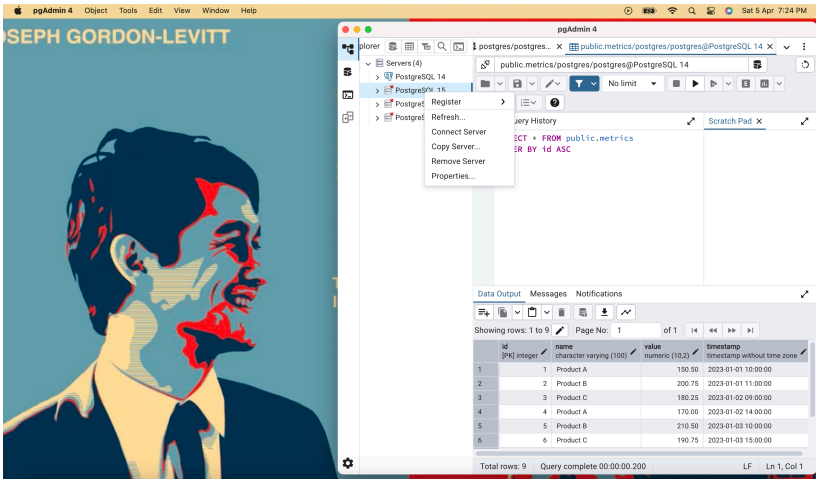
Assignment Documentation

1.Install postgreSQL

Install postgresSQL 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

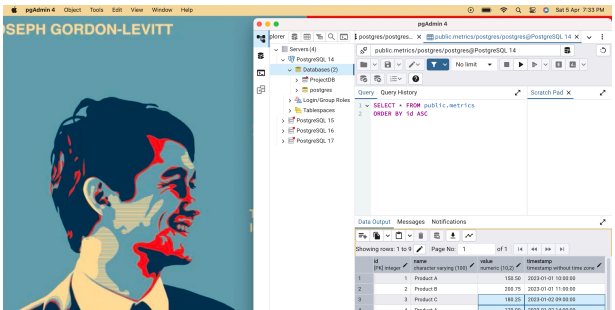
```
CREATE DATABASE databases
```

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');
```

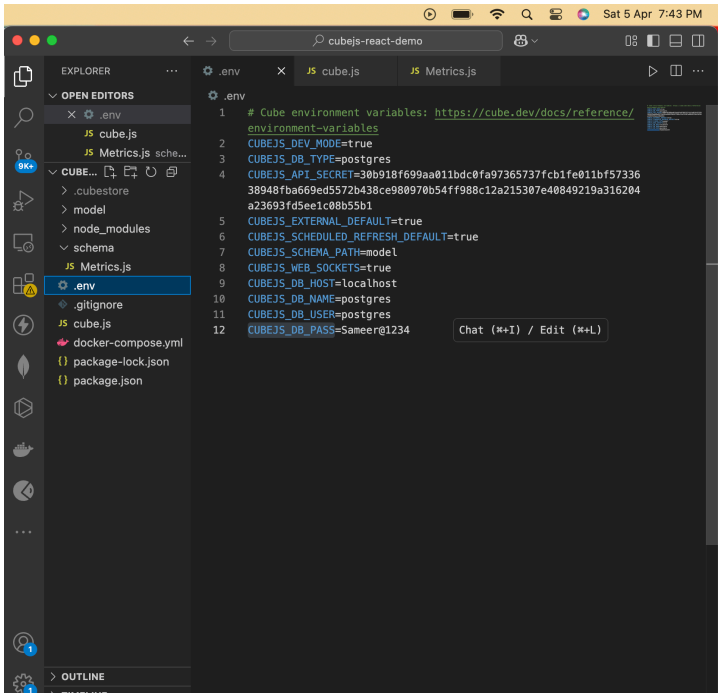


2. Cubejs setup and install

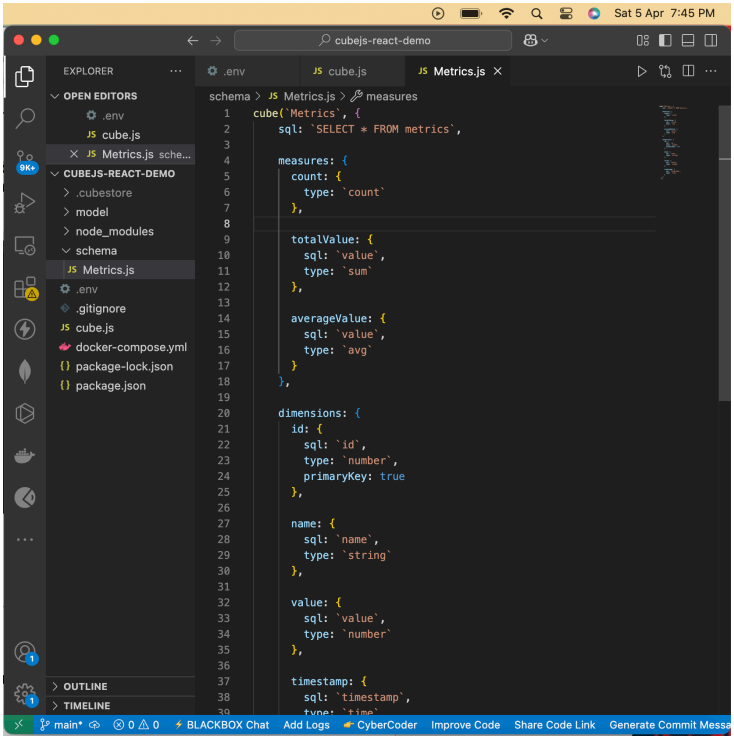
First install our cubejs into out system npm install -g cubejs-cli

Once install then create folder enter this command into my terminal

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



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



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.

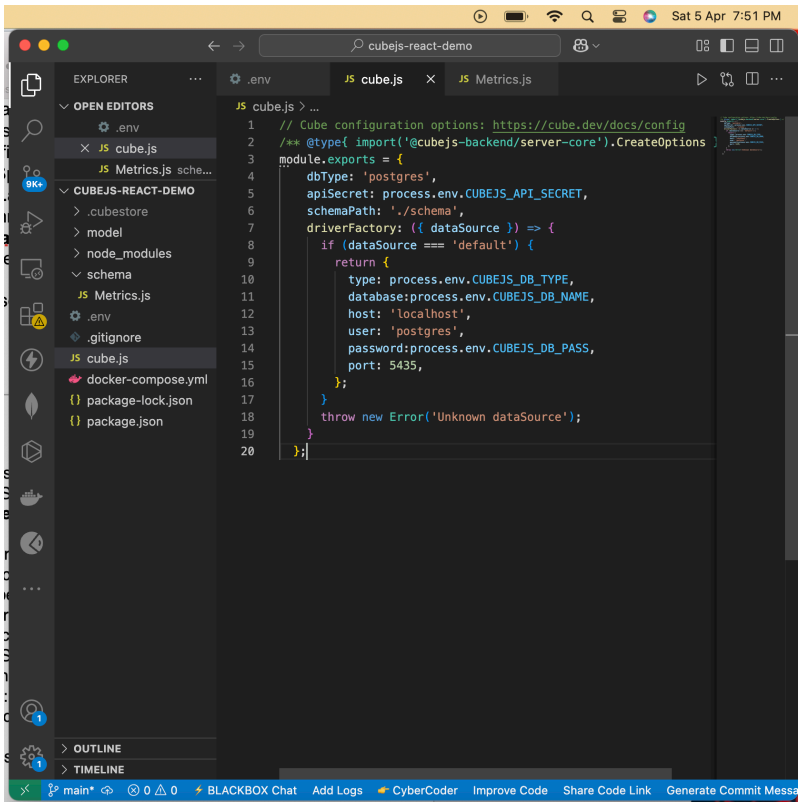
2.**sql: 'SELECT * FROM metrics'**: This specifies the SQL query that Cube.js will use to fetch data from the database. In this case, it selects all columns from the metrics table.

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

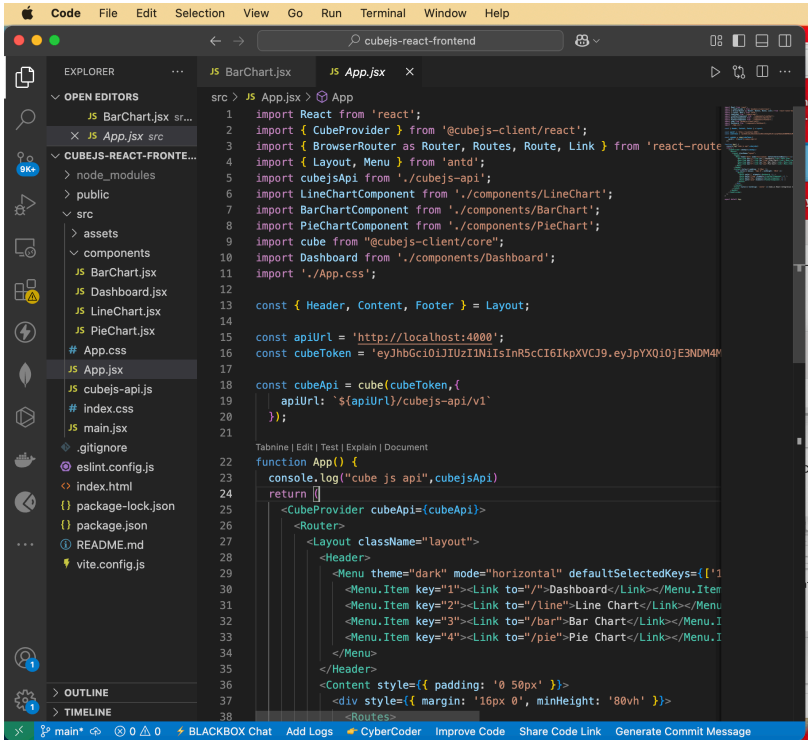


3. React App frontend

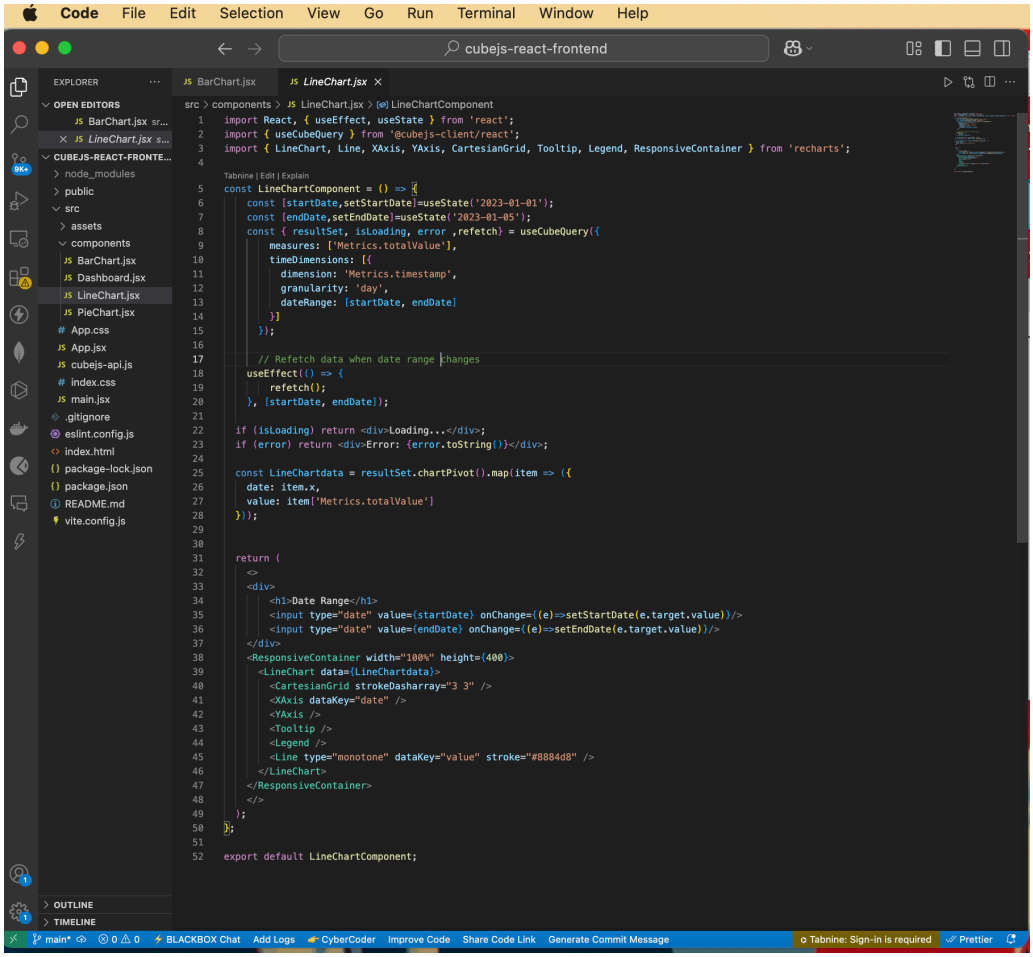
Create frontend app with following command

Once this app is created install following things

Once this created then setup cubejsApi with following things



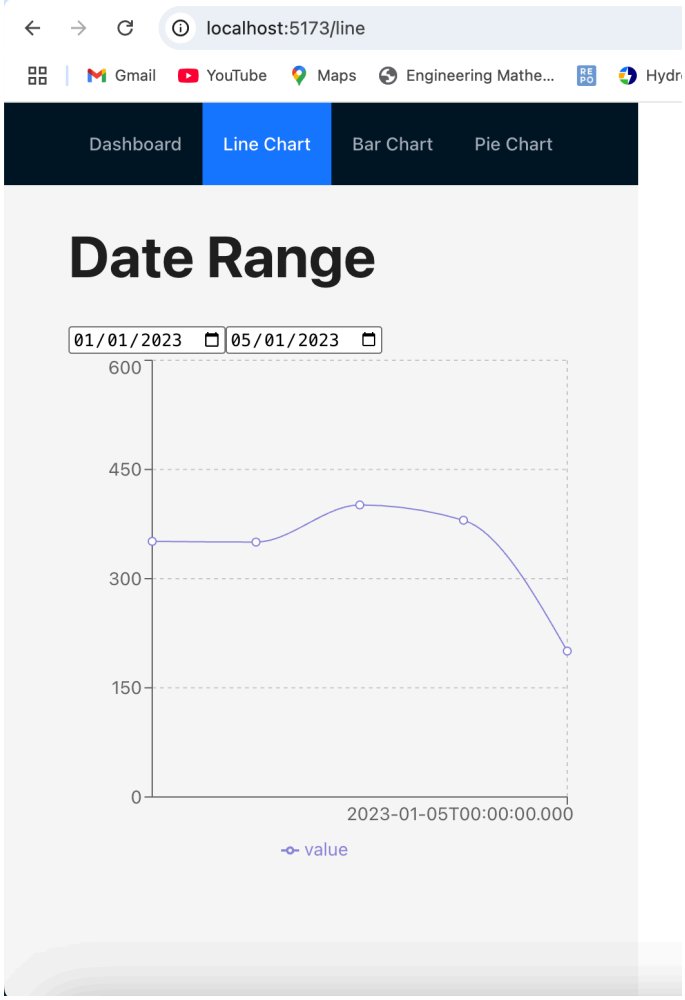
Once this is created then create a line chart component with following code where we can change the range with the help of startData and endDate state accordingly code is



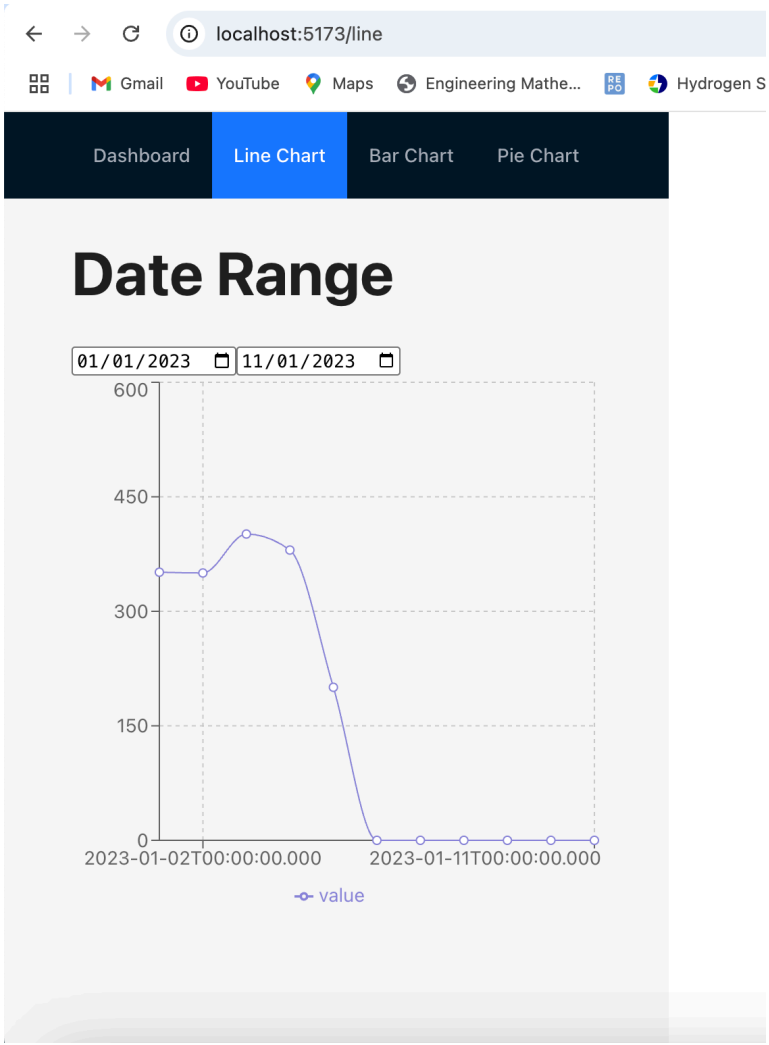
Once we start our server we get this output first here date range

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

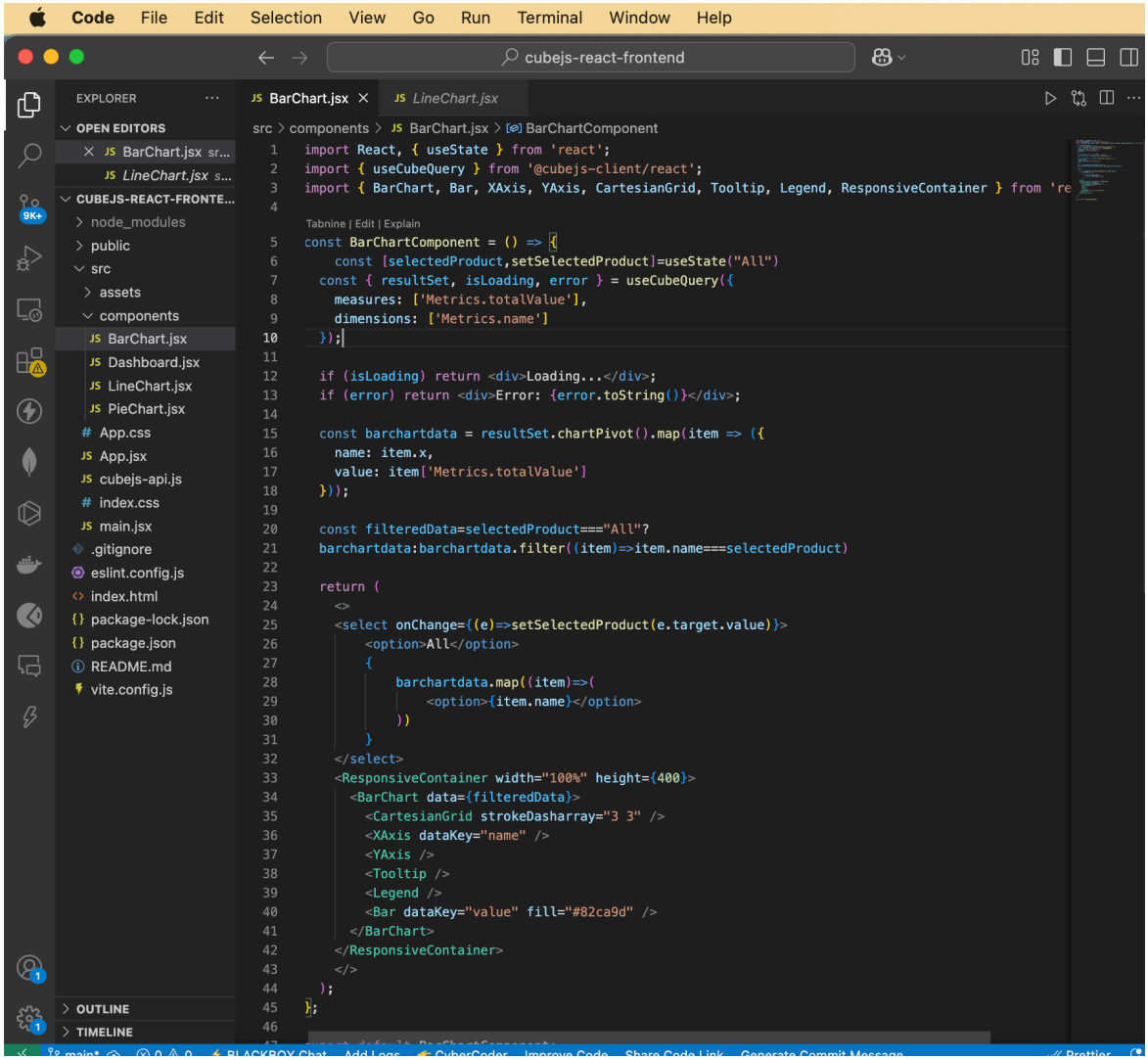
Then with no range set this will be our output



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

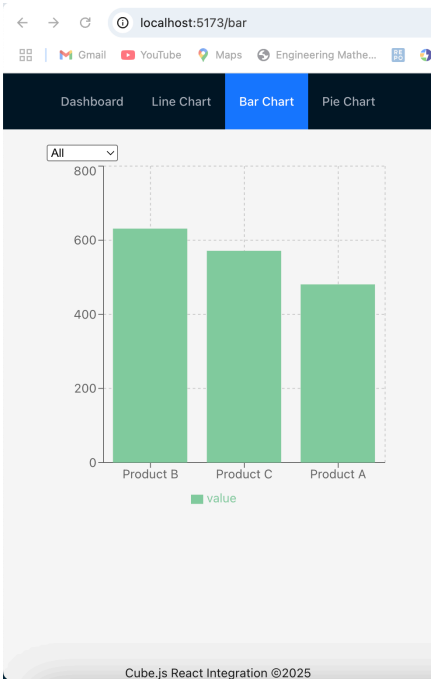


For barChart this is our code

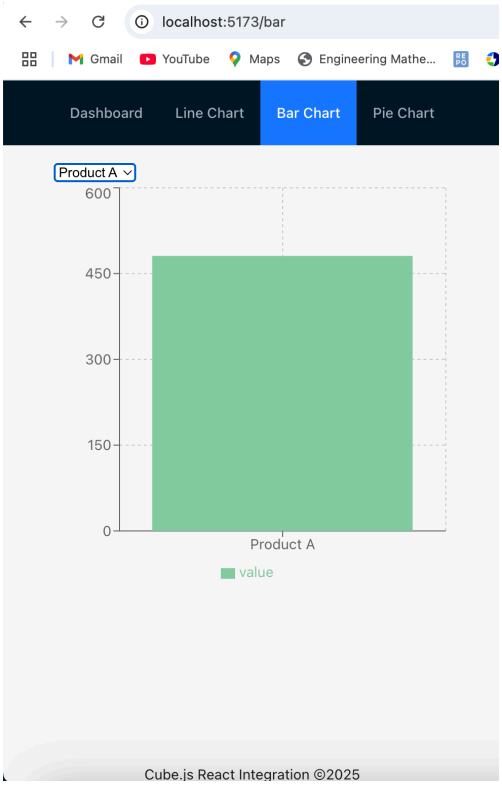


```jsx
import React, { useState } from 'react';
import { useCubeQuery } from '@cubejs-client/react';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer } from 're

Tabnine | Edit | Explain
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>{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