types.ts

```ts
export type Country = {
  __typename?: 'Country';
  _id?: Maybe<Scalars['String']>;
  /** ISO 3166-1 alpha-2 codes are two-letter country codes defined in ISO 3166-1, part of the ISO
  , to represent countries, dependent territories, and special areas of geographical interest. http
  alpha2Code: Scalars['String'];
  /** ISO 3166-1 alpha-3 codes are three-letter country codes defined in ISO 3166-1, part of the IS
  (ISO), to represent countries, dependent territories, and special areas of geographical interest.
  alpha3Code: Scalars['String'];
  alternativeSpellings?: Maybe<Array<Maybe<AlternativeSpelling>>>;
  /** The area in square kilometer, you can convert the area unit and population density through th
  area?: Maybe<Scalars['Float']>;
  borders?: Maybe<Array<Maybe<Country>>>;
  callingCodes?: Maybe<Array<Maybe<CallingCode>>>;
  capital: Scalars['String'];
  convertedArea?: Maybe<Area>;
  currencies?: Maybe<Array<Maybe<Currency>>>;
  /** identifies residents or natives of a particular place, usually derived from the name of the p
  demonym: Scalars['String'];
  distanceToOtherCountries?: Maybe<Array<Maybe<DistanceToOtherCountry>>>;
  flag?: Maybe<Flag>;
  /** In economics, the Gini coefficient, sometimes called the Gini index or Gini ratio, is a measu
  distribution of a nation's residents, and is the most commonly used measurement of inequality. ht
  gini?: Maybe<Scalars['Float']>;
  location: _Neo4jPoint;
  name: Scalars['String'];
  nameTranslation?: Maybe<Scalars['String']>;
  nameTranslations?: Maybe<Array<Maybe<Translation>>>;
  nativeName: Scalars['String'];
  numericCode?: Maybe<Scalars['String']>;
  officialLanguages?: Maybe<Array<Maybe<Language>>>;
  population: Scalars['Float'];
  /** The population per square kilometer */
  populationDensity?: Maybe<Scalars['Float']>;
```

CountryList.tsx

```tsx
import { gql, useQuery } from '@apollo/client';
import { Country } from '../__generated__/types';
import { Loader, ErrorScreen, CountryCard } from './components';

const COUNTRY_LIST = gql`
  query CountryList {
    Execute Query
    countries: Country {
      _id
      name
      subregion {
        _id
        name
      }
    }
  }
`;

type CountryList = {
  countries: Country[];
};

const CountriesPage: React.FC = () => {
  const { data, loading, error } = useQuery<CountryList>(COUNTRY_LIST);

  if (loading) {
    return <Loader />;
  }

  if (error) {
    return <ErrorScreen message={error.message} />;
  }

  return (
```