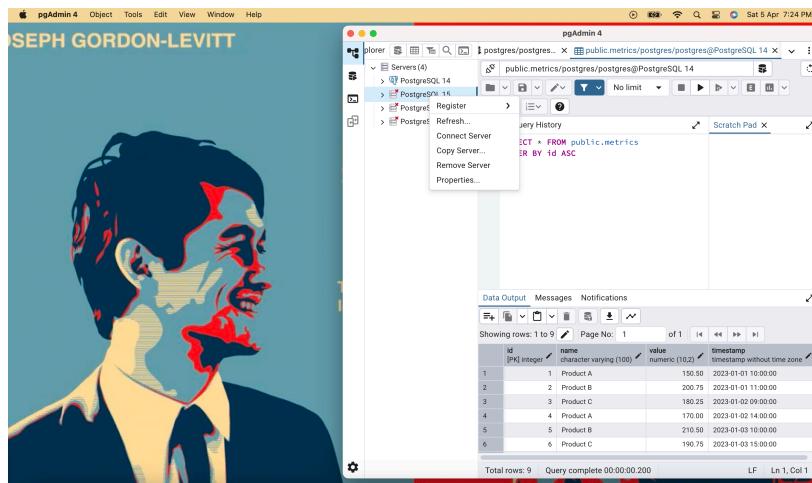


Assignment Documentation

1. Install PostgreSQL

Install PostgreSQL from this site <https://www.postgresql.org/download/>

Once downloading completed then create a server and connect the server with this



Once connected to server then create a database with command

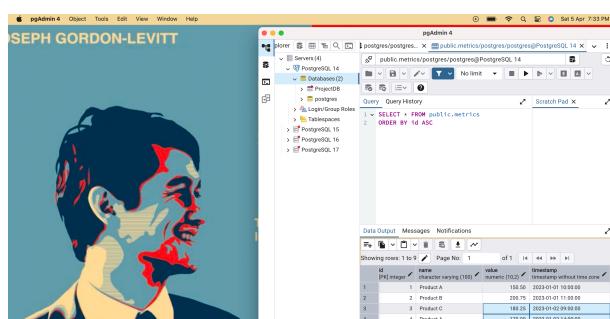
```
CREATE DATABASE databases
```

Once this database created then create a table name Postgres by right clicking on this database and going to query tool

```
CREATE TABLE metrics (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    value NUMERIC,
    timestamp TIMESTAMP
)
```

Once this table created then add data to table by following command

```
INSERT INTO metrics (id, name, value, timestamp) VALUES
(1, 'Product A', 150.50, '2023-01-01 10:00:00'),
(2, 'Product B', 200.75, '2023-01-01 11:00:00'),
(3, 'Product C', 180.25, '2023-01-02 09:00:00'),
(4, 'Product A', 170.00, '2023-01-02 14:00:00'),
(5, 'Product B', 210.50, '2023-01-03 10:00:00'),
(6, 'Product C', 190.75, '2023-01-03 15:00:00);
```



2. Cubejs setup and install

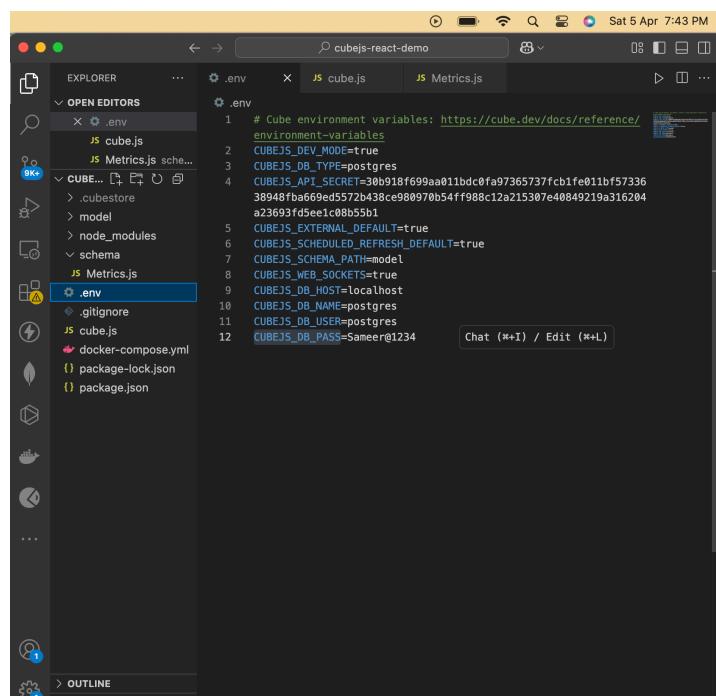
First install our cubejs into out system `npm install -g cubejs-cli`

Once install then create folder enter this command into my terminal



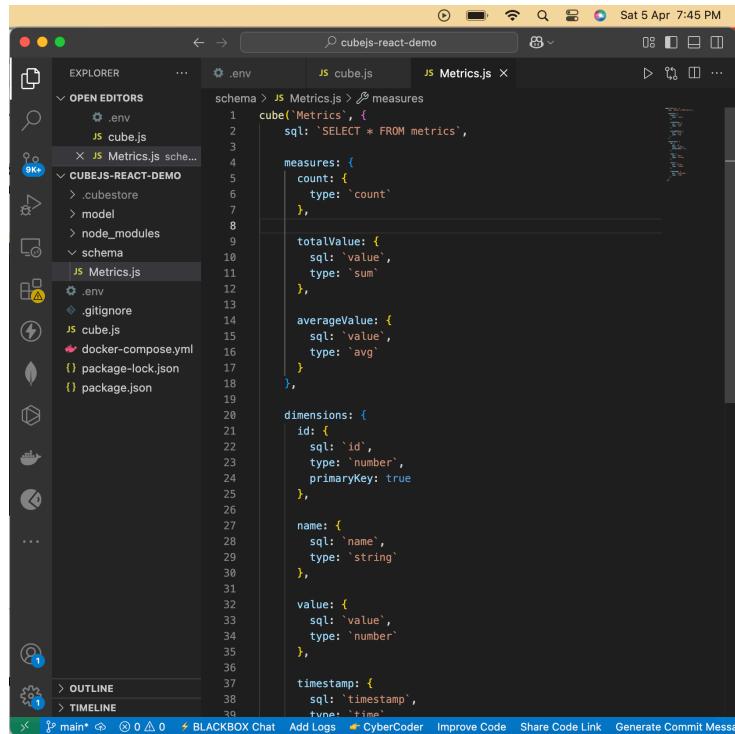
```
Last login: Sat Apr  5 19:37:08 on ttys006
(base) sameerraj@Sameers-MacBook-Air webdevelopment % npx cubejs-cli create cube
js-react-demo -d postgres
```

Once this project is created then enter .env data into our like Database name, database url and many more thing



```
# Cube environment variables: https://cube.dev/docs/reference/
CUBEJS_DEV_MODE=true
CUBEJS_DB_TYPE=postgres
CUBEJS_API_SECRET=30b918f699aa011bdc0fa97365737fc1fe011bf57336
38948fb4669ed5572b438ce9800970b54ff988c12a215307e40849219a316204
a23693fd5ee1c08b55b1
CUBEJS_EXTERNAL_DEFAULT=true
CUBEJS_SCHEDULED_REFRESH_DEFAULT=true
CUBEJS_SCHEMA_PATH=model
CUBEJS_WEB_SOCKETS=true
CUBEJS_DB_HOST=localhost
CUBEJS_DB_NAME=postgres
CUBEJS_DB_USER=postgres
CUBEJS_DB_PASS=Sameer@1234
```

Then create schema with creating schema folder and in this folder create Metrics.js



```
schema > JS Metrics.js > measures
1  cube('Metrics', {
2    sql: `SELECT * FROM metrics`,
3
4    measures: {
5      count: {
6        type: 'count'
7      },
8
9      totalValue: {
10        sql: 'value',
11        type: 'sum'
12      },
13
14      averageValue: {
15        sql: 'value',
16        type: 'avg'
17      }
18    },
19
20    dimensions: {
21      id: {
22        sql: 'id',
23        type: 'number',
24        primaryKey: true
25      },
26
27      name: {
28        sql: 'name',
29        type: 'string'
30      },
31
32      value: {
33        sql: 'value',
34        type: 'number'
35      },
36
37      timestamp: {
38        sql: 'timestamp',
39        type: 'time'
40      }
41    }
42  }
43}
```

This metrics.js explains

1. `cube('Metrics', { ... })`: This defines a new cube named Metrics. The name of the cube is typically the same as the table name in the database, but it can be different if needed.

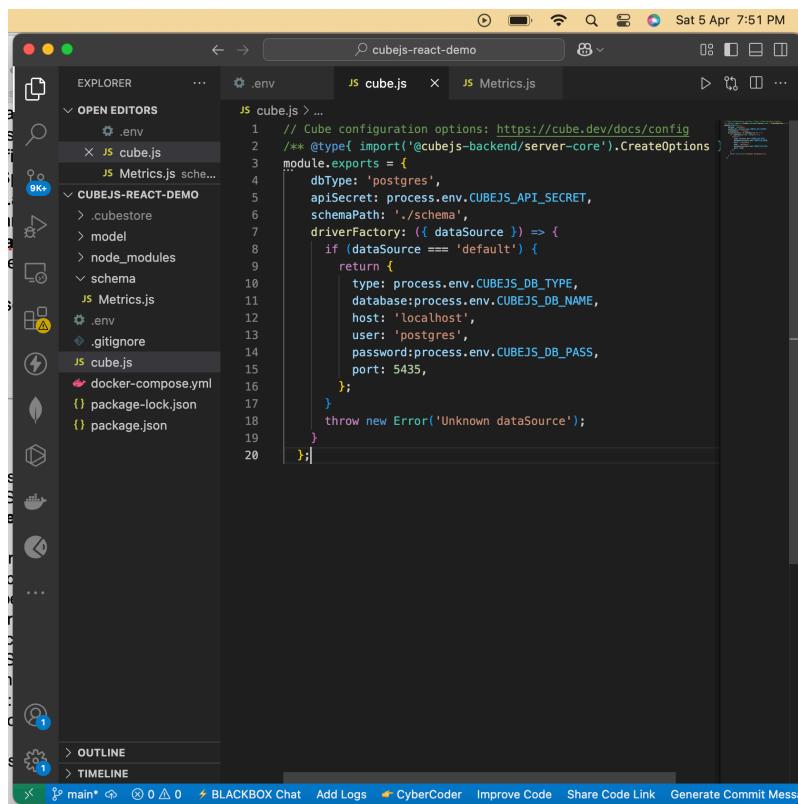
2. `sql: 'SELECT * FROM metrics'`: This specifies the SQL query that Cube.js will use to fetch data from the database. In this case, it selects all columns from the metrics table.

3. `count`: This measure counts the number of rows in the metrics table
`type: 'count'`: Specifies that this measure should count the number of rows.
`totalValue`: This measure calculates the sum of the value column.
`sql: 'value'`: Specifies the column to use for the aggregation.
`type: 'sum'`: Specifies that this measure should sum the values in the value column.
`averageValue`: This measure calculates the average of the value column.
`sql: 'value'`: Specifies the column to use for the aggregation.
`type: 'avg'`: Specifies that this measure should calculate the average of the values in the value column.

4. `id`: This dimension represents the id column.

- **sql: 'id'**: Specifies the column to use.
- **type: 'number'**: Specifies that this dimension is of type number.
- **primaryKey: true**: Specifies that this dimension is the primary key of the table.
- **name**: This dimension represents the name column.
 - **sql: 'name'**: Specifies the column to use.
 - **type: 'string'**: Specifies that this dimension is of type string.
- **value**: This dimension represents the value column.
 - **sql: 'value'**: Specifies the column to use.
 - **type: 'number'**: Specifies that this dimension is of type number.
- **timestamp**: This dimension represents the timestamp column.
 - **sql: 'timestamp'**: Specifies the column to use.
 - **type: 'time'**: Specifies that this dimension is of type time.

Once you created this metrics schema then change this following things in cube.js file



```

// Cube configuration options: https://cube.dev/docs/config
/** @type{ import('@cubejs-backend/server-core').CreateOptions } */
module.exports = {
  dbType: 'postgres',
  apiSecret: process.env.CUBEJS_API_SECRET,
  schemaPath: './schema',
  driverFactory: ({ dataSource }) => {
    if (dataSource === 'default') {
      return {
        type: process.env.CUBEJS_DB_TYPE,
        database:process.env.CUBEJS_DB_NAME,
        host: 'localhost',
        user: 'postgres',
        password:process.env.CUBEJS_DB_PASS,
        port: 5435,
      };
    }
    throw new Error(`Unknown dataSource`);
  }
};

```

3. React App frontend

Create frontend app with following command

```
Last login: Sat Apr  5 19:37:08 on ttys006
(base) sameerraj@Sameers-MacBook-Air webdevelopment % npm create vite@latest cubejs-react-frontend -- --template react
```

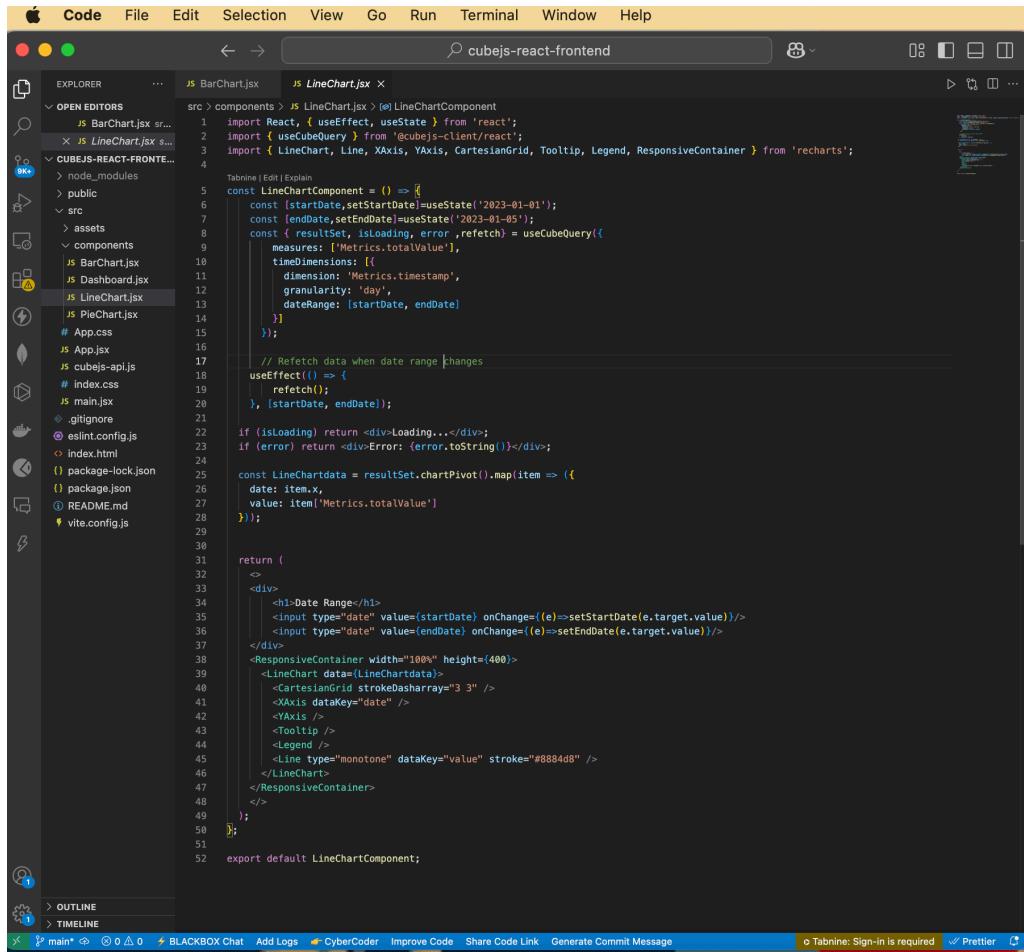
Once this app is created install following things

```
Last login: Sat Apr  5 19:37:08 on ttys006
(base) sameerraj@Sameers-MacBook-Air webdevelopment % npm install @cubejs-client/core @cubejs-client/react recharts antd react-router-dom
```

Once this created then setup cubejsApi with following things

```
src > JS App.jsx > App
1 import React from 'react';
2 import { CubeProvider } from '@cubejs-client/react';
3 import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
4 import { Layout, Menu } from 'antd';
5 import cubejsApi from './cubejs-api';
6 import LineChartComponent from './components/LineChart';
7 import BarChartComponent from './components/BarChart';
8 import PieChartComponent from './components/PieChart';
9 import cube from '@cubejs-client/core';
10 import Dashboard from './components/Dashboard';
11 import './App.css';
12
13 const { Header, Content, Footer } = Layout;
14
15 const apiUrl = 'http://localhost:4000';
16 const cubeToken = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJpXQiOjE3NDM4Mw';
17
18 const cubeApi = cube(cubeToken, {
19   apiUrl: `${apiUrl}/cubejs-api/v1`
20 });
21
22 function App() {
23   console.log("cube js api", cubejsApi)
24   return [
25     <CubeProvider cubeApi={cubeApi}>
26       <Router>
27         <Layout className="layout">
28           <Header>
29             <Menu theme="dark" mode="horizontal" defaultSelectedKeys={['1']}
30               <MenuItem key="1"><Link to="/">Dashboard</Link></MenuItem>
31               <MenuItem key="2"><Link to="/line">Line Chart</Link></MenuItem>
32               <MenuItem key="3"><Link to="/bar">Bar Chart</Link></MenuItem>
33               <MenuItem key="4"><Link to="/pie">Pie Chart</Link></MenuItem>
34             </Menu>
35           </Header>
36           <Content style={{ padding: '0 50px' }}>
37             <div style={{ margin: '16px 0', minHeight: '80vh' }}>
38               <Routes>
```

Once this is created then create a line chart component with following code where we can change the range with the help of startDate and endDate state accordingly code is

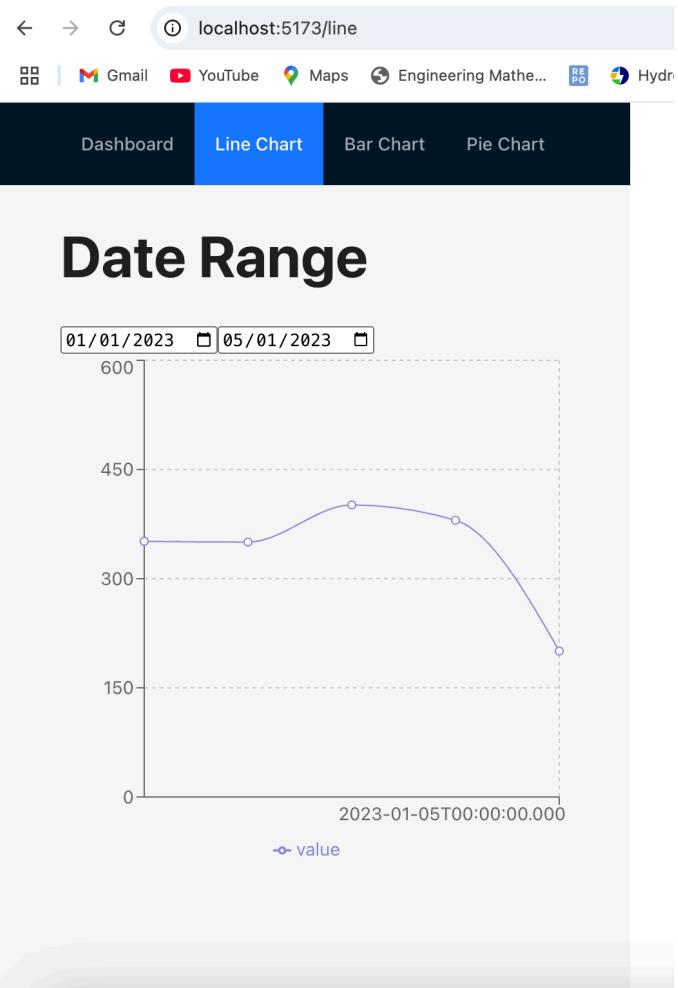


```
Code File Edit Selection View Go Run Terminal Window Help
OPEN EDITORS JS BarChart.jsx JS LineChart.jsx ...
src > components > JS LineChart.jsx > LineChartComponent
1 import React, { useEffect, useState } from 'react';
2 import { useCubeQuery } from '@cubejs-client/react';
3 import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts';
4
5 Tabnine (Edit Explain)
const LineChartComponent = () => {
6   const [startDate, setStartDate] = useState('2023-01-01');
7   const [endDate, setEndDate] = useState('2023-01-05');
8   const { resultSet, isLoading, error, refetch } = useCubeQuery({
9     measures: ['Metrics.totalValue'],
10    timeDimensions: [
11      dimension: 'Metrics.timestamp',
12      granularity: 'day',
13      dateRange: [startDate, endDate]
14    ]
15  });
16
17  // Refetch data when date range changes
18  useEffect(() => {
19    refetch();
20  }, [startDate, endDate]);
21
22  if (isLoading) return <div>Loading...</div>;
23  if (error) return <div>Error: <code>{error.toString()}</code></div>;
24
25  const LineChartData = resultSet.chartPivot().map(item => ({
26    date: item.x,
27    value: item['Metrics.totalValue']
28  }));
29
30
31  return (
32    <>
33    <div>
34      <h1>Date Range</h1>
35      <input type="date" value={startDate} onChange={(e)>setStartDate(e.target.value)} />
36      <input type="date" value={endDate} onChange={(e)>setEndDate(e.target.value)} />
37    </div>
38    <ResponsiveContainer width="100%" height={400}>
39      <LineChart data={LineChartData}>
40        <CartesianGrid strokeDasharray="3 3" />
41        <XAxis dataKey="date" />
42        <YAxis />
43        <Tooltip />
44        <Legend />
45        <Line type="monotone" dataKey="value" stroke="#8884d8" />
46      </LineChart>
47    </ResponsiveContainer>
48  );
49}
50
51 export default LineChartComponent;
```

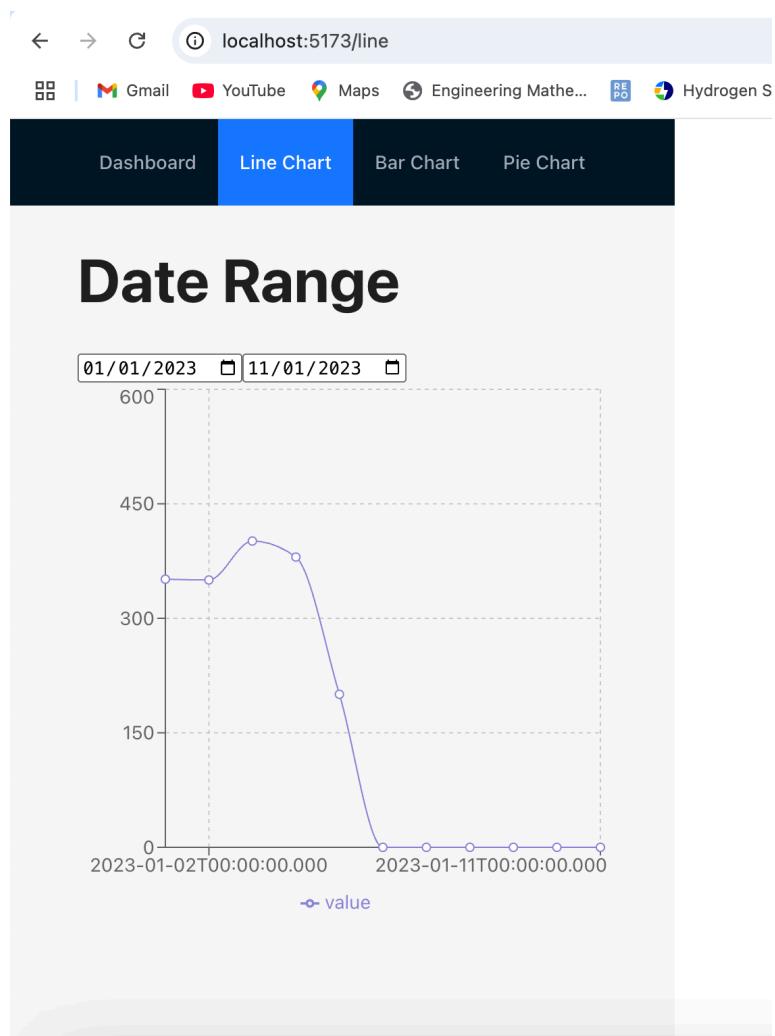
Once we start our server we get this output first here date range

```
const [startDate, setStartDate] = useState('2023-01-01');
const [endDate, setEndDate] = useState('2023-01-05');
```

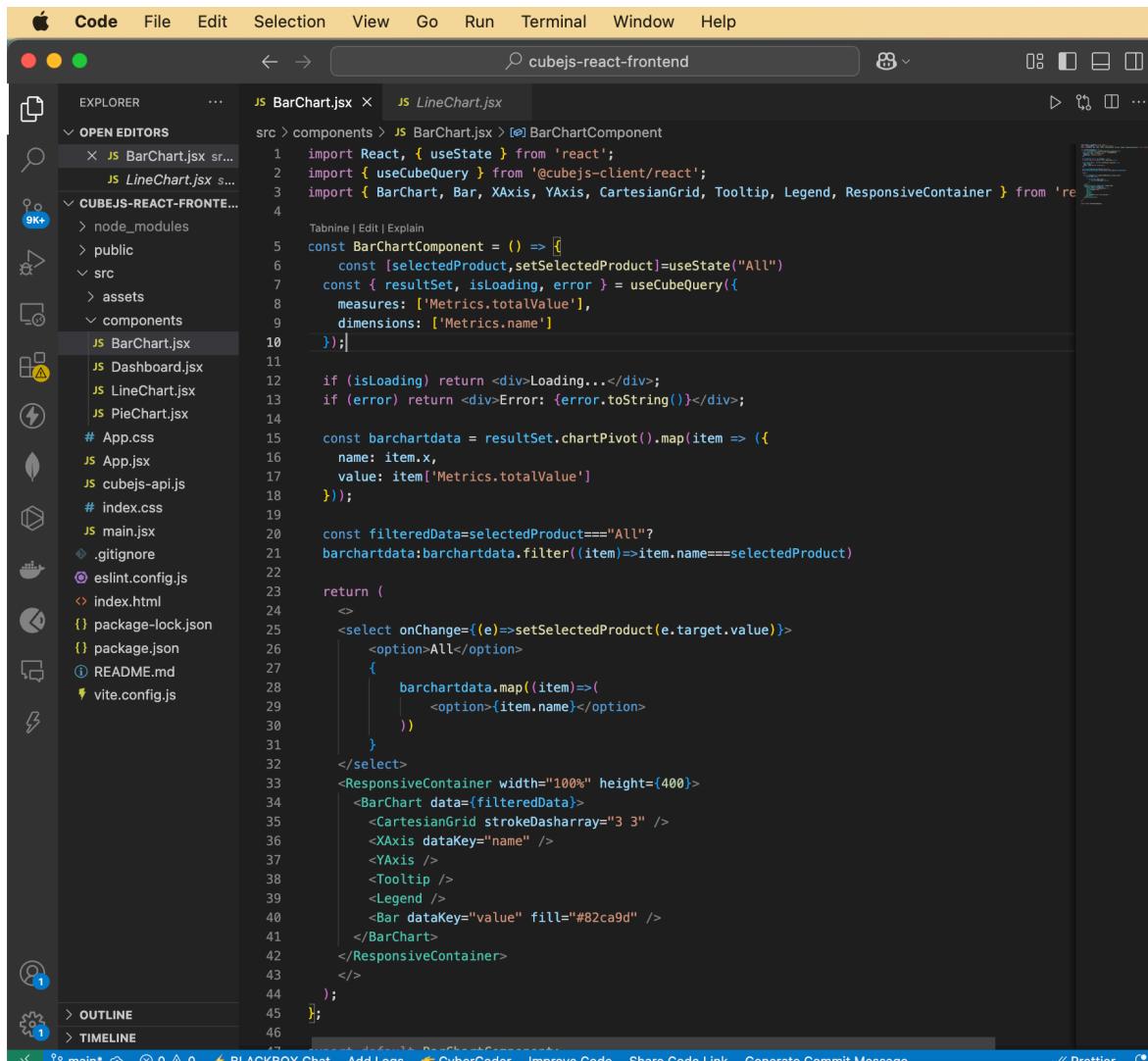
Then with no range set this will be our output



And when we set start and end date different then this will be our output



For barChart this is our code



The screenshot shows a code editor window with the title "cubejs-react-frontend". The left sidebar displays a file tree for a project named "CUBEJS-REACT-FRONTEND". The "src" folder contains several files: BarChart.jsx, Dashboard.jsx, LineChart.jsx, PieChart.jsx, App.css, App.jsx, cubejs-api.js, index.css, main.jsx, .gitignore, eslint.config.js, index.html, package-lock.json, package.json, README.md, and vite.config.js. The "BarChart.jsx" file is open in the editor, showing the following code:

```
import React, { useState } from 'react';
import { useCubeQuery } from '@cubejs-client/react';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 'recharts';

const BarChartComponent = () => [
  const [selectedProduct, setSelectedProduct] = useState("All");
  const { resultSet, isLoading, error } = useCubeQuery({
    measures: ['Metrics.totalValue'],
    dimensions: ['Metrics.name']
  });

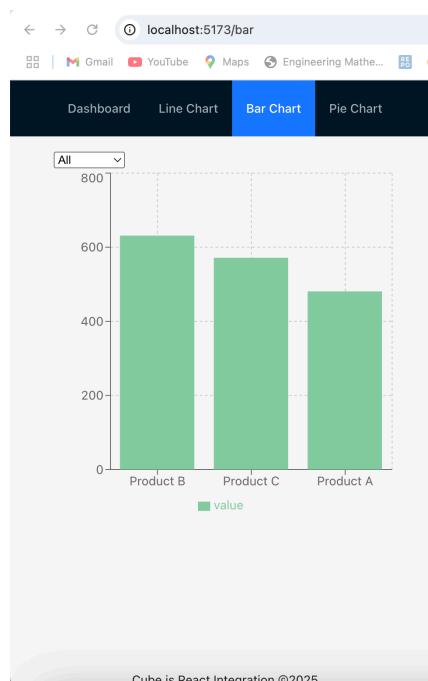
  if (isLoading) return <div>Loading...</div>;
  if (error) return <div>Error: {error.toString()}</div>;

  const barchartdata = resultSet.chartPivot().map(item => ({
    name: item.x,
    value: item['Metrics.totalValue']
  }));

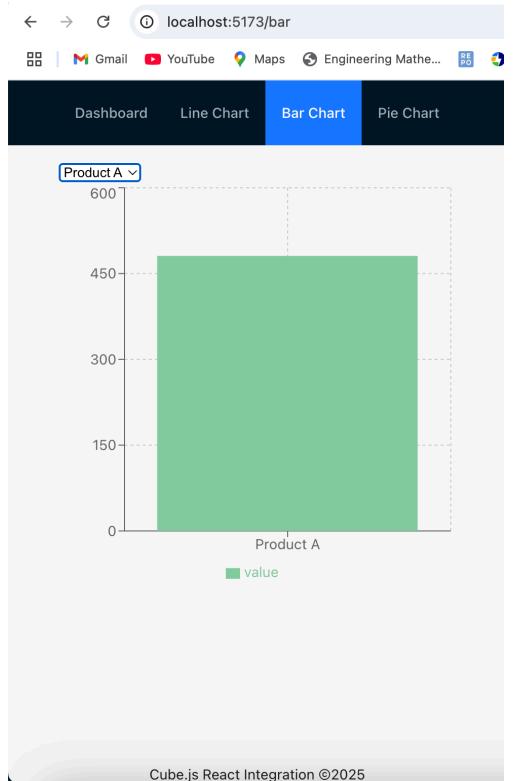
  const filteredData = selectedProduct === "All" ?
    barchartdata : barchartdata.filter(item => item.name === selectedProduct);

  return (
    <>
      <select onChange={(e) => setSelectedProduct(e.target.value)}>
        <option>All</option>
        {
          barchartdata.map(item => (
            <option value={item.name}>{item.name}</option>
          ))
        }
      </select>
      <ResponsiveContainer width="100%" height={400}>
        <BarChart data={filteredData}>
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey="name" />
          <YAxis />
          <Tooltip />
          <Legend />
          <Bar dataKey="value" fill="#82ca9d" />
        </BarChart>
      </ResponsiveContainer>
    </>
  );
};
```

And here we can also change our filter with products and this is my output for filter All



For product A



And this is my pieChart

