

types.ts X

```
751 export type Country = {
752   __typename?: 'Country';
753   _id?: Maybehttp
756   alpha2Code: Scalars['String'];
757   /** ISO 3166-1 alpha-3 codes are three-letter country codes defined in ISO 3166-1, part of the IS
758   (ISO), to represent countries, dependent territories, and special areas of geographical interest.
759   alpha3Code: Scalars['String'];
760   alternativeSpellings?: Maybe<Array<Maybe<AlternativeSpelling>>>;
761   /** The area in square kilometer, you can convert the area unit and population density through th
762   area?: Maybe<Scalars['Float']>;
763   borders?: Maybe<Array<Maybe<Country>>>;
764   callingCodes?: Maybe<Array<Maybe<CallingCode>>>;
765   capital: Scalars['String'];
766   convertedArea?: Maybe<Area>;
767   currencies?: Maybe<Array<Maybe<Currency>>>;
768   /** identifies residents or natives of a particular place, usually derived from the name of the p
769   demonym: Scalars['String'];
770   distanceToOtherCountries?: Maybe<Array<Maybe<DistanceToOtherCountry>>>;
771   flag?: Maybe<Flag>;
772   /** In economics, the Gini coefficient, sometimes called the Gini index or Gini ratio, is a measu
773   distribution of a nation's residents, and is the most commonly used measurement of inequality. ht
774   gini?: Maybe<Scalars['Float']>;
775   location: _Neo4jPoint;
776   name: Scalars['String'];
777   nameTranslation?: Maybe<Scalars['String']>;
778   nameTranslations?: Maybe<Array<Maybe<Translation>>>;
779   nativeName: Scalars['String'];
780   numericCode?: Maybe<Scalars['String']>;
781   officialLanguages?: Maybe<Array<Maybe<Language>>>;
782   population: Scalars['Float'];
783   /** The population per square kilometer */
784   populationDensity?: Maybe<Scalars['Float']>;
```

CountryList.tsx X

```
1  import { gql, useQuery } from '@apollo/client';
2  import { Country } from '../__generated__/types';
3  import { Loader, ErrorScreen, CountryCard } from './components';
4
5  const COUNTRY_LIST = gql`
6    query CountryList {
7      Execute Query
8      countries: Country {
9        _id
10       name
11       subregion {
12         _id
13         name
14       }
15     }
16 `;
17
18 type CountryList = {
19   countries: Country[];
20 };
21
22 const CountriesPage: React.FC = () => {
23   const { data, loading, error } = useQuery<CountryList>(COUNTRY_LIST);
24
25   if (loading) {
26     return <Loader />;
27   }
28
29   if (error) {
30     return <ErrorScreen message={error.message} />;
31   }
32
33   return (
```