTPUT::

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.cpm
July	Dooley	july@example.com

3. Bootstrapping with Buttons.

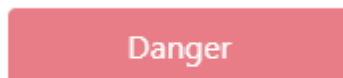
Use a Bootstrap class to style the button properly with a red color.

- b. Change the size of the buttons in the following order: large, medium, small and xsmall.
- c. Make the button span the entire width of the parent element.
- d. Use a Bootstrap class to disable the button.

CODE :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>UI</title>
    <link rel="stylesheet" href="../css/bootstrap.min.css">
    <script src="../js/bootstrap.bundle.min.js"></script>
  </head>
  <body>
    <div class="container">
      <div class="mt-4 text-center">
        <button type="button" class="col-12 col-sm-4 col-md-4 col-lg-6 col-xl-6 btn btn-danger btn-
        block" disabled>Danger</button>
      </div>
    </div>
  </body>
</html>
```

::OUTPUT::



4. Style the below given html form using bootstrap to get the output shown below.

CODE :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Exercise #6: Simple form</title>
  <link rel="stylesheet" href="../css/bootstrap.min.css">
  <script src="../js/bootstrap.bundle.min.js"></script>

  <style type="text/css">
    .border{
      border-color: #E2E2E2;
      border-bottom-left-radius: 5px;
      border-bottom-right-radius: 5px;
      border-bottom-width: 5px;
      border-top-left-radius: 5px;
      border-top-right-radius: 5px;
    }
    .font-weight{
      font-weight: bold;
    }
    .margin-right{
      margin-right: 5px;
    }
  </style>
</head>
<body>
  <form class="form col-12 p-4" action="#" >
    <div class="mt-2">
      <label class="font-weight col-12" for="first_name">First name:</label>
      <input class="mt-1 col-12 border p-1" type="text" name="first_name" id="first_name"/>
    </div>
    <div class="mt-2">
      <label class="font-weight col-12" for="last_name">Last name:</label>
      <input class="mt-1 col-12 border p-1" type="text" name="last_name" id="last_name"/>
    </div>
    <div class="mt-2">
      <label><input class="margin-right" type="radio" name="gender" value="male"/>male</label>
      <label><input class="margin-right" type="radio" name="gender"
value="female"/>female</label>
    </div>
```

```
<div class="mt-2">
  <label class="font-weight col-12" for="birth_date">Date of birth:</label>
  <input class="mt-1 col-12 border p-1" type="date" name="birth_date" id="birth_date"/>
</div>
<div class="mt-3">
  <input class="btn pl-2 pr-2 btn-primary text-center" type="submit" value="Add"/>
</div>
</form>
</body>
</html>
```


::OUTPUT::

First name:

Last name:

☐ male ☐ female

Date of birth:

Add

Tutorial-2

```
C:\Users\bharg>mongoimport --db restaurant --collection res --file E:\restaurants.json
2022-01-28T16:10:40.236+0530   connected to: mongod://localhost/
2022-01-28T16:10:40.792+0530   3772 document(s) imported successfully. 0 document(s) failed to import.
```

```
> show dbs
Students      0.000GB
admin          0.000GB
config         0.000GB
local          0.000GB
reg            0.000GB
restaurant     0.001GB
```

```
> use restaurant
switched to db restaurant
> show collections
res
```

5. Write a MongoDB query to display all the documents in the collection restaurants.

CODE : AND : OUTPUT :

```
> db.res.find().pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305c88"),
  "address" : {
    "building" : "351",
    "coord" : [
      -73.98513559999999,
      40.7676919
    ],
    "street" : "West 57 Street",
    "zipcode" : "10019"
  },
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "grades" : [
    {
      "date" : ISODate("2014-09-06T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-07-22T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    },
    {
      "date" : ISODate("2012-07-31T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2011-12-29T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    }
  ],
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
```


6. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

CODE : AND : OUTPUT :

```
> db.res.find({}, {restaurant_id:1, name:1, borough:1, cuisine:1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305c88"),
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "_id" : ObjectId("61f3c82886916f00ad305c89"),
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "_id" : ObjectId("61f3c82886916f00ad305c8a"),
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "_id" : ObjectId("61f3c82886916f00ad305c8b"),
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "_id" : ObjectId("61f3c82886916f00ad305c8c"),
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
```

7. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant.

CODE : AND : OUTPUT :

```
> db.res.find({}, {restaurant_id:1, name:1, borough:1, cuisine:1, _id:0}).pretty()
{
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
```

8. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.

CODE : AND : OUTPUT :

```
> db.res.find({}, {restaurant_id:1, name:1, borough:1, "address.zipcode":1, _id:0}).pretty()
{
  "address" : {
    "zipcode" : "10019"
  },
  "borough" : "Manhattan",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "address" : {
    "zipcode" : "11225"
  },
  "borough" : "Brooklyn",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "address" : {
    "zipcode" : "11224"
  },
  "borough" : "Brooklyn",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "address" : {
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
```

9. Write a MongoDB query to display all the restaurant which is in the borough Bronx.

CODE : AND : OUTPUT :

```
> db.res.find({borough:"Bronx"}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305c8b"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2011-11-23T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2011-03-10T00:00:00Z"),
      "grade" : "B",
      "score" : 14
    }
  ]
}
```

10. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

CODE : AND : OUTPUT :

```
> db.res.find({borough:"Bronx"}).limit(5).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305c8b"),
  "address" : {
    "building" : "1007",
    "coord" : [
      -73.856077,
      40.848447
    ],
    "street" : "Morris Park Ave",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2014-03-03T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-09-11T00:00:00Z"),
      "grade" : "A",
      "score" : 6
    },
    {
      "date" : ISODate("2013-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    },
    {
      "date" : ISODate("2011-11-23T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2011-03-10T00:00:00Z"),
      "grade" : "B",
      "score" : 14
    }
  ]
}
```

11. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

CODE : AND : OUTPUT :

```
> db.res.find({borough:"Bronx"}).limit(5).skip(5).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305cc4"),
  "address" : {
    "building" : "658",
    "coord" : [
      -73.81363999999999,
      40.82941100000001
    ],
    "street" : "Clarence Ave",
    "zipcode" : "10465"
  },
  "borough" : "Bronx",
  "cuisine" : "American ",
  "grades" : [
    {
      "date" : ISODate("2014-06-21T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2012-07-11T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }
  ],
  "name" : "Manhem Club",
  "restaurant_id" : "40364363"
},
{
  "_id" : ObjectId("61f3c82886916f00ad305cde"),
  "address" : {
    "building" : "2222",
    "coord" : [
      -73.84971759999999,
      40.8304811
    ],
    "street" : "Haviland Avenue",
    "zipcode" : "10462"
  },
  "borough" : "Bronx",
  "cuisine" : "American ",
  "grades" : [
    {
      "date" : ISODate("2014-06-21T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2012-07-11T00:00:00Z"),
      "grade" : "A",
      "score" : 10
    }
  ],
  "name" : "Manhem Club",
  "restaurant_id" : "40364363"
}
```

12. Write a MongoDB query to find the restaurants who achieved a score more than 90.

CODE : AND : OUTPUT :

```
> db.res.find({grades:{$elemMatch:{score:{$gt:90}}}},{"grades.$":1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305de5"),
  "grades" : [
    {
      "date" : ISODate("2014-03-28T00:00:00Z"),
      "grade" : "C",
      "score" : 131
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305e87"),
  "grades" : [
    {
      "date" : ISODate("2012-04-06T00:00:00Z"),
      "grade" : "C",
      "score" : 92
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305fe9"),
  "grades" : [
    {
      "date" : ISODate("2014-06-17T00:00:00Z"),
      "grade" : "C",
      "score" : 98
    }
  ]
}
>
```

13. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

CODE : AND : OUTPUT :

```
> db.res.find({grades:{$elemMatch:{score:{$gt:80,$lt:100}}}},{"grades.$":1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305e87"),
  "grades" : [
    {
      "date" : ISODate("2012-04-06T00:00:00Z"),
      "grade" : "C",
      "score" : 92
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305fe9"),
  "grades" : [
    {
      "date" : ISODate("2014-06-17T00:00:00Z"),
      "grade" : "C",
      "score" : 98
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad306855"),
  "grades" : [
    {
      "date" : ISODate("2014-06-27T00:00:00Z"),
      "grade" : "C",
      "score" : 89
    }
  ]
}
>
```


14. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

CODE : AND : OUTPUT :

```
> db.res.find({"address.coord.1":{"$lt:-25.754168}}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305d70"),
  "address" : {
    "building" : "155157",
    "coord" : [
      153.1628795,
      -28.0168595
    ],
    "street" : "Christie St.",
    "zipcode" : "10002"
  },
  "borough" : "Manhattan",
  "cuisine" : "Steak",
  "grades" : [
    {
      "date" : ISODate("2014-11-12T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2013-09-24T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2013-04-12T00:00:00Z"),
      "grade" : "B",
      "score" : 26
    },
    {
      "date" : ISODate("2012-09-21T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2012-04-10T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    }
  ],
  "name" : "Sammy'S Steakhouse",
  "restaurant_id" : "40368552"
}
```

15. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

CODE : AND : OUTPUT :

```
> db.res.find({$and:[{cuisine:{$ne:"American"}},{grades:{$elemMatch:{score:{$gt:70}}}],{"address.coord.1":{$lt:40.7308729}}},{cuisine:1,"grades.$":1,"address.coord":1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305e87"),
  "address" : {
    "coord" : [
      -73.9864626,
      40.7266739
    ]
  },
  "cuisine" : "Indian",
  "grades" : [
    {
      "date" : ISODate("2012-04-06T00:00:00Z"),
      "grade" : "C",
      "score" : 92
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad306145"),
  "address" : {
    "coord" : [
      -73.94610279999999,
      40.7137587
    ]
  },
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2011-05-03T00:00:00Z"),
      "grade" : "C",
      "score" : 77
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad306855"),
  "address" : {
    "coord" : [
      -74.0163793,
      40.7167671
    ]
  },
  "cuisine" : "Bakery",
  "grades" : [
    {
      "date" : ISODate("2011-05-03T00:00:00Z"),
      "grade" : "C",
      "score" : 77
    }
  ]
}
```

16. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168. Note : Do this query without using \$and operator.

CODE : AND : OUTPUT :

```
> db.res.find({cuisine:{$ne:"American"},grades:{$elemMatch:{score:{$gt:70}}},"address.coord.0":{$lt:-65.754168}},{cuisine:1,"grades.$":1,"address.coord":1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305de5"),
  "address" : {
    "coord" : [
      -73.9782725,
      40.7624022
    ]
  },
  "cuisine" : "American ",
  "grades" : [
    {
      "date" : ISODate("2014-03-28T00:00:00Z"),
      "grade" : "C",
      "score" : 131
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305e87"),
  "address" : {
    "coord" : [
      -73.9864626,
      40.7266739
    ]
  },
  "cuisine" : "Indian",
  "grades" : [
    {
      "date" : ISODate("2012-04-06T00:00:00Z"),
      "grade" : "C",
      "score" : 92
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305f15"),
  "address" : {
    "coord" : [
      -73.9883909,
      40.740735
    ]
  },
  "cuisine" : "American ",
```

17. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

CODE : AND : OUTPUT :

```
> db.res.find({cuisine:{$ne:"American"},grades:{$elemMatch:{grade:"A"}},borough:{$ne:"Brooklyn"}},{cuisine:1,"grades.$":1,borough:1}).sort({cuisine:-1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad306393"),
  "borough" : "Manhattan",
  "cuisine" : "Vietnamese/Cambodian/Malaysia",
  "grades" : [
    {
      "date" : ISODate("2014-08-21T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad30644d"),
  "borough" : "Queens",
  "cuisine" : "Vietnamese/Cambodian/Malaysia",
  "grades" : [
    {
      "date" : ISODate("2013-05-20T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad306836"),
  "borough" : "Manhattan",
  "cuisine" : "Vietnamese/Cambodian/Malaysia",
  "grades" : [
    {
      "date" : ISODate("2014-10-01T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    }
  ]
}
{
  "_id" : ObjectId("61f3c82886916f00ad305f37"),
  "borough" : "Manhattan",
  "cuisine" : "Vegetarian",
  "grades" : [
    {
      "date" : ISODate("2014-02-05T00:00:00Z"),
      "grade" : "A",
```

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

CODE : AND : OUTPUT :

```
> db.res.find({name:{$regex:/^Wil/}}, {restaurant_id:1,name:1,cuisine:1,borough:1,
id:0}).pretty()
{
  "borough" : "Bronx",
  "cuisine" : "American ",
  "name" : "Wild Asia",
  "restaurant_id" : "40357217"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Delicatessen",
  "name" : "Wilken'S Fine Food",
  "restaurant_id" : "40356483"
}
{
  "borough" : "Bronx",
  "cuisine" : "Pizza",
  "name" : "Wilbel Pizza",
  "restaurant_id" : "40871979"
}
```

19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

CODE : AND : OUTPUT :

```
> db.res.find({name:{$regex:/ces$/}},{restaurant_id:1,name:1,cuisine:1,borough:1,
id:0}).pretty()
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Pieces",
  "restaurant_id" : "40399910"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "S.M.R Restaurant Services",
  "restaurant_id" : "40403857"
}
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Good Shepherd Services",
  "restaurant_id" : "40403989"
}
{
  "borough" : "Queens",
  "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
  "name" : "The Ice Box-Ralph'S Famous Italian Ices",
  "restaurant_id" : "40690899"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Jewish/Kosher",
  "name" : "Alices",
  "restaurant_id" : "40782042"
}
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Re: Sources",
  "restaurant_id" : "40876068"
}
```

20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

CODE : AND : OUTPUT :

```
> db.res.find({name:{$regex:/Reg/}}, {restaurant_id:1,name:1,cuisine:1,borough:1,_id:0}).pretty()
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Regina Caterers",
  "restaurant_id" : "40356649"
}
{
  "borough" : "Manhattan",
  "cuisine" : "Café/Coffee/Tea",
  "name" : "Caffe Reggio",
  "restaurant_id" : "40369418"
}
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Regency Hotel",
  "restaurant_id" : "40382679"
}
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Regency Whist Club",
  "restaurant_id" : "40402377"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "Rego Park Cafe",
  "restaurant_id" : "40523342"
}
{
  "borough" : "Queens",
  "cuisine" : "Pizza",
  "name" : "Regina Pizza",
  "restaurant_id" : "40801325"
}
{
  "borough" : "Manhattan",
  "cuisine" : "American ",
  "name" : "Regal Entertainment Group",
  "restaurant_id" : "40891782"
}
```

21. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish

CODE : AND : OUTPUT :

```
> db.res.find({borough:"Bronx",cuisine:{$in:["American","Chinese"]},{restaurant_
id:1,name:1,cuisine:1,borough:1,_id:0}).pretty()
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Happy Garden",
  "restaurant_id" : "40363289"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Happy Garden",
  "restaurant_id" : "40364296"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "China Wok Ii",
  "restaurant_id" : "40510328"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Dragon City",
  "restaurant_id" : "40529203"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Hunan Balcony",
  "restaurant_id" : "40551996"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Great Wall Restaurant",
  "restaurant_id" : "40552226"
}
{
  "borough" : "Bronx",
  "cuisine" : "Chinese",
  "name" : "Lucky House Restaurant",
  "restaurant_id" : "40571587"
}
{
}
```


22. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.

CODE : AND : OUTPUT :

```
> db.res.find({borough:{$in:["Staten Island","Queens","Bronx","Brooklyn"]}}, {restaurant_id:1,name:1,cuisine:1,borough:1,_id:0}).pretty()
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "Brunos On The Boulevard",
  "restaurant_id" : "40356151"
}
{
  "borough" : "Queens",
  "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
  "name" : "Carvel Ice Cream",
  "restaurant_id" : "40361322"
}
{
  "borough" : "Queens",
  "cuisine" : "Delicatessen",
  "name" : "Sal'S Deli",
  "restaurant_id" : "40361618"
}
{
  "borough" : "Queens",
  "cuisine" : "Delicatessen",
  "name" : "Steve Chu'S Deli & Grocery",
  "restaurant_id" : "40361998"
}
{
  "borough" : "Queens",
  "cuisine" : "Chinese",
  "name" : "Ho Mei Restaurant",
  "restaurant_id" : "40362432"
}
{
  "borough" : "Queens",
  "cuisine" : "Delicatessen",
  "name" : "Tony'S Deli",
  "restaurant_id" : "40363333"
}
```

23. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronx or Brooklyn.

CODE : AND : OUTPUT :

```
> db.res.find({borough:{$nin:["Staten Island","Queens","Bronx or Brooklyn"]}},{restaurant_id:1,name:1,cuisine:1,borough:1,_id:0}).pretty()
{
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Regina Caterers",
  "restaurant_id" : "40356649"
}
{
  "borough" : "Bronx",
  "cuisine" : "American ",
  "name" : "Wild Asia",
  "restaurant_id" : "40357217"
}
```

24. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

CODE : AND : OUTPUT :

```
> db.res.find({$or:[{$and:[{cuisine:{$ne:"American"}},{cuisine:{$ne:"Chinese"}}]},
{name:{$regex:/^Wil/}}]}, {restaurant_id:1,name:1,cuisine:1,borough:1,_id:0}).pretty()
{
  "borough" : "Manhattan",
  "cuisine" : "Irish",
  "name" : "Dj Reynolds Pub And Restaurant",
  "restaurant_id" : "30191841"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "Hamburgers",
  "name" : "Wendy'S",
  "restaurant_id" : "30112340"
}
{
  "borough" : "Brooklyn",
  "cuisine" : "American ",
  "name" : "Riviera Caterer",
  "restaurant_id" : "40356018"
}
{
  "borough" : "Bronx",
  "cuisine" : "Bakery",
  "name" : "Morris Park Bake Shop",
  "restaurant_id" : "30075445"
}
{
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
{
  "borough" : "Queens",
  "cuisine" : "Jewish/Kosher",
  "name" : "Tov Kosher Kitchen",
  "restaurant_id" : "40356068"
}
{
  "borough" : "Queens",
  "cuisine" : "American ",
  "name" : "Brunos On The Boulevard",
}
```

25. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns

CODE : AND : OUTPUT :

```
> db.res.find().sort({name:1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad306917"),
  "address" : {
    "building" : "129",
    "coord" : [
      -73.962943,
      40.685007
    ],
    "street" : "Gates Avenue",
    "zipcode" : "11238"
  },
  "borough" : "Brooklyn",
  "cuisine" : "Italian",
  "grades" : [
    {
      "date" : ISODate("2014-03-06T00:00:00Z"),
      "grade" : "A",
      "score" : 5
    },
    {
      "date" : ISODate("2013-08-29T00:00:00Z"),
      "grade" : "A",
      "score" : 2
    },
    {
      "date" : ISODate("2013-03-08T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2012-06-27T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2011-11-17T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    }
  ],
  "name" : "(Lewis Drug Store) Locanda Vini E Olii",
  "restaurant_id" : "40804423"
}
```

26. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

CODE : AND : OUTPUT :

```
> db.res.find().sort({name:-1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305d47"),
  "address" : {
    "building" : "6946",
    "coord" : [
      -73.8811834,
      40.7017759
    ],
    "street" : "Myrtle Avenue",
    "zipcode" : "11385"
  },
  "borough" : "Queens",
  "cuisine" : "German",
  "grades" : [
    {
      "date" : ISODate("2014-09-24T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    },
    {
      "date" : ISODate("2014-04-17T00:00:00Z"),
      "grade" : "A",
      "score" : 7
    },
    {
      "date" : ISODate("2013-03-12T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    },
    {
      "date" : ISODate("2012-10-02T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2012-05-09T00:00:00Z"),
      "grade" : "A",
      "score" : 13
    },
    {
      "date" : ISODate("2011-12-28T00:00:00Z"),
      "grade" : "B",
      "score" : 24
    }
  ]
}
```

27. Write a MongoDB query to arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order

CODE : AND : OUTPUT :

```
> db.res.find().sort({suisine:1,borough:-1}).pretty()
{
  "_id" : ObjectId("61f3c82886916f00ad305c8c"),
  "address" : {
    "building" : "2206",
    "coord" : [
      -74.1377286,
      40.6119572
    ],
    "street" : "Victory Boulevard",
    "zipcode" : "10314"
  },
  "borough" : "Staten Island",
  "cuisine" : "Jewish/Kosher",
  "grades" : [
    {
      "date" : ISODate("2014-10-06T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    },
    {
      "date" : ISODate("2014-05-20T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2013-04-04T00:00:00Z"),
      "grade" : "A",
      "score" : 12
    },
    {
      "date" : ISODate("2012-01-24T00:00:00Z"),
      "grade" : "A",
      "score" : 9
    }
  ],
  "name" : "Kosher Island",
  "restaurant_id" : "40356442"
}
```

28. Find out how many times each cuisine is offered at various restaurants.

CODE : AND : OUTPUT :

```
> db.res.aggregate([{$group:{_id:"$cuisine","offered to":{$sum:1}}}]).pretty()
{ "_id" : "Caribbean", "offered to" : 75 }
{ "_id" : "Greek", "offered to" : 25 }
{ "_id" : "Czech", "offered to" : 1 }
{ "_id" : "Pancakes/Waffles", "offered to" : 7 }
{ "_id" : "Jewish/Kosher", "offered to" : 60 }
{ "_id" : "Delicatessen", "offered to" : 78 }
{ "_id" : "English", "offered to" : 3 }
{ "_id" : "Bangladeshi", "offered to" : 1 }
{ "_id" : "Steak", "offered to" : 21 }
{ "_id" : "Barbecue", "offered to" : 8 }
{ "_id" : "Hamburgers", "offered to" : 159 }
{ "_id" : "Italian", "offered to" : 325 }
{ "_id" : "Soups & Sandwiches", "offered to" : 8 }
{ "_id" : "Russian", "offered to" : 13 }
{ "_id" : "Seafood", "offered to" : 35 }
{ "_id" : "American ", "offered to" : 1255 }
{ "_id" : "Pizza/Italian", "offered to" : 106 }
{ "_id" : "Hotdogs", "offered to" : 4 }
{ "_id" : "Eastern European", "offered to" : 7 }
{ "_id" : "Spanish", "offered to" : 42 }
Type "it" for more
```

29. Find out how many times each cuisine is offered at various restaurants in descending order.

CODE : AND : OUTPUT :

```
> db.res.aggregate([{$group:{_id:"$cuisine","offered to":{$sum:1}}},{ $sort:{_id:-1}}]).pretty()
{ "_id" : "Vietnamese/Cambodian/Malaysia", "offered to" : 4 }
{ "_id" : "Vegetarian", "offered to" : 10 }
{ "_id" : "Turkish", "offered to" : 11 }
{ "_id" : "Thai", "offered to" : 14 }
{ "_id" : "Tex-Mex", "offered to" : 13 }
{ "_id" : "Tapas", "offered to" : 4 }
{ "_id" : "Steak", "offered to" : 21 }
{ "_id" : "Spanish", "offered to" : 42 }
{ "_id" : "Soups & Sandwiches", "offered to" : 8 }
{ "_id" : "Soul Food", "offered to" : 6 }
{ "_id" : "Seafood", "offered to" : 35 }
{ "_id" : "Sandwiches/Salads/Mixed Buffet", "offered to" : 19 }
{ "_id" : "Sandwiches", "offered to" : 19 }
{ "_id" : "Salads", "offered to" : 1 }
{ "_id" : "Russian", "offered to" : 13 }
{ "_id" : "Portuguese", "offered to" : 2 }
{ "_id" : "Polish", "offered to" : 10 }
{ "_id" : "Pizza/Italian", "offered to" : 106 }
{ "_id" : "Pizza", "offered to" : 270 }
{ "_id" : "Peruvian", "offered to" : 2 }
Type "it" for more
```

30. Which cuisine is highly offered among all restaurants?

CODE : AND : OUTPUT :

```
> db.res.aggregate([{$group:{_id:"$cuisine","offered to":{$sum:1}}},{ $sort:{"offered to":-1}},{ $limit:1}]).pretty()
{ "_id" : "Caribbean", "offered to" : 75 }
```

31. Find out the top 5 highly offered cuisines among all restaurants?

CODE : AND : OUTPUT :

```
> db.res.aggregate([{$group:{_id:"$cuisine","offered to":{$sum:1}}},{ $sort:{"offered to":-1}},{ $limit:5}]).pretty()
{ "_id" : "Pancakes/Waffles", "offered to" : 7 }
{ "_id" : "Jewish/Kosher", "offered to" : 60 }
{ "_id" : "Greek", "offered to" : 25 }
{ "_id" : "Caribbean", "offered to" : 75 }
{ "_id" : "Czech", "offered to" : 1 }
```


Tutorial – 3 - CRUD with Nodejs

1. Create and Emit a custom event that checks whether the age of the person is greater than 18 or not depending on the date of birth passed to the event.

CODE ::

```
var events = require('events')
var em = new events.EventEmitter();
em.on('ageEvent', function(data) {
    var dob = new Date(data);
    var mn = Date.now() - dob.getTime();
    var age_dt = new Date(mn);
    var year = age_dt.getUTCFullYear();
    var age = Math.abs(year - 1997);

    if (age >= 18) {
        console.log("Eligible...");
    } else {
        console.log("Not Eligible...");
    }
});
em.emit('ageEvent', "09-02-2002");
```

:: OUTPUT ::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Tutorial-3 CRUD with Nodejs> nodemon custome_event.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node custome_event.js`
Not Eligible...
[nodemon] clean exit - waiting for changes before restart
```

2. Create a node script that gets the parameters using the GET method from the form.html file and log it on the console.

CODE ::

from_get.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Get API</title>
</head>

<body>
  <form method="get" action="http://localhost:3000/submit">
    Name : <input type="text" name="uname"><br>
    Branch : <input type="text" name="branch"><br>
    <input type="submit" name="submit" value="SUBMIT">
  </form>
</body>

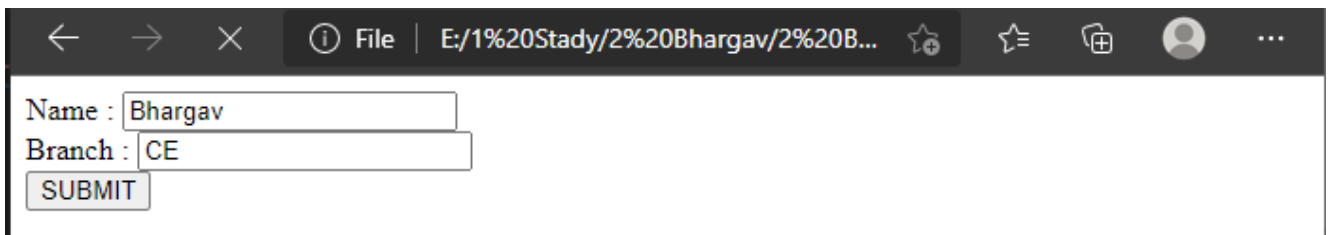
</html>
```

get.js

```
var http = require('http')
var url = require('url')
var querystring = require('querystring')
```

```
var server = http.createServer(function(req, res) {  
    query = url.parse(req.url).query;  
    uname = querystring.parse(query)['uname'];  
    branch = querystring.parse(query)['branch'];  
  
    console.log('Name : ', uname);  
    console.log('Branch : ', branch);  
}).listen(3000);
```

:: OUTPUT ::



Name :

Branch :

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Tutorial-3 CRUD with Nodejs> nodemon get  
[nodemon] 2.0.15  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node get.js`  
Name : undefined  
Branch : undefined  
Name : Bhargav  
Branch : CE
```

3. Create a node script that gets the parameters using POST method from the form.html file and log it on console.

CODE ::

→**form_post :**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>POST API</title>
```

```
</head>
```

```
<body>
```

```
  <form method="post" action="http://localhost:3000/submit">
```

```
    Name : <input type="text" name="uname"><br>
```

```
    Branch : <input type="text" name="branch"><br>
```

```
    <input type="submit" name="submit" value="SUBMIT">
```

```
  </form>
```

```
</body>
```

```
</html>
```

→**post.js ::**

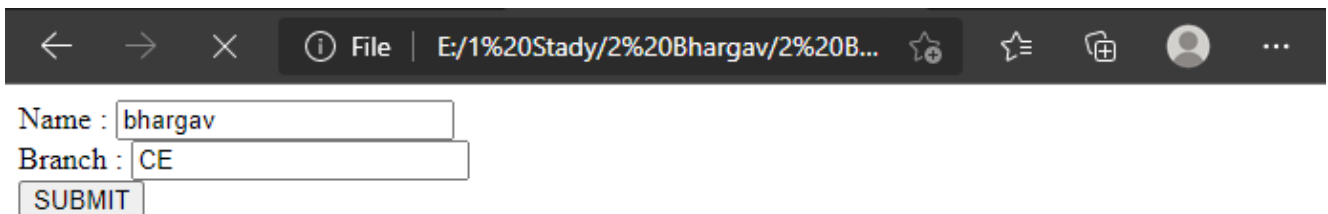
```
var http = require('http')
```

```
var querystring = require('querystring')
```

```
var server = http.createServer(function(req, res) {
```

```
var dt = "";  
req.on('data', function(chunk) {  
    dt += chunk  
})  
  
req.on('end', function() {  
    un = querystring.parse(dt)['uname'];  
    var br = querystring.parse(dt)['branch'];  
  
    console.log('Name : ', un);  
    console.log('Branch : ', br);  
})  
}).listen(3000)
```

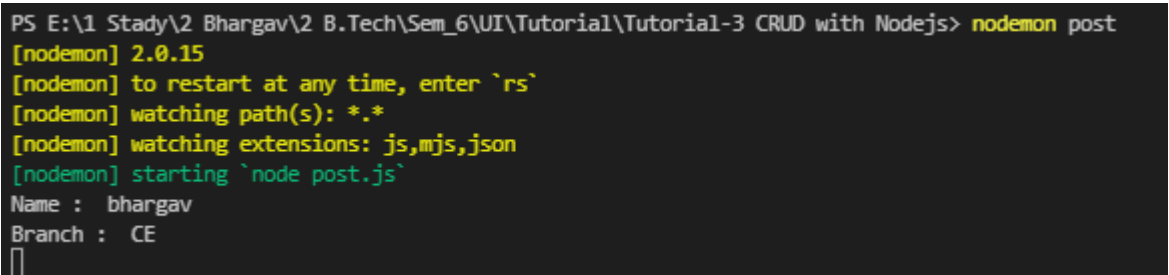
:: OUTPUT ::



← → × ⓘ File | E:/1%20Stady/2%20Bhargav/2%20B... ☆ ☆ ⌕ 👤 ...

Name :

Branch :



```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Tutorial-3 CRUD with Nodejs> nodemon post  
[nodemon] 2.0.15  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,json  
[nodemon] starting `node post.js`  
Name : bhargav  
Branch : CE  
█
```

4. Create mongodb for students and nodejs that connects to mongodb using mongojs module.(install monojs and nodemon packages).

[Student document must contain: s_id,s_name, s_branch, s_city, s_mobilenos, s_add]

CODE ::

```
var mongoose = require('mongoose')
var express = require('express')

mongoose.connect('mongodb://127.0.0.1:27017/students').then(()=>{
  console.log('Connection Success...');
  var app = express();
  app.listen(3000, ()=>{
    console.log('Server Connected...');
  })
}).catch((e)=>{
  console.log('Connection Error...');
});
```

:: OUTPUT ::

5. Modify the above created script to insert static data in student db.

CODE ::

:: OUTPUT ::

6. Create a nodejs that fetches all the documents from student db and logs it on the console.

CODE ::

:: OUTPUT ::

7. Modify the program 6, in order to save records from the form.html file.

CODE ::

:: OUTPUT ::

8. Create a nodejs script to update student records based on the student id.

CODE ::

:: OUTPUT ::

9. Create a noejs script to delete student records based on student id.

CODE ::

:: OUTPUT ::

Tutorial – 04 (Creating APIS using Mongoose)

➔package.json

```
{
  "name": "demoapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index",
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.1",
    "express": "^4.17.2",
    "mongoose": "^6.2.0",
    "nodemon": "^2.0.15"
  }
}
```

➔index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>Hosted Successfully..!!</h1> </body>
</html>
```


➔ **index.js**

```
var mongoose = require("mongoose")
var express = require("express")
var route = require('./routes')
var bodyParser = require('body-parser')
var app = express()

mongoose.connect("mongodb://127.0.0.1:27017/SPElectronics").then(() => {

    console.log("Router Connected...!!")

    app.use(bodyParser.urlencoded({
        extended: false
    }))
    app.use('/api', route)

    app.get('/', (req, res) => {
        res.sendFile('index.html', {
            root: __dirname
        })
    })

    app.listen((process.env.PORT || 3000), () => {
        console.log('Server Started Successfully...!!')
    })
}).catch((e) => {
    console.log(e.toString())
})
```

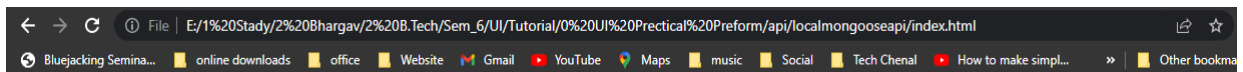
➔ **Models ➔ Accessorie.js**

```
var mongoose = require('mongoose');

var accessoriesSchema = mongoose.Schema({
    name: String,
    price: String
})

module.exports = mongoose.model("accessories", accessoriesSchema)
```

```
PS E:\1 Study\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Practical Preform\api\localmongooseapi> nodemon index
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Router Connected...!!
Server Started Successfully...!!
```



Hosted Successfully...!!

32. Create a NodeAPI to insert records into products database using mongoose library.

CODE ::

```
var express = require('express');
var router = express.Router();
var Accessorie = require('./Models/Accessorie')

//to insert data to the SPElectronics data base in accessories table
router.post("/accessories", async (req, res) => {

    const accessorie = new Accessorie({
        name: req.body.name,
        price: req.body.price
    })

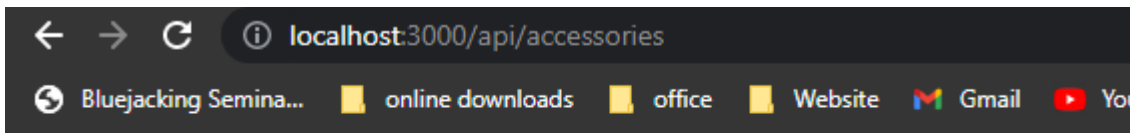
    await accessorie.save((err, msg) => {
        if (err) {
            res.status(500).json({
                "error": err
            })
        } else {
```

```

    res.status(200).json({
      "My-message": msg
    })
  }
})
module.exports = router;

```

:: OUTPUT ::



[]

POST http://localhost:3000/api/accessories **Send**

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	apple			
<input checked="" type="checkbox"/>	price	69999			
	Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 200 OK Time: 80 ms Size: 323 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "My-message": {
3     "name": "apple",
4     "price": "69999",
5     "_id": "62066cd21ea6bd7573cd70d2",
6     "_v": 0
7   }
8 }

```

33. Create a NodeAPI to get products from the database using mongoose library.

CODE ::

```
var express = require('express');

var router = express.Router();

var Accessorie = require('./Models/Accessorie')

//to fetch data from the SPElectronics data base in accessories table

router.get('/accessories', async (req, res) => {

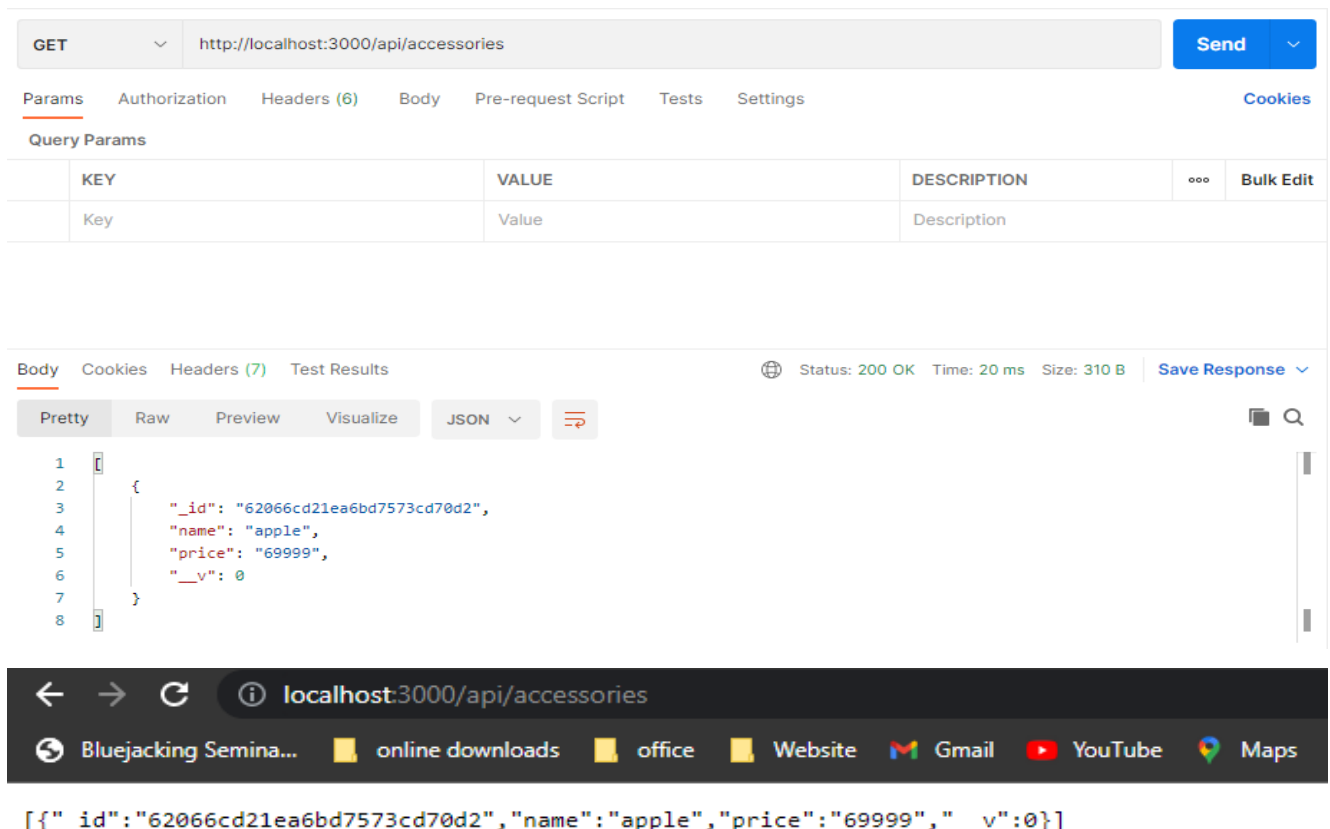
  const accessorie = await Accessorie.find()

  res.send(accessorie)

})

module.exports = router;
```

:: OUTPUT ::



The screenshot shows a REST client interface with a GET request to `http://localhost:3000/api/accessories`. The response is a JSON object representing an accessory.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
  "_id": "62066cd21ea6bd7573cd70d2",
  "name": "apple",
  "price": "69999",
  "__v": 0
}
```

The browser's address bar shows the URL `localhost:3000/api/accessories` and the response is displayed in the console as `[{"_id": "62066cd21ea6bd7573cd70d2", "name": "apple", "price": "69999", "__v": 0}]`.

34. Create a NodeAPI to update products from the database using mongoose library.

CODE ::

```
var express = require('express');
var router = express.Router();
var Accessorie = require('./Models/Accessorie')

// to update data for the SPElectronics data base in accessories table
router.patch('/accessories/:id', async (req, res) => {
  const accessorie = await Accessorie.findOne({
    _id: req.params.id
  })
  accessorie.name = req.body.name
  accessorie.price = req.body.price
  await accessorie.save((err, msg) => {
    if (err) {
      res.status(500).json({
        error: err
      })
    } else {
      res.status(200).json({
        msg: msg
      })
    }
  })
})
module.exports = router;
```

:: OUTPUT ::

```
\\|\\|
\\|\\
\\
```

PATCH ⌵
http://localhost:3000/api/accessories/62066cd21ea6bd7573cd70d2
Send ⌵

Params
Authorization
Headers (8)
Body ●
Pre-request Script
Tests
Settings
Cookies

● none
● form-data
● x-www-form-urlencoded
● raw
● binary
● GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	blackberry			
<input checked="" type="checkbox"/>	price	5000			
	Key	Value	Description		

Body
Cookies
Headers (7)
Test Results
⌚ Status: 200 OK Time: 28 ms Size: 320 B Save Response ⌵

Pretty
Raw
Preview
Visualize
JSON ⌵
≡

```

1 {
2   "msg": {
3     "_id": "62066cd21ea6bd7573cd70d2",
4     "name": "blackberry",
5     "price": "5000",
6     "__v": 0
7   }
8 }
```

⬅ ➡ ↺
localhost:3000/api/accessories

Bluejacking Semina...
online downloads
office
Website
Gmail
YouTube
Maps

```
[{"_id":"62066cd21ea6bd7573cd70d2","name":"blackberry","price":"5000","__v":0}]
```

35. Create a NodeAPI to delete products from the database using mongoose library.

CODE ::

```
var express = require('express');
var router = express.Router();
var Accessorie = require('./Models/Accessorie')
```

//to delete data for the SPElectronics data base in accessories table

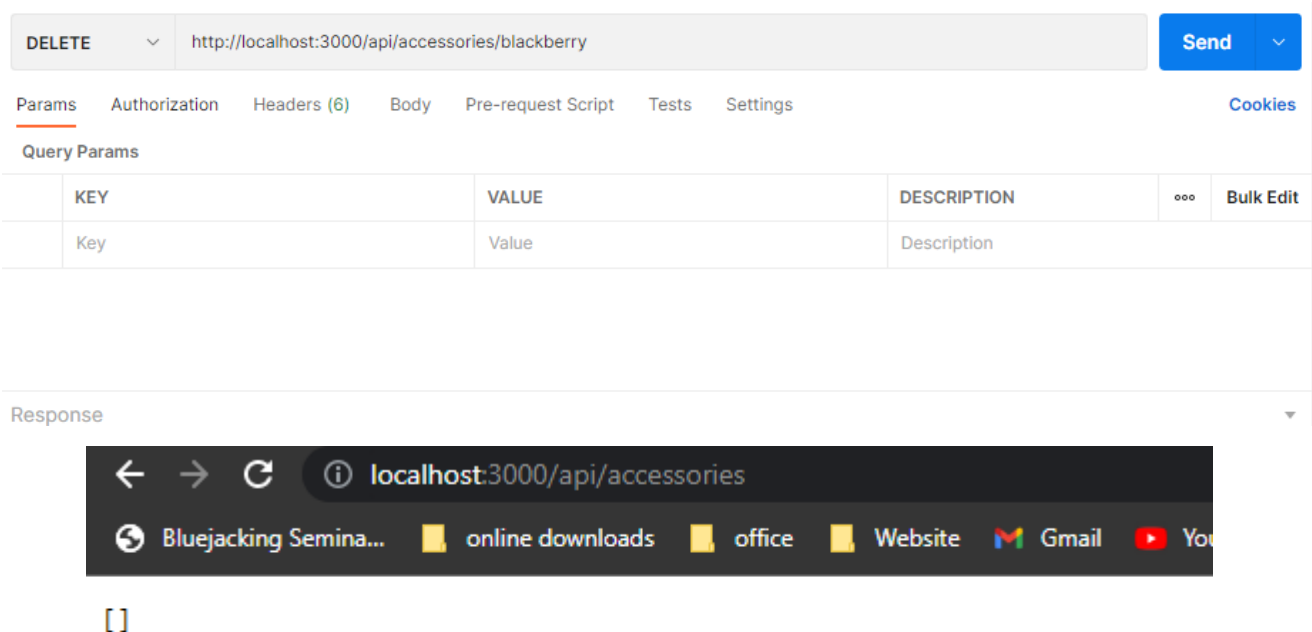
```
router.delete("/accessories/:name", async (req, res) => {
  await Accessorie.deleteOne({
    name: req.params.name
  }, (err, msg) => {
```

```

    if (err) {
      res.status(500).json({
        error: err
      })
    } else {
      res.status(200).json({
        msg: msg
      })
    }
  })
  const accessorie = await Accessorie.find()
  res.send(accessorie)
})
module.exports = router;

```

:: OUTPUT ::



The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/api/accessories/blackberry`. The response is an empty array `[]`.

Request Details:

- Method: DELETE
- URL: `http://localhost:3000/api/accessories/blackberry`
- Params: Query Params (Empty)

Response:

```
[ ]
```

Tutorial – 05

→Models→Accessories.js

```
var mongoose = require('mongoose');
```

```
var accessoriesSchema = mongoose.Schema({  
  name:String,  
  price:String  
})
```

```
module.exports = mongoose.model("accessories",accessoriesSchema)
```

→index.html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
</head>  
<body>  
  <h1>Hosted Successfully..!!</h1>  
</body>  
</html>
```

→index.js

```
var mongoose = require("mongoose")
```



```
var express = require("express")
var route = require('./routes')
var bodyParser = require('body-parser')
var app = express()

mongoose.connect("mongodb://127.0.0.1:27017/SPElectronics", { useUnifiedTopology: true
}).then(() => {

console.log("Router Connected...!!")

app.use(bodyParser.urlencoded({ extended: false }))
app.use('/api', route)

app.get('/', (req, res) => {
  res.sendFile('index.html', { root: __dirname })
})

app.listen((process.env.PORT || 3000), () => {
  console.log('Server Started Successfully...!!')
})
}).catch((e) => {
  console.log(e.toString())
})

→ routes.js
var express = require('express');
var router = express.Router();
var Accessorie = require('./Models/Accessorie')

//to fetch data from the SPElectronics data base in accessories table
```

```
router.get('/accessories', async (req, res) => {
  const accessorie = await Accessorie.find()
  res.send(accessorie)
})

//to insert data to the SPElectronics data base in accessories table
router.post("/accessories", async (req, res) => {
  const accessorie = new Accessorie({
    name: req.body.name,
    price: req.body.price
  })

  await accessorie.save((err, msg) => {
    if (err) {
      res.status(500).json({
        "error": err
      })
    }
    else {
      res.status(200).json({
        "My-message": msg
      })
    }
  })
})

// to update data for the SPElectronics data base in accessories table
router.patch('/accessories/:id', async (req, res) => {
  const accessorie = await Accessorie.findOne({_id:req.params.id})
```

```
accessorie.name = req.body.name
accessorie.price = req.body.price
await accessorie.save((err,msg)=>{
  if(err){
    res.status(500).json({
      error:err
    })
  }
  else{
    res.status(200).json({
      msg:msg
    })
  }
})

//to delete data for the SPElectronics data base in accessories table
router.delete("/accessories/:name",async(req,res)=>{
  await Accessorie.deleteOne({name:req.params.name}),(err,msg)=>{
    if(err){
      res.status(500).json({
        error:err
      })
    }
    else{
      res.status(200).json({
        msg:msg
      })
    }
  }
})
```

```
    })  
    const accessorie = await Accessorie.find()  
    res.send(accessorie)  
  })  
  
module.exports = router;
```

Tutorial – 6 (Working with Typescript)

36. Install typescript.

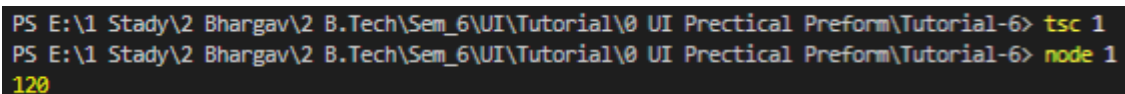
Under the cmd terminal you can type → `npm i -g typescript`

37. Create an arrow function that calculates the sum of n natural numbers. n is passed as a parameter.

CODE :

```
var sum = (n: number) => {  
  return (n*(n + 1) / 2)  
}  
console.log(sum(15));
```

::OUTPUT::



```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> tsc 1  
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> node 1  
120
```

38. Create three arrow functions that demonstrate usage of default parameter, optional parameter and rest parameter.

CODE :

****Default Parameter****

```
console.log("Default Parameter");  
let user = (fname: string = 'Bhargav', lname: string) => {  
  var fristname = fname;  
  var lastname = lname;  
  console.log(fristname);  
  console.log(lastname);  
}  
user('Bk', 'patel')
```

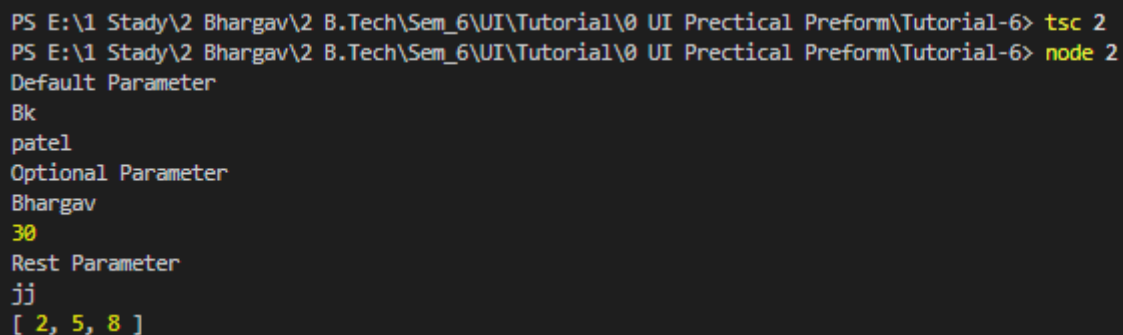
****Optional Parameter****

```
console.log("Optional Parameter");  
let student = (fname: String, erno ? : number) => {  
  var fristname = fname;  
  var enrollmentno = erno;  
  console.log(fristname);  
}
```

```
        console.log(enrollmentno);
    }
    student('Bhargav', 30)

**Rest Parameter**
    console.log("Rest Parameter");
    let staff = (fname: String, ...id: number[]) => {
        var fristname = fname;
        var idd = id;
        console.log(fristname);
        console.log(idd);
    }
    staff('jj', 2, 5, 8)
```

::OUTPUT::



```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> tsc 2
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> node 2
Default Parameter
Bk
patel
Optional Parameter
Bhargav
30
Rest Parameter
jj
[ 2, 5, 8 ]
```

39. Create an interface called student with name, city, branch properties and display method.

Also, create an object to utilize the student interface.

CODE :

```
interface student {
    name: string,
    city: string,
    branch: string,
    display: () => {}
}

var obj: student = {
    name: "Bhargav",
    city: "Amreli",
    branch: "Computer Engineering",
    display: (): string => { return "Name : " + obj.name + "\n" +
        "City : " + obj.city + "\n" +
        "Branch : " + obj.branch }
```

```
}  
console.log(obj.display())
```

::OUTPUT::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> tsc 3  
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> node 3  
Name : Bhargav  
City : Amreli  
Branch : Computer Engineering
```

40. Demonstrate the usage of single level and multiple inheritance of interface.

CODE :

//Single Inheritance

```
interface base {  
    name: string  
}  
interface derived extends base {  
    enrollment: string,  
    display: () => {}  
}  
  
var student: derived = {  
    name: "Bhargav Vasani",  
    enrollment: "20SOECE13031",  
    display: (): string => { return "Name : " + student.name + "\n" + "Enrollment : " +  
student.enrollment }  
}  
console.log(student.display())
```

//Multiple Inheritance

```
interface A {  
    name: string  
}  
interface B {  
    enrollment: string  
}  
  
interface marks extends A, B {  
    CGPA: number  
}  
  
var result: marks = {
```

```
name: "Bhargav Vasani",  
enrollment: "20SOECE13031",  
CGPA: 8.7  
}  
console.log("Name : " + result.name + "\n" + "Enrollment Number: " + result.enrollment + "\n" +  
"CGPA : " + result.CGPA)
```

::OUTPUT::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> tsc 4  
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\0 UI Prectical Preform\Tutorial-6> node 4  
Name : Bhargav Vasani  
Enrollment : 20SOECE13031  
Name : Bhargav Vasani  
Enrollment Number: 20SOECE13031  
CGPA : 8.7
```

41. Define a class Clock with three private integer data members hour, min and sec.

Define a no

argument constructor to initialize time value to 12:00:00.

Define a three argument constructor to initialize the time.

Define a methods to

- a. Increment time to the next second.**
- b. Display the time value.**
- c. Return the hour (getHour():number)**
- d. Return the minute (getMinute():number)**
- e. Return the seconds (getSeconds():number)**

CODE :

```
class clock {  
    private hours: number  
    private min: number  
    private sec: number  
  
    constructor() {  
        this.hours = 12  
        this.min = 0  
        this.sec = 0  
    }  
}
```



```
incSec() {  
    return this.sec++  
}  
  
getHour(): number {  
    return this.hours  
}  
  
getMinute(): number {  
  
    return this.min  
}  
  
getSeconds(): number {  
    return this.sec  
}  
  
Display() {  
    this.incSec()  
    console.log(this.getHour() + ":" + this.getMinute() + ":" + this.getSeconds())  
}  
}  
  
var c = new clock()  
c.Display();
```

::OUTPUT::

```
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Tutorial-6> tsc 6  
PS E:\1 Stady\2 Bhargav\2 B.Tech\Sem_6\UI\Tutorial\Tutorial-6> node 6  
12:0:1
```

42. The employee interface for a company contains employee code, name, designation and basic pay. The employee is given a house rent allowance (HRA) of 10% of the basic pay and dearness allowance (DA) of 45% of the basic pay. The total pay of the employee is calculated as Basic Pay + HRA + DA. Write a class to define the details of the employee. Write a constructor to assign the required initial values. Add a method to calculate HRA, DA and total pay and print them. Create objects for three different employees and calculate HRA, DA and total pay.

CODE :

::OUTPUT::

Tutorial - 7

Create a component that displays data the from the live API created using node(refer prev tutorials).

Make sure to use following things:

- ngFor
- bootstrap (any)
- Pagination
- Sorting
- Searching
- Service
- Interface

Code :

→App.module.ts

```
import{ NgModule} from '@angular/core';  
import{ BrowserModule} from '@angular/platform-browser';  
import{ AppRoutingModule} from './app-routing.module';  
import{ AppComponent} from './app.component';  
import{ PostsComponent} from './posts/posts.component';  
import{ HttpClientModule} from '@angular/common/http';  
import{ NgxPaginationModule} from 'ngx-pagination';  
import{ OrderModule} from 'ngx-order-pipe';
```

```
@NgModule({  
  declarations:[  
    AppComponent,  
    PostsComponent  
  ],  
  imports:[  
    BrowserModule,  
    AppRoutingModule,  
    HttpClientModule,
```

```
    NgxPaginationModule,  
    OrderModule  
  ],  
  providers:[],  
  bootstrap:[AppComponent]  
})  
exportclass AppModule{ }
```

→Post.service.ts

```
import{ Injectable} from'@angular/core';  
import{HttpClient} from'@angular/common/http';
```

```
@Injectable({  
  providedIn:'root'  
})  
exportclass PostService{
```

```
  url= "https://jsonplaceholder.typicode.com/posts"  
  constructor(private_http:HttpClient) { }  
  getData(){  
    returnthis._http.get(this.url);  
  }  
}
```

→Posts.component.ts

```
import{ Component, OnInit} from'@angular/core';  
import{ PostService} from'../Service/post.service';
```

```
@Component({
  selector: 'app-posts',
  templateUrl: './posts.component.html',
  styleUrls: ['./posts.component.css']
})
export class PostsComponent implements OnInit {
  data: any = []
  p: number = 1;

  key = 'id';
  reverse: boolean = false;
  constructor(private _postService: PostService) {}
  ngOnInit(): void {
    this._postService.getData().subscribe(res => {
      this.data = res;
      console.log(this.data)
    })
  }
  sort(key: string) {
    this.key = key;
    this.reverse = !this.reverse;
  }
}
```

→ Posts.component.html

```
<table class="table">
  <thead>
```

```
<th(click)="sort('id')">Id</th>

<th(click)="sort('title')">Title</th>

<th(click)="sort('body')">Body</th>

</thead>

<tbody>

<tr*ngFor="let item of data | paginate: { itemsPerPage: 10, currentPage: p } | orderBy:key:reverse">

  <td>{{item.id}}</td>

  <td>{{item.title}}</td>

  <td>{{item.body}}</td>

</tr>

</tbody>

</table>

<pagination-controls(pageChange)="p = $event"></pagination-controls>
```

::Output::

Id	Title	Body
1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
6	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
7	magnum facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
8	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
9	nesciunt iure omnis dolorem tempora et accusantium	consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
10	optio molestias id quia eum	quo et expedita modi cum officia vel magni doloribus qui repudiandae vero nisi sit quos veniam quod sed accusamus veritatis error

Id	Title	Body
10	optio molestias id quia eum	quo et expedita modi cum officia vel magni doloribus qui repudiandae vero nisi sit quos veniam quod sed accusamus veritatis error
9	nesciunt iure omnis dolorem tempora et accusantium	consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
8	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
7	magnam facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
6	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciat et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto

Id	Title	Body
11	et ea vero quia laudantium autem	delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus accusamus in eum beatae sit vel qui neque voluptates ut commodi qui incidunt ut animi commodi
12	in quibusdam tempore odit est dolorem	itaque id aut magnam praesentium quia et ea odit et ea voluptas et sapiente quia nihil amet occaecati quia id voluptatem incidunt ea est distinctio odio
13	dolorum ut in voluptas mollitia et saepe quo animi	aut dicta possimus sint mollitia voluptas commodi quo doloremque iste corrupti reiciendis voluptatem eius rerum sit cumque quod eligendi laborum minima perferendis recusandae assumenda consectetur porro architecto ipsum ipsam
14	voluptatem eligendi optio	fuga et accusamus dolorum perferendis illo voluptas non doloremque neque facere ad qui dolorum molestiae beatae sed aut voluptas totam sit illum
15	eveniet quod temporibus	reprehenderit quos placeat velit minima officia dolores impedit repudiandae molestiae nam voluptas recusandae quis delectus officiis harum fugiat vitae
16	sint suscipit perspiciat velit dolorum rerum ipsa laboriosam odio	suscipit nam nisi quo aperiam aut asperiores eos fugit maiores voluptatibus quia voluptatem quis ullam qui in alias quia est consequatur magni mollitia accusamus ea nisi voluptate dicta
17	fugit voluptas sed molestias voluptatem provident	eos voluptas et aut odit natus earum aspernatur fuga molestiae ullam deserunt ratione qui eos qui nihil ratione nemo velit ut aut id quo
18	voluptate et itaque vero tempora molestiae	eveniet quo quis laborum totam consequatur non dolor ut et est repudiandae est voluptatem vel debitis et magnam
19	adipisci placeat illum aut reiciendis qui	illum quis cupiditate provident sit magnam ea sed aut omnis veniam maiores ullam consequatur atque adipisci quo iste expedita sit quos voluptas
20	doloribus ad provident suscipit at	qui consequuntur ducimus possimus quisquam amet similique suscipit porro ipsam amet eos veritatis officiis exercitationem vel fugit aut necessitatibus totam omnis rerum consequatur expedita quidem cumque explicabo

Id	Title	Body
41	non est facere	molestias id nostrum excepturi molestiae dolore omnis repellendus quaerat saepe consectetur iste quaerat tenetur asperiores accusamus ex ut nam quidem est ducimus sunt debitis saepe
42	commodi ullam sint et excepturi error explicabo praesentium voluptas	odio fugit voluptatum ducimus earum autem est incidunt voluptatem odit reiciendis aliquam sunt sequi nulla dolorem non facere repellendus voluptates quia ratione harum vitae ut
43	eligendi iste nostrum consequuntur adipisci praesentium sit beatae perferendis	similique fugit est illum et dolorum harum et voluptate eaque quidem exercitationem quos nam commodi possimus cum odio nihil nulla dolorum exercitationem magnam ex et a et distinctio debitis
44	optio dolor molestias sit	temporibus est consectetur dolore et libero debitis vel velit laboriosam quia ipsum quibusdam qui itaque fuga rem aut ea et iure quam sed maxime ut distinctio quae
45	ut numquam possimus omnis eius suscipit laudantium iure	est natus reiciendis nihil possimus aut provident ex et dolor repellat pariat est nobis rerum repellendus dolorem autem
46	aut quo modi neque nostrum ducimus	voluptatem quisquam iste voluptatibus natus officiis facilis dolorem quis quas ipsam vel et voluptatum in aliquid
47	quibusdam cumque rem aut deserunt	voluptatem assumenda ut qui ut cupiditate aut impedit veniam occaecati nemo illum voluptatem laudantium molestiae beatae rerum ea iure soluta nostrum eligendi et voluptate
48	ut voluptatem illum ea doloribus itaque eos	voluptates quo voluptatem facilis iure occaecati vel assumenda rerum officia et illum perspiciatis ab deleniti laudantium repellat ad ut et autem reprehenderit
49	laborum non sunt aut ut assumenda perspiciatis voluptas	inventore ab sint natus fugit id nulla sequi architecto nihil quaerat eos tenetur in in eum veritatis non quibusdam officiis aspernatur cumque aut commodi aut
50	repellendus qui recusandae incidunt voluptates tenetur qui omnis exercitationem	error suscipit maxime adipisci consequuntur recusandae voluptas eligendi et est et voluptates quia distinctio ab amet quaerat molestiae et vitae adipisci impedit sequi nesciunt quis consectetur

« Previous 1 ... 4 **5** 6 ... 10 Next »