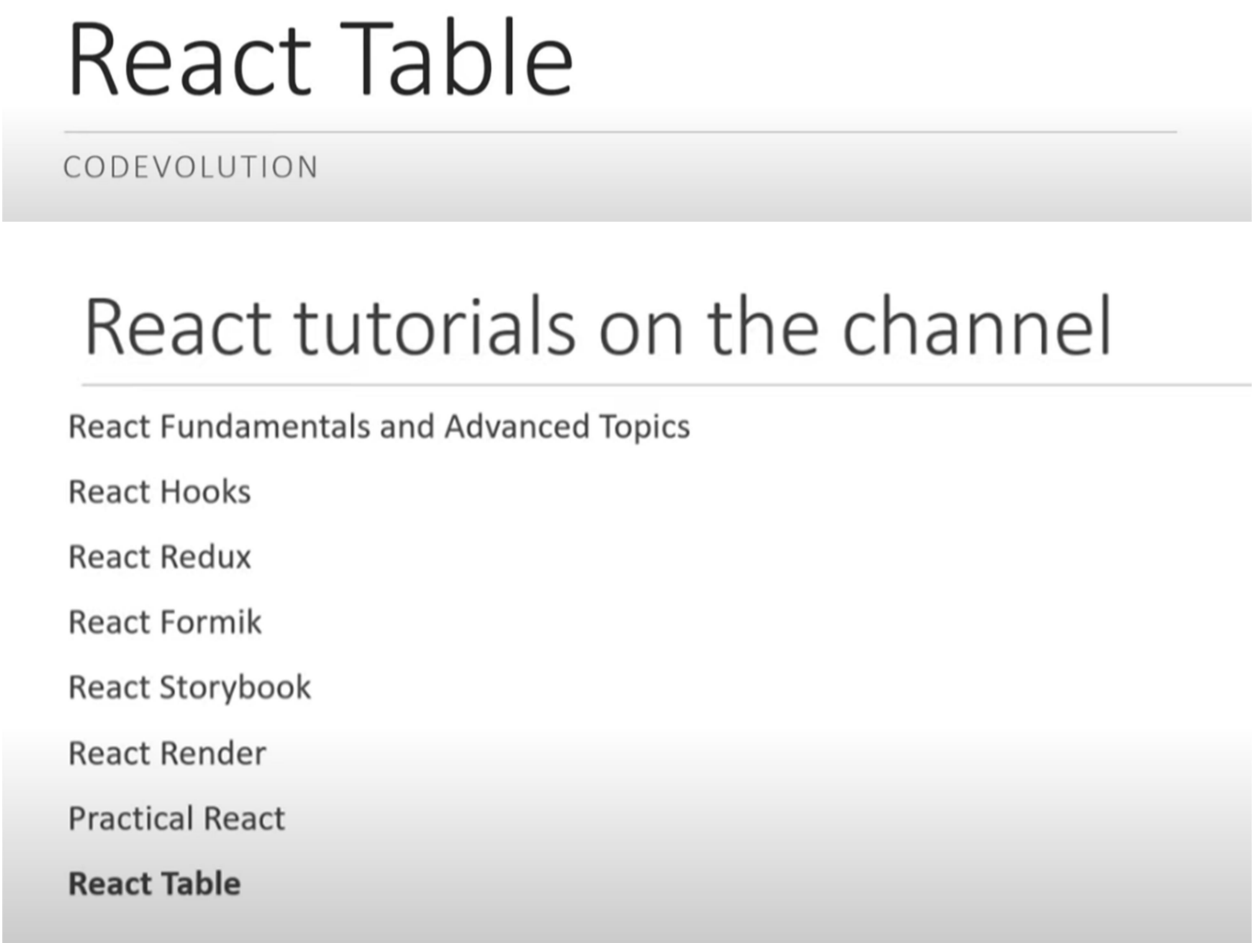
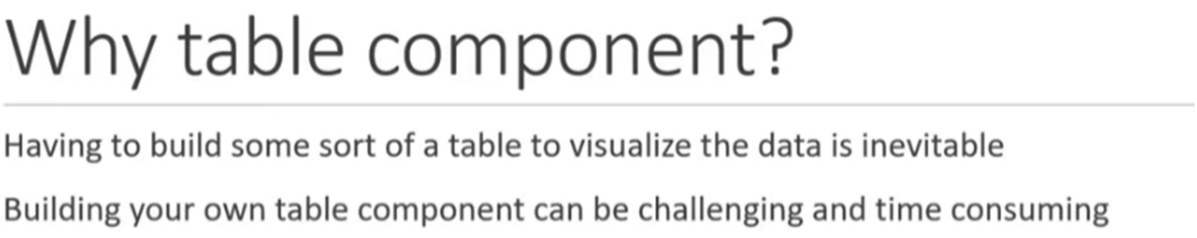
Chapter 1:

React Table is the utlity, to get the features, we need to import some or the other hooks.





React Table

“

React Table is a collection of hooks for building powerful tables and data grid experiences

These hooks are lightweight, composable, and ultra-extensible

They do not render any markup or styles for you

”

It is a table utility and not a table component

Course Structure

React Table setup

Basic table

Group headers and footers

Sorting, Filtering and Pagination

Select rows

Column ordering, Column Hiding and Sticky Columns

Prerequisites

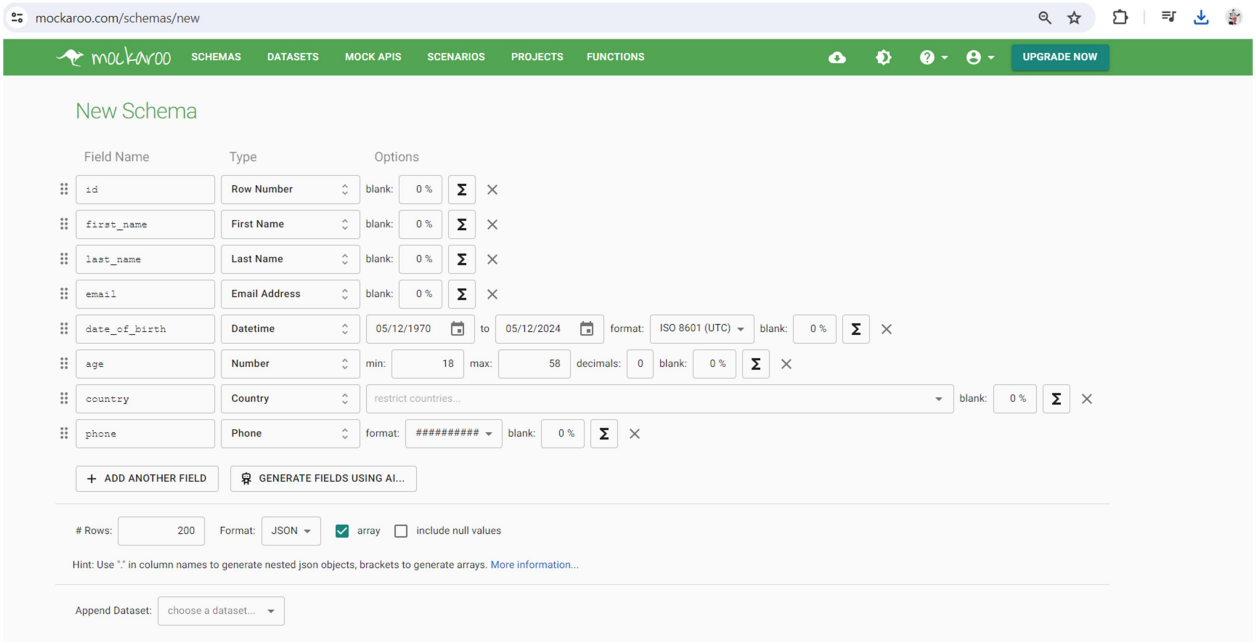
React Fundamentals

React Hooks

Chapter 2:

1. Create npx create-react-app my-react-table
2. Install the depdency npm i react-table
3. Create a component folder in src
4. For dummy data, use https://www.mockaroo.com/schemas/new

5. Generate the data as below, and past in component folder



Chapter 3:

Basic Table

Get the data you want to display

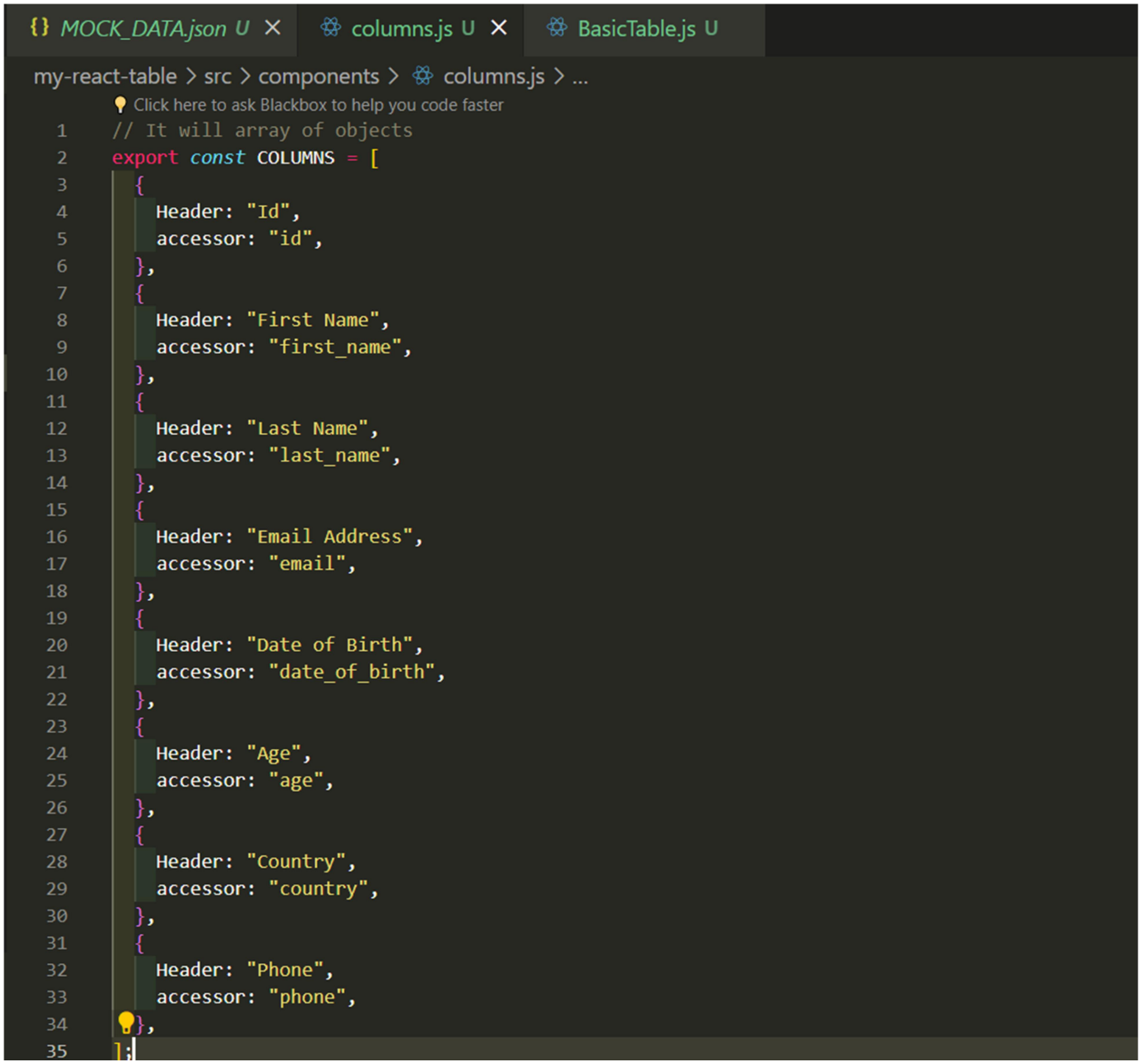
Define the columns for your table

Use the data and columns defined to create a table instance using react-table

Define a basic table structure using plain HTML

Use the table instance created in step 3 to bring life to the HTML defined in step 4

Include the desired CSS



Now create a basic table: BasicTable.js

We use, useMemo, so that Data is not created on every render

Copy the css styles from https://www.w3schools.com/css/tryit.asp?filename=trycss_table_fancy

BasicTable.js

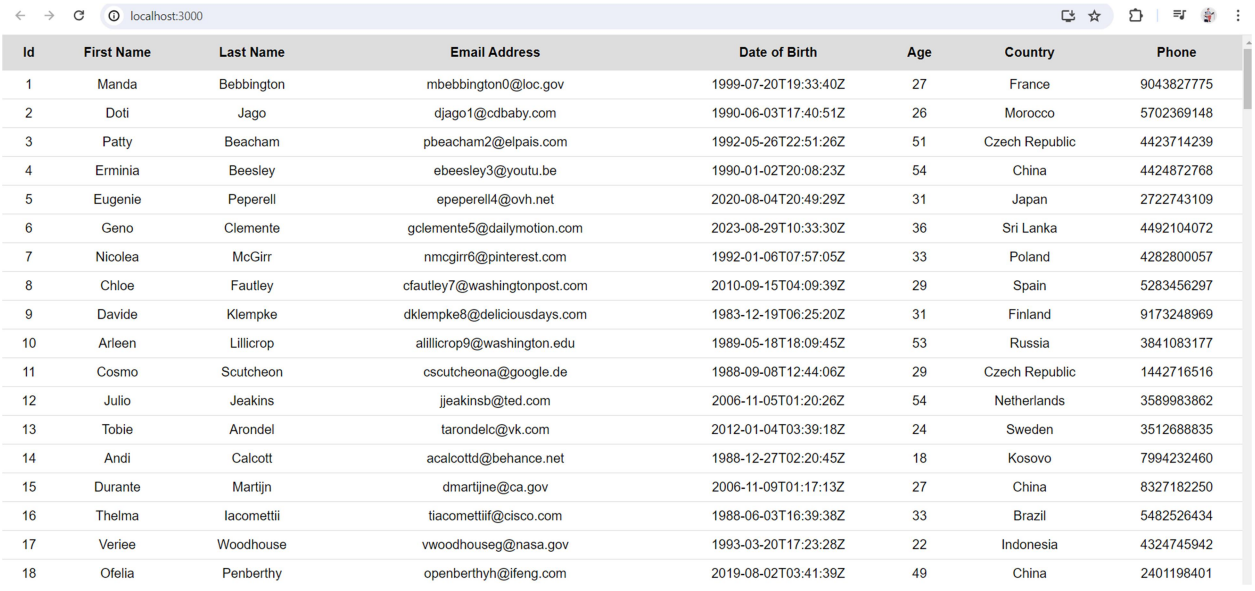
```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable } from "react-table";
import { COLUMNS } from "./columns";
import MOCK_DATA from "./MOCK_DATA.json";

import "./table.css";
```

```jsx
const BasicTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  const columns = useMemo(() => COLUMNS, []);
  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable({
    /*     columns: columns,
    data: data, */
    columns,
    data,
  });

  const { getTableProps, getTableBodyProps, headerGroups, rows, prepareRow } =
    tableInstance;
  return (
    <div>
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>{column.render("Header")}</th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
      </table>
    </div>
  );
};

export default BasicTable;
```

table.css

```css
.customers {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

.customers td,
.customers th {
  padding: 8px;
  border-bottom: 1px solid #ddd;
}

/* .customers tr:nth-child(even) {
  background-color: #f2f2f2;
} */

.customers tr:hover {
  background-color: #ddd;
}

.customers th {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: center;
  /*  background-color: #04aa6d;
  color: white; */
  background-color: #ddd;
  color: #000;
}
```

Output:



Id	First Name	Last Name	Email Address	Date of Birth	Age	Country	Phone
1	Manda	Bebbington	mbebbington0@loc.gov	1999-07-20T19:33:40Z	27	France	9043827775
2	Doti	Jago	djago1@cdbaby.com	1990-06-03T17:40:51Z	26	Morocco	5702369148
3	Patty	Beacham	pbeacham2@elpais.com	1992-05-26T22:51:26Z	51	Czech Republic	4423714239
4	Erminia	Beesley	ebeesley3@youtu.be	1990-01-02T20:08:23Z	54	China	4424872768
5	Eugenie	Peperell	epeperell4@ovh.net	2020-08-04T20:49:29Z	31	Japan	2722743109
6	Geno	Clemente	gclemente5@dailymotion.com	2023-08-29T10:33:30Z	36	Sri Lanka	4492104072
7	Nicolea	McGirr	nmcgirr6@pinterest.com	1992-01-06T07:57:05Z	33	Poland	4282800057
8	Chloe	Fautley	cfautley7@washingtonpost.com	2010-09-15T04:09:39Z	29	Spain	5283456297
9	Davide	Klempke	dklempke8@deliciousdays.com	1983-12-19T06:25:20Z	31	Finland	9173248969
10	Arleen	Lillicrop	alillicrop9@washington.edu	1989-05-18T18:09:45Z	53	Russia	3841083177
11	Cosmo	Scutcheon	cscutcheona@google.de	1988-09-08T12:44:06Z	29	Czech Republic	1442716516
12	Julio	Jeakins	jjeakinsb@ted.com	2006-11-05T01:20:26Z	54	Netherlands	3589983862
13	Tobie	Arondel	tarondelc@vk.com	2012-01-04T03:39:18Z	24	Sweden	3512688835
14	Andi	Calcott	acalcottd@behance.net	1988-12-27T02:20:45Z	18	Kosovo	7994232460
15	Durante	Martijn	dmartijne@ca.gov	2006-11-09T01:17:13Z	27	China	8327182250
16	Thelma	Iacomettii	tiacomettiif@cisco.com	1988-06-03T16:39:38Z	33	Brazil	5482526434
17	Veriee	Woodhouse	vwoodhouseg@nasa.gov	1993-03-20T17:23:28Z	22	Indonesia	4324745942
18	Ofelia	Penberthy	openberthyh@ifeng.com	2019-08-02T03:41:39Z	49	China	2401198401

Chapter 4: **Adding footer**:

1. Adding the footer would be similar to adding Header in column, Just replicate the same as we did for Header, add Footer in columns.js
2. Now, Destructure the footGroups from table instance

BasicTable.js

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable } from "react-table";
import { COLUMNS } from "./columns";
import MOCK_DATA from "./MOCK_DATA.json";

import "./table.css";

const BasicTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  const columns = useMemo(() => COLUMNS, []);
  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable({
    /*     columns: columns,
    data: data, */
    columns,
    data,
  });

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
  } = tableInstance;
  return (
    <div>
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>{column.render("Header")}</th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
```

```
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
        <tfoot>
          {footerGroups.map((footerGroup) => (
            <tr {...footerGroup.getFooterGroupProps()}>
              {footerGroup.headers.map((column) => (
                <th {...column.getFooterProps()}>{column.render("Footer")}</th>
              ))}
            </tr>
          ))}
        </tfoot>
      </table>
    </div>
  );
};

export default BasicTable;
```

table.css

```
.customers {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

.customers td {
  padding: 8px;
  border-bottom: 1px solid #ddd;
}
.customers th {
  padding: 8px;
  border: 1px solid #fff;
}

/* .customers tr:nth-child(even) {
  background-color: #f2f2f2;
} */

.customers tr:hover {
  background-color: #ddd;
```

```
}

.customers th,
tfoot td {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: center;
  /*  background-color: #04aa6d;
  color: white; */
  background-color: #ddd;
  color: #000;
}
```

Columns.js

```
// It will array of objects
export const COLUMNS = [
  {
    Header: "Id",
    Footer: "Id",
    accessor: "id",
  },
  {
    Header: "First Name",
    Footer: "First Name",
    accessor: "first_name",
  },
  {
    Header: "Last Name",
    Footer: "Last Name",
    accessor: "last_name",
  },
  {
    Header: "Email Address",
    Footer: "Email Address",
    accessor: "email",
  },
  {
    Header: "Date of Birth",
    Footer: "Date of Birth",
    accessor: "date_of_birth",
  },
  {
    Header: "Age",
    Footer: "Age",
    accessor: "age",
  },
  {
    Header: "Country",
    Footer: "Country",
```

```
    accessor: "country",
  },
  {
    Header: "Phone",
    Footer: "Phone",
    accessor: "phone",
  },
];
```

Chapter 5: **Grouping Headers**

columns.js

```
// It will array of objects
export const COLUMNS = [
  {
    Header: "Id",
    Footer: "Id",
    accessor: "id",
  },
  {
    Header: "Email Address",
    Footer: "Email Address",
    accessor: "email",
  },
  {
    Header: "Date of Birth",
    Footer: "Date of Birth",
    accessor: "date_of_birth",
  },
  {
    Header: "Age",
    Footer: "Age",
    accessor: "age",
  },
  {
    Header: "Country",
    Footer: "Country",
    accessor: "country",
  },
  {
    Header: "Phone",
    Footer: "Phone",
    accessor: "phone",
  },
];

export const GROUP_COLUMNS = [
  {
    Header: "Id",
    Footer: "Id",
```

```
    accessor: "id",
  },
  {
    Header: "Name",
    Footer: "Name",
    columns: [
      {
        Header: "First Name",
        Footer: "First Name",
        accessor: "first_name",
      },
      {
        Header: "Last Name",
        Footer: "Last Name",
        accessor: "last_name",
      },
    ],
  },
  {
    Header: "Info",
    Footer: "Info",
    columns: [
      {
        Header: "Date of Birth",
        Footer: "Date of Birth",
        accessor: "date_of_birth",
      },
      {
        Header: "Age",
        Footer: "Age",
        accessor: "age",
      },
      {
        Header: "Country",
        Footer: "Country",
        accessor: "country",
      },
      {
        Header: "Phone",
        Footer: "Phone",
        accessor: "phone",
      },
    ]
  }
];
```

BasicTable.js

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable } from "react-table";
```

```jsx
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "./MOCK_DATA.json";

import "./table.css";

const BasicTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  // const columns = useMemo(() => COLUMNS, []);
  const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable({
    /*     columns: columns,
    data: data, */
    columns,
    data,
  });

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
  } = tableInstance;
  return (
    <div>
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>{column.render("Header")}</th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
```

```
          })}
        </tbody>
        <tfoot>
          {footerGroups.map((footerGroup) => (
            <tr {...footerGroup.getFooterGroupProps()}>
              {footerGroup.headers.map((column) => (
                <th {...column.getFooterProps()}>{column.render("Footer")}</th>
              ))}
            </tr>
          ))}
        </tfoot>
      </table>
    </div>
  );
};

export default BasicTable;
```

Output:

	Name		Info			
Id	First Name	Last Name	Date of Birth	Age	Country	Phone
1	Manda	Bebbington	1999-07-20T19:33:40Z	27	France	9043827775
2	Doti	Jago	1990-06-03T17:40:51Z	26	Morocco	5702369148
3	Patty	Beacham	1992-05-26T22:51:26Z	51	Czech Republic	4423714239
4	Erminia	Beesley	1990-01-02T20:08:23Z	54	China	4424872768
5	Eugenie	Peperell	2020-08-04T20:49:29Z	31	Japan	2722743109
6	Geno	Clemente	2023-08-29T10:33:30Z	36	Sri Lanka	4492104072
7	Nicolea	McGirr	1992-01-06T07:57:05Z	33	Poland	4282800057
8	Chloe	Fautley	2010-09-15T04:09:39Z	29	Spain	5283456297
9	Davide	Klempke	1983-12-19T06:25:20Z	31	Finland	9173248969
10	Arleen	Lillicrop	1989-05-18T18:09:45Z	53	Russia	3841083177
11	Cosmo	Scutcheon	1988-09-08T12:44:06Z	29	Czech Republic	1442716516

Chapter 6: **Sorting**

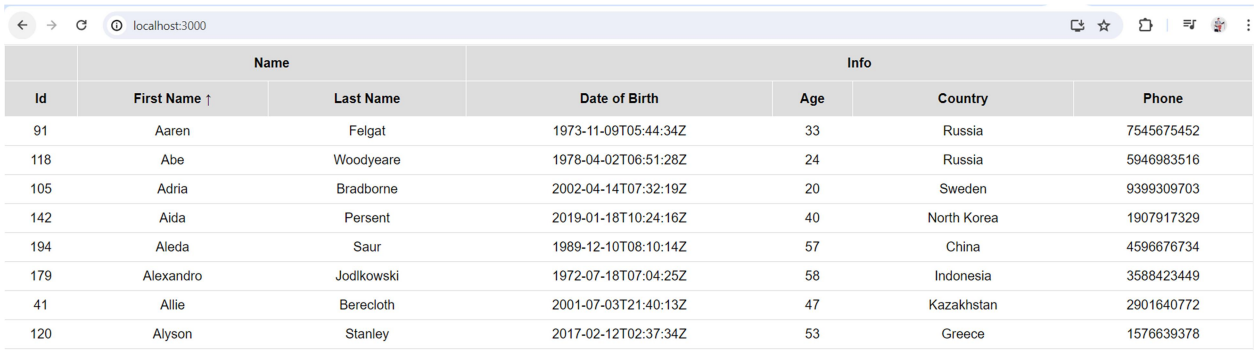
1. Create the SortingTable.js by replicating the existing BasicTble.js
2. Import ‘useSortBy’ from react-table
3. Pass the useSortBy in useTable
4. Pass ‘column.getSortByToggleProps()’ in th
5. Add the sorting icons column.getSortByToggleProps() ↑ ↓
6. If the column is not sorted then show the empty string.

SortingTable.js

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, useSortBy } from "react-table";
```

```jsx
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "./MOCK_DATA.json";

import "./table.css";

const SortingTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  // const columns = useMemo(() => COLUMNS, []);
  const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
      /*     columns: columns,
    data: data, */
      columns,
      data,
    },
    useSortBy
  );

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
  } = tableInstance;
  return (
    <div>
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps(column.getSortByToggleProps())}>
                  {column.render("Header")}
                  <span>
                    {column.isSorted ? (column.isSortedDesc ? " ↓ " : " ↑ ") : ""}
                  </span>
                </th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
```

```
            <tr {...row.getRowProps()}>
              {row.cells.map((cell) => {
                return (
                  <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                );
              })}
            </tr>
          );
        })}
      </tbody>
      <tfoot>
        {footerGroups.map((footerGroup) => (
          <tr {...footerGroup.getFooterGroupProps()}>
            {footerGroup.headers.map((column) => (
              <th {...column.getFooterProps()}>{column.render("Footer")}</th>
            ))}
          </tr>
        ))}
      </tfoot>
    </table>
  </div>
 );
};

export default SortingTable;
```



localhost:3000

	Name		Info			
Id	First Name ↑	Last Name	Date of Birth	Age	Country	Phone
91	Aaren	Felgat	1973-11-09T05:44:34Z	33	Russia	7545675452
118	Abe	Woodyeare	1978-04-02T06:51:28Z	24	Russia	5946983516
105	Adria	Bradborne	2002-04-14T07:32:19Z	20	Sweden	9399309703
142	Aida	Persent	2019-01-18T10:24:16Z	40	North Korea	1907917329
194	Aleda	Saur	1989-12-10T08:10:14Z	57	China	4596676734
179	Alexandro	Jodlkowski	1972-07-18T07:04:25Z	58	Indonesia	3588423449
41	Allie	Berecloth	2001-07-03T21:40:13Z	47	Kazakhstan	2901640772
120	Alyson	Stanley	2017-02-12T02:37:34Z	53	Greece	1576639378

Chapter 7: **Formatting the Header for Sorting**

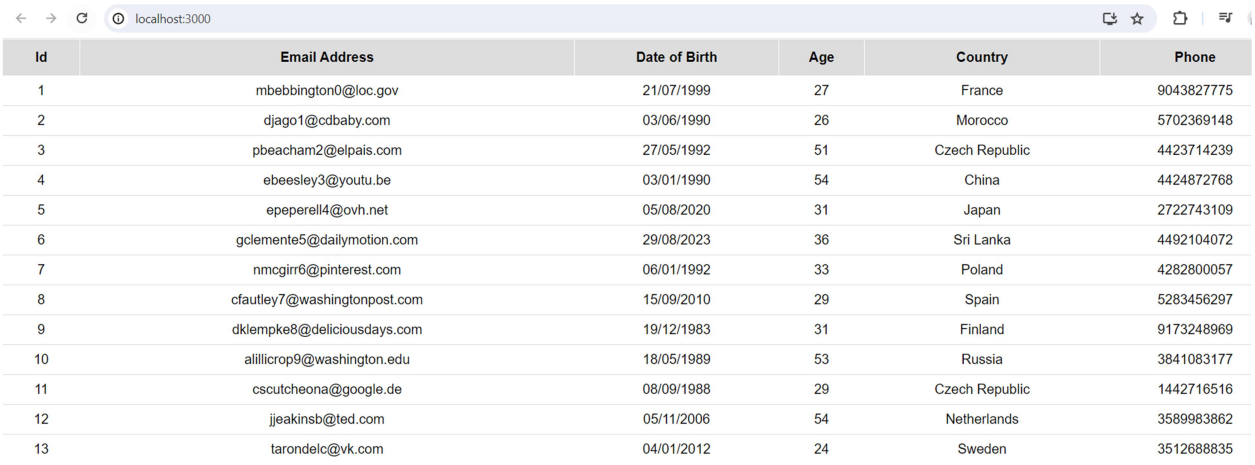
- By default sorting is happening based on string, so we need to sort based on the column
- We can use ‘npm i date-fn’ to format the date column.

Added the condition in column.js as below,

```
{

   Header: "Date of Birth",
   Footer: "Date of Birth",
```

```
    accessor: "date_of_birth",
    cell: ({ value }) => {
      return format(new Date(value), "dd/MM/yyyy");
    },
  },
```

Output:



Id	Email Address	Date of Birth	Age	Country	Phone
1	mbebbington0@loc.gov	21/07/1999	27	France	9043827775
2	djago1@cdbaby.com	03/06/1990	26	Morocco	5702369148
3	pbeacham2@elpais.com	27/05/1992	51	Czech Republic	4423714239
4	ebeesley3@youtu.be	03/01/1990	54	China	4424872768
5	epeperell4@ovh.net	05/08/2020	31	Japan	2722743109
6	gclemente5@dailymotion.com	29/08/2023	36	Sri Lanka	4492104072
7	nmcgirr6@pinterest.com	06/01/1992	33	Poland	4282800057
8	cfautley7@washingtonpost.com	15/09/2010	29	Spain	5283456297
9	dklempke8@deliciousdays.com	19/12/1983	31	Finland	9173248969
10	alillicrop9@washington.edu	18/05/1989	53	Russia	3841083177
11	cscutcheona@google.de	08/09/1988	29	Czech Republic	1442716516
12	jjeakinsb@ted.com	05/11/2006	54	Netherlands	3589983862
13	tarondelc@vk.com	04/01/2012	24	Sweden	3512688835

Chapter 8: **Filtering (**Global**)**

Filtering is of 2 types.

1.Global

2. Column

- Use ‘useGlobalFilter’ from react table
- Need to 2 import, state, setGlobalFilter from Table instance
- Destructure globalFilter from state.
- Create a input search box in separate file called GlobalFilte having 2 props 1. Filter, setFilter, pass these 2 values from FilteringTable.js

GlobalFilter.js:

```
import React from "react";

const GlobalFilter = ({ filter, setFilter }) => {
  return (
    <span>
      Search:{" "}
      <input value={filter || ""} onChange={(e) => setFilter(e.target.value)} />
    </span>
```

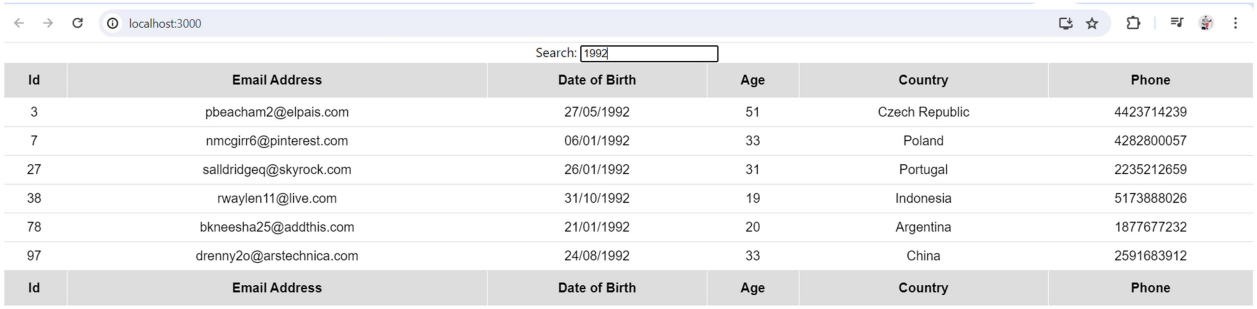
```
  );
};

export default GlobalFilter;
```

FilteringTable.js:

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, useGlobalFilter } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "./MOCK_DATA.json";

import "./table.css";
import GlobalFilter from "./GlobalFilter";

const FilteringTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  const columns = useMemo(() => COLUMNS, []);
  // const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
      /*     columns: columns,
    data: data, */
      columns,
      data,
    },
    useGlobalFilter
  );

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
    state,
    setGlobalFilter,
  } = tableInstance;

  const { globalFilter } = state;
  return (
    <div>
      <GlobalFilter filter={globalFilter} setFilter={setGlobalFilter} />
      {/* Step 4: Define the Table with basic HTML */}
```

```
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>{column.render("Header")}</th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
        <tfoot>
          {footerGroups.map((footerGroup) => (
            <tr {...footerGroup.getFooterGroupProps()}>
              {footerGroup.headers.map((column) => (
                <th {...column.getFooterProps()}>{column.render("Footer")}</th>
              ))}
            </tr>
          ))}
        </tfoot>
      </table>
    </div>
  );
};

export default FilteringTable;
```

Output:



Chapter 9: **Filtering (**Column**)**

- Create ColumnFilter.js by resuing the GlobalFilter code, having search input box.
- ColumnFilter component will get column as a prop.
- Get 'filterValue, setFilter' from column prop.
- In ColumnFilteringTable.js, import ColumnFilter.js
- useFilters should be extracted from react table and pass it to table instance.
- `<div>{column.canFilter ? column.render('Filter') : null}</div>`
- Add Above condition in th
- For column filter, in Column.js, Need to add Column field.

ColumnFilter.js:

```
import React from "react";

const ColumnFilter = ({ column }) => {
  const { filterValue, setFilter } = column;
  return (
    <span>
      Search:{" "}
      <input
        value={filterValue || ""}
        onChange={(e) => setFilter(e.target.value)}
      />
    </span>
  );
};

export default ColumnFilter;
```

ColumnsForColumFilter.js

```
import { format } from "date-fns";
import ColumnFilter from "./ColumnFilter";

// It will array of objects
export const COLUMNS = [
  {
    Header: "Id",
    Footer: "Id",
    accessor: "id",
    Filter: ColumnFilter,
  },
  {
    Header: "First Name",
    Footer: "First Name",
    accessor: "first_name",
    Filter: ColumnFilter,
  },
  {
    Header: "Last Name",
    Footer: "Last Name",
    accessor: "last_name",
    Filter: ColumnFilter,
  },
  {
    Header: "Email Address",
    Footer: "Email Address",
    accessor: "email",
    Filter: ColumnFilter,
  },
  {
    Header: "Date of Birth",
    Footer: "Date of Birth",
    accessor: "date_of_birth",
    Cell: ({ value }) => {
      return format(new Date(value), "dd/MM/yyyy");
    },
    Filter: ColumnFilter,
  },
  {
    Header: "Age",
    Footer: "Age",
    accessor: "age",
    Filter: ColumnFilter,
  },
  {
    Header: "Country",
    Footer: "Country",
    accessor: "country",
    Filter: ColumnFilter,
  },
  {
    Header: "Phone",
    Footer: "Phone",
    accessor: "phone",
```

```
    Filter: ColumnFilter,
  },
];

export const GROUP_COLUMNS = [
  {
    Header: "Id",
    Footer: "Id",
    accessor: "id",
  },
  {
    Header: "Name",
    Footer: "Name",
    columns: [
      {
        Header: "First Name",
        Footer: "First Name",
        accessor: "first_name",
      },
      {
        Header: "Last Name",
        Footer: "Last Name",
        accessor: "last_name",
      },
    ],
  },
  {
    Header: "Info",
    Footer: "Info",
    columns: [
      {
        Header: "Date of Birth",
        Footer: "Date of Birth",
        accessor: "date_of_birth",
        /* cell: ({ value }) => {
          return format(new Date(value), "dd/MM/yyyy");
        }, */
        cell: ({ value }) => {
          const date = new Date(value);
          if (isNaN(date)) {
            return "Invalid date"; // or handle as needed
          }
          return format(date, "dd/MM/yyyy");
        },
      },
      {
        Header: "Age",
        Footer: "Age",
        accessor: "age",
      },
      {
        Header: "Country",
        Footer: "Country",
        accessor: "country",
```

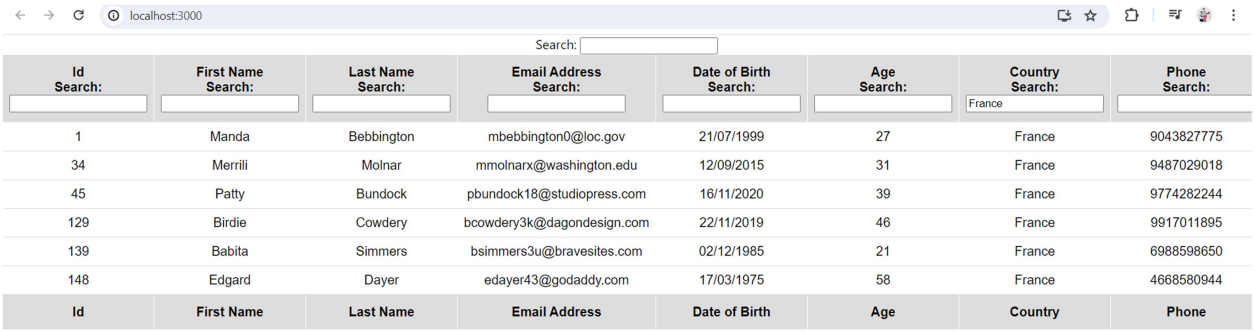
```
      },
      {
        Header: "Phone",
        Footer: "Phone",
        accessor: "phone",
      },
    ],
  },
];
```

ColumnWiseFilterTable.js

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, useGlobalFilter, useFilters } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columnsForColumnFilter";
import MOCK_DATA from "../MOCK_DATA.json";

import "../table.css";
import GlobalFilter from "../GlobalFilter";

const ColumnWiseFilteringTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  const columns = useMemo(() => COLUMNS, []);
  // const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
      /*     columns: columns,
    data: data, */
      columns,
      data,
    },
    useGlobalFilter,
    useFilters
  );

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
    state,
    setGlobalFilter,
  } = tableInstance;
```

```
  const { globalFilter } = state;
  return (
    <div>
      <GlobalFilter filter={globalFilter} setFilter={setGlobalFilter} />
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>
                  {column.render("Header")}
                  <div>{column.canFilter ? column.render("Filter") : null}</div>
                </th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
        <tfoot>
          {footerGroups.map((footerGroup) => (
            <tr {...footerGroup.getFooterGroupProps()}>
              {footerGroup.headers.map((column) => (
                <th {...column.getFooterProps()}>{column.render("Footer")}</th>
              ))}
            </tr>
          ))}
        </tfoot>
      </table>
    </div>
  );
};

export default ColumnWiseFilteringTable;
```

Output:



Chapter 10 : - More on Filtering

- If we don't want fiiltering at any column then, we cant directly remove the condition 'Filter: ColumnFilter'. But we can add extra property disableFilters: true. And remove the Filter:ColumnFilter in column.js
- Use debounce hook while filtering the data.

Global Filter:

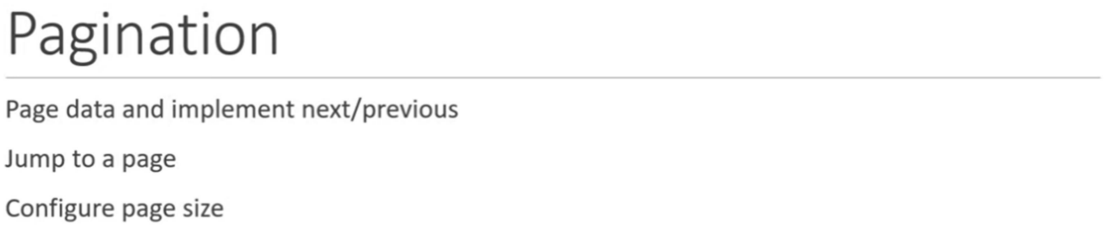
```
import React, { useState } from "react";
import { useAsyncDebounce } from "react-table";

const GlobalFilterDebounce = ({ filter, setFilter }) => {
  const [value, setValue] = useState(filter);

  const onChange = useAsyncDebounce((value) => {
    setFilter(value || undefined);
  }, 1000);

  return (
    <span>
      Search:{" "}
      <input
        value={value || ""}
        onChange={(e) => {
          setValue(e.target.value);
          onChange(e.target.value);
        }}
      />
    </span>
  );
};

export default GlobalFilterDebounce;
```

Filter Table:

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, useGlobalFilter, useFilters } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columnsForColumnFilter";
import MOCK_DATA from "../MOCK_DATA.json";

import "../table.css";
import GlobalFilterDebounce from "./GlobalFilterDebounce";
import ColumnFilter from "./ColumnFilter";

const ColumnWiseFilteringTableDebounce = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  const columns = useMemo(() => COLUMNS, []);
  // const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Setting default filter column
  const defaultColumn = useMemo(() => {
    return {
      Filter: ColumnFilter,
    };
  }, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
      /*     columns: columns,
    data: data, */
      columns,
      data,
      defaultColumn,
    },
    useGlobalFilter,
    useFilters
  );

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    footerGroups,
    rows,
    prepareRow,
    state,
    setGlobalFilter,
  } = tableInstance;
```

```
  const { globalFilter } = state;
  return (
    <div>
      <GlobalFilterDebounce filter={globalFilter} setFilter={setGlobalFilter} />
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>
                  {column.render("Header")}
                  <div>{column.canFilter ? column.render("Filter") : null}</div>
                </th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {rows.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
        <tfoot>
          {footerGroups.map((footerGroup) => (
            <tr {...footerGroup.getFooterGroupProps()}>
              {footerGroup.headers.map((column) => (
                <th {...column.getFooterProps()}>{column.render("Footer")}</th>
              ))}
            </tr>
          ))}
        </tfoot>
      </table>
    </div>
  );
};

export default ColumnWiseFilteringTableDebounce;
```

Chapter 11 : Pagination



- Copy the reference from **BasicTable**.js and Create **PaginationTable**.js
- We don't need footer, so need to remove footer
- Step1: Import '**usePagination'** from react table, and pass to the table instance.
- Destructure page instead of row. So change row to page
- We will get only 10 rows in basic pagination.
- Destructure, nextPage and previousPage from useTable.
- Add 2 button previous and next, call **nextPage** and **previousPage** function on each button.
- To make disabled button, we use **canNextPage**, **canPreviousPage** from useTable hook.
- We can see to **current & Total page**, so use '**state, pageOptions'** from **useTable** hook.
- Destructure **pageIndex** from state.

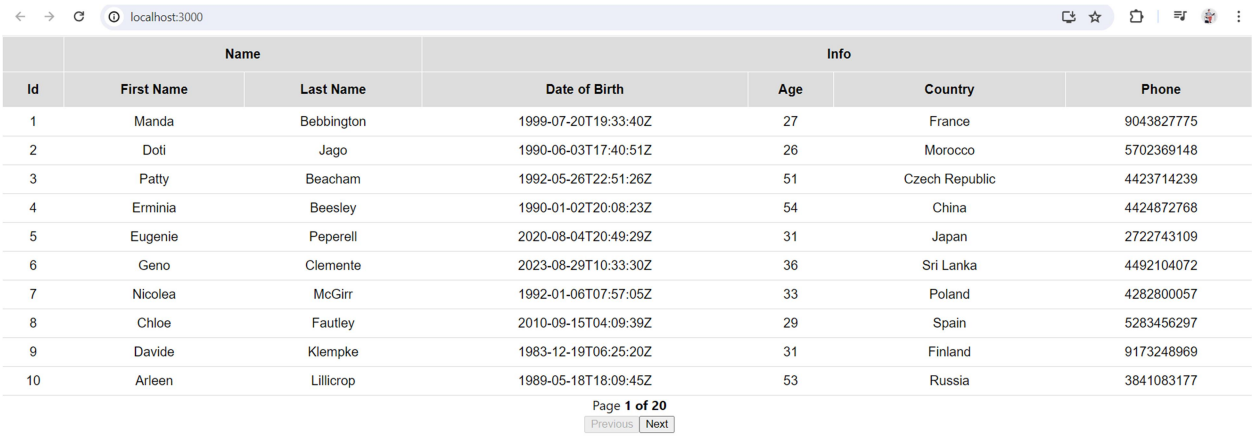
PaginationTable.js:

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, usePagination } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "../MOCK_DATA.json";

import "../table.css";

const PaginationTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  // const columns = useMemo(() => COLUMNS, []);
  const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
```

```
    /*     columns: columns,
  data: data, */
    columns,
    data,
  },
  usePagination
);

const {
  getTableProps,
  getTableBodyProps,
  headerGroups,
  page,
  prepareRow,
  nextPage,
  previousPage,
  canNextPage,
  canPreviousPage,
  state,
  pageOptions,
} = tableInstance;

const { pageIndex } = state;
return (
  <>
    {/* Step 4: Define the Table with basic HTML */}
    <table {...getTableProps()} className="customers">
      <thead>
        {headerGroups.map((headerGroup) => (
          <tr {...headerGroup.getHeaderGroupProps()}>
            {headerGroup.headers.map((column) => (
              <th {...column.getHeaderProps()}>{column.render("Header")}</th>
            ))}
          </tr>
        ))}
      </thead>
      <tbody {...getTableBodyProps}>
        {page.map((row) => {
          prepareRow(row);
          return (
            <tr {...row.getRowProps()}>
              {row.cells.map((cell) => {
                return (
                  <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                );
              })}
            </tr>
          );
        })}
      </tbody>
    </table>

    <span>
      Page{" "}
```

```
        <strong>
          {pageIndex + 1} of {pageOptions.length}
        </strong>
      </span>
      <div>
        <button onClick={() => previousPage()} disabled={!canPreviousPage}>
          Previous
        </button>
        <button onClick={() => nextPage()} disabled={!canNextPage}>
          Next
        </button>
      </div>
    </>
  );
};

export default PaginationTable;
```

Output:



Chapter 12 : Pagination (Goto Specific Page)

- We can use ‘gotoPage’ and ‘pageCount’ from useTable hook.
- We can go directly to first page or last page using >> or << based on handlers

```
<button onClick={() => gotoPage(0)} disabled={!canPreviousPage}>
    {"<<"}
</button>
```

and

```
<button onClick={() => gotoPage(pageCount - 1)} disabled={!canNextPage}>
  {">>"}
</button>
```

- We can jump to any of the rows

```
      <span>
         | Go to page:{" "}
         <input
           type="number"
           defaultValue={pageIndex + 1}
           onChange={(e) => {
             const pageNumber = e.target.value
               ? Number(e.target.value) - 1
               : 0;
             gotoPage(pageNumber);
           }}
           style={{ width: '50px'}}
         />
       </span>
```

Complete PaginationTable.js

```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, usePagination } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "../MOCK_DATA.json";

import "../table.css";

const PaginationTable = () => {
  // Step 2: To get the Column and set the data
  //Here, Data is not created on every render
  // const columns = useMemo(() => COLUMNS, []);
  const columns = useMemo(() => GROUP_COLUMNS, []);

  const data = useMemo(() => MOCK_DATA, []);

  // Step 3: Create the instance of the table
  const tableInstance = useTable(
    {
      /*     columns: columns,
    data: data, */
      columns,
      data,
    },
    usePagination
  );

  const {
    getTableProps,
    getTableBodyProps,
    headerGroups,
    page,
```

```
    prepareRow,
    nextPage,
    previousPage,
    canNextPage,
    canPreviousPage,
    state,
    pageOptions,
    gotoPage,
    pageCount,
  } = tableInstance;

  const { pageIndex } = state;
  return (
    <>
      {/* Step 4: Define the Table with basic HTML */}
      <table {...getTableProps()} className="customers">
        <thead>
          {headerGroups.map((headerGroup) => (
            <tr {...headerGroup.getHeaderGroupProps()}>
              {headerGroup.headers.map((column) => (
                <th {...column.getHeaderProps()}>{column.render("Header")}</th>
              ))}
            </tr>
          ))}
        </thead>
        <tbody {...getTableBodyProps}>
          {page.map((row) => {
            prepareRow(row);
            return (
              <tr {...row.getRowProps()}>
                {row.cells.map((cell) => {
                  return (
                    <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                  );
                })}
              </tr>
            );
          })}
        </tbody>
      </table>

      <span>
        Page{" "}
        <strong>
          {pageIndex + 1} of {pageOptions.length}
        </strong>
      </span>
      <div>
        <span>
          | Go to page:{" "}
          <input
            type="number"
            defaultValue={pageIndex + 1}
```

```
            onChange={(e) => {
              const pageNumber = e.target.value
                ? Number(e.target.value) - 1
                : 0;
              gotoPage(pageNumber);
            }}
            style={{ width: '50px'}}
          />
        </span>
        <button onClick={() => gotoPage(0)} disabled={!canPreviousPage}>
          {"<<"}
        </button>
        <button onClick={() => previousPage()} disabled={!canPreviousPage}>
          Previous
        </button>
        <button onClick={() => nextPage()} disabled={!canNextPage}>
          Next
        </button>
        <button onClick={() => gotoPage(pageCount - 1)} disabled={!canNextPage}>
          {">>"}
        </button>
      </div>
    </>
  );
};

export default PaginationTable;
```

Chapter 13 : Pagination (Page Size)

- We can set the page size, how many rows we need to have in the table.
- Destructure **setPageSize** from useTable hook.
- Destructure **pageSize** from state.
- Define the predefined value where user can select from dropdown.

PaginationTable.js:

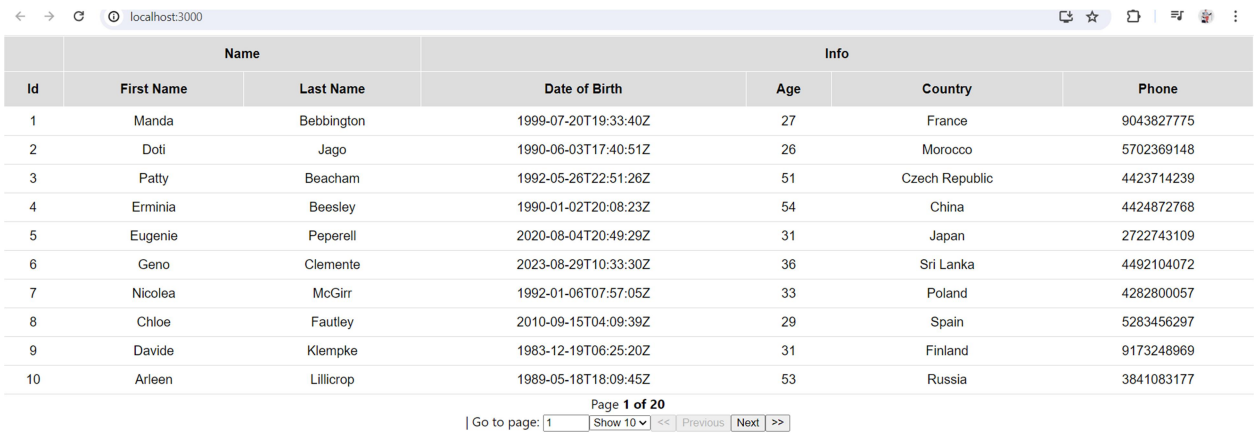
```
import React, { useMemo } from "react";
// Step 1 : To import
import { useTable, usePagination } from "react-table";
import { COLUMNS, GROUP_COLUMNS } from "./columns";
import MOCK_DATA from "../MOCK_DATA.json";

import "../table.css";

const PaginationTable = () => {
```

```
// Step 2: To get the Column and set the data
//Here, Data is not created on every render
// const columns = useMemo(() => COLUMNS, []);
const columns = useMemo(() => GROUP_COLUMNS, []);

const data = useMemo(() => MOCK_DATA, []);

// Step 3: Create the instance of the table
const tableInstance = useTable(
  {
    /*     columns: columns,
  data: data, */
    columns,
    data,
  },
  usePagination
);

const {
  getTableProps,
  getTableBodyProps,
  headerGroups,
  page,
  prepareRow,
  nextPage,
  previousPage,
  canNextPage,
  canPreviousPage,
  state,
  pageOptions,
  gotoPage,
  pageCount,
  setPageSize,
} = tableInstance;

const { pageIndex, pageSize } = state;
return (
  <>
    {/* Step 4: Define the Table with basic HTML */}
    <table {...getTableProps()} className="customers">
      <thead>
        {headerGroups.map((headerGroup) => (
          <tr {...headerGroup.getHeaderGroupProps()}>
            {headerGroup.headers.map((column) => (
              <th {...column.getHeaderProps()}>{column.render("Header")}</th>
            ))}
          </tr>
        ))}
      </thead>
      <tbody {...getTableBodyProps}>
        {page.map((row) => {
          prepareRow(row);
          return (
            <tr {...row.getRowProps()}>
```

```
              {row.cells.map((cell) => {
                return (
                  <td {...cell.getCellProps()}>{cell.render("Cell")}</td>
                );
              })}
            </tr>
          );
        })}
      </tbody>
    </table>

    <span>
      Page{" "}
      <strong>
        {pageIndex + 1} of {pageOptions.length}
      </strong>
    </span>
    <div>
      <span>
        | Go to page:{" "}
        <input
          type="number"
          defaultValue={pageIndex + 1}
          onChange={(e) => {
            const pageNumber = e.target.value
              ? Number(e.target.value) - 1
              : 0;
            gotoPage(pageNumber);
          }}
          style={{ width: "50px" }}
        />
      </span>

      <select
        value={pageSize}
        onChange={(e) => setPageSize(Number(e.target.value))}
      >
        {[10, 25, 50].map((pageSize) => (
          <option key={pageSize} value={pageSize}>
            Show {pageSize}
          </option>
        ))}
        <options></options>
      </select>
      <button onClick={() => gotoPage(0)} disabled={!canPreviousPage}>
        {"<<"}
      </button>
      <button onClick={() => previousPage()} disabled={!canPreviousPage}>
        Previous
      </button>
      <button onClick={() => nextPage()} disabled={!canNextPage}>
        Next
      </button>
      <button onClick={() => gotoPage(pageCount - 1)} disabled={!canNextPage}>
```

```
          {">>"}
        </button>
      </div>
    </>
  );
};

export default PaginationTable;
```

Ouput:



Chapter 14 : Selecting Rows

- In some condition, we have to select 1 or more rows and send the data to the API.
- Create RowSelection.js, copy the content from BasicTable.js
- We can select only 10 rows, we say **rows.slice(0,10);**
- To select the row, it is better to use checkbox element. So create Checkbox.js
- We can use '**useRowSelect**' from useTable and destructure '**selectedFlatRows**'