In [ ]:

```
import polars as pl
import altair as alt

# avoids errors from maximum allowed rows in altair
alt.data_transformers.disable_max_rows()
```

Out[ ]:

```
DataTransformerRegistry.enable('default')
```

In [ ]:

```
#read in emissions data
emissions = pl.read_csv('data/emissions_high_granularity.csv', skip_rows = 1


emissions = emissions.with_columns((pl.col("total_emissions_MtCO2e") - pl.co
                                                                         )).alias(

emissions.head(10)
```

Out[ ]: shape: (10, 17)

year	parent_entity	parent_type	reporting_entity	commodity	production_value	pro
i64	str	str	str	str	f64	
2000	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	695.4	"
2001	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	669.8	"
2002	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	616.9	"
2003	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	675.3	"
2004	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	713.6	"
2005	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	839.5	"
2006	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	894.3	"
2007	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	846.8	"
2008	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	888.8	"

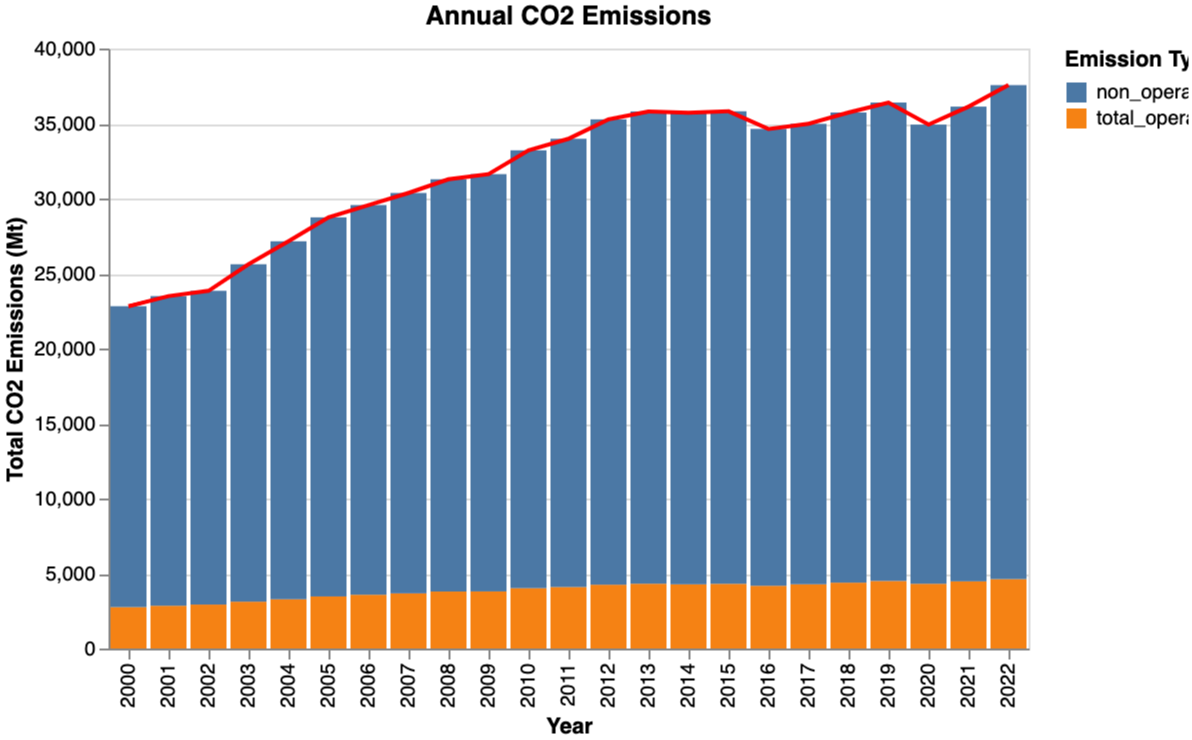
year	parent_entity	parent_type	reporting_entity	commodity	production_value	pro
i64	str	str	str	str	f64	
2009	"Abu Dhabi National Oil Company"	"State-owned Entity"	"Abu Dhabi National Oil Company"	"Oil & NGL"	779.3	"

Annual CO2 Emissions

In [ ]:

```
def annual_emissions(df):

    # find total annual emissions
    df = df.group_by("year"
                    ).agg(pl.col(["total_emissions_MtCO2e","non_operational_

    total_emissions = alt.Chart(df.sort("year"), title = "Annual CO2 Emissio
        ["total_operational_emissions_MtCO2e", "non_operational_emissions_Mt
    ).mark_bar().encode(
        alt.X("year:O").title("Year"),
        alt.Y("value:Q").title("Total CO2 Emissions (Mt)"),
        alt.Color("key:N", title = "Emission Type"),
    )

    mean_line = alt.Chart(df.sort("year"), title = "Average CO2 Emissions").
        alt.X("year:O").title("Year"),
        alt.Y("mean(total_emissions_MtCO2e):Q").title("Total CO2 Emissions (
    )
    return total_emissions + mean_line

annual_emissions(emissions)
```

Out[ ]:

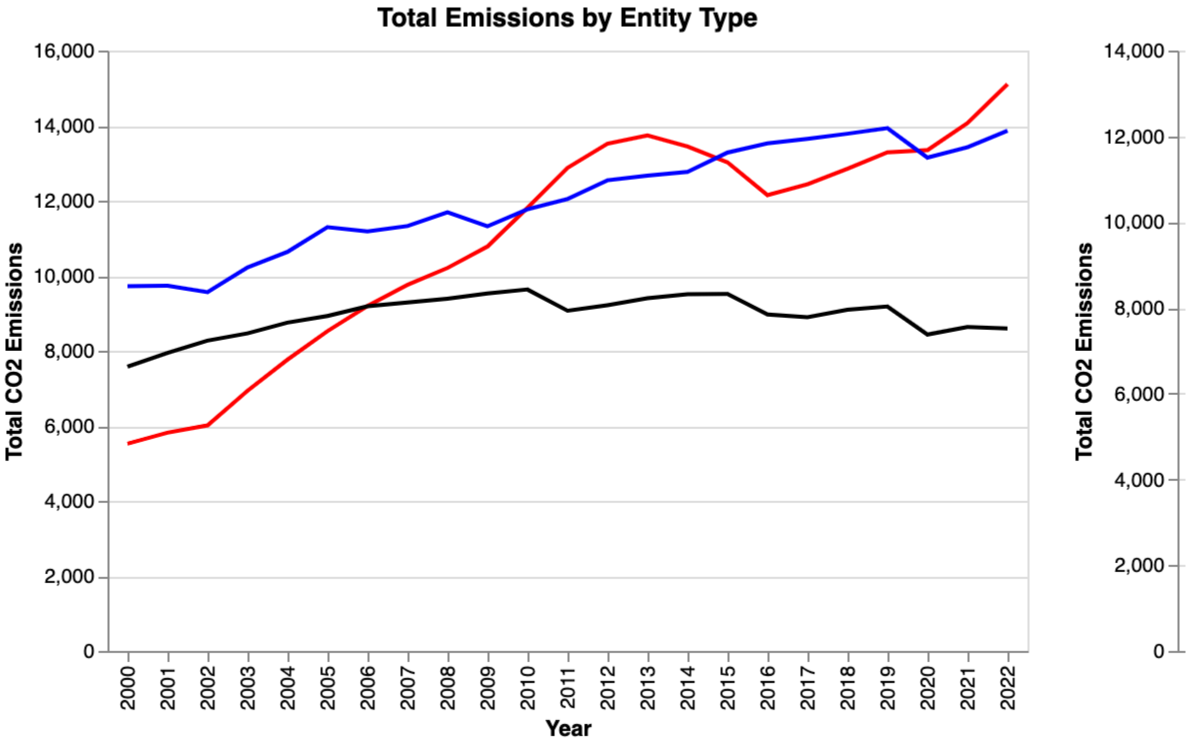


Annual Emissions by Entity Type

In [ ]:

```
def emissions_by_entity_type(df):

    # find total emissions by year & parent_type
    total = df.group_by(["year", "parent_type"]
                    ).agg(pl.col("total_emissions_MtCO2e").sum()
                         ).pivot("parent_type", index = "year", values = "to

    operational = df.group_by(["year", "parent_type"]
                    ).agg(pl.col("total_operational_emissions_MtCO2e").sum()
                         ).pivot("parent_type", index = "year", values = "to

    non_operational = df.group_by(["year", "parent_type"]
                    ).agg(pl.col("non_operational_emissions_MtCO2e").sum()
                         ).pivot("parent_type", index = "year", values = "no

    # set colors for each entity
    color_scale = alt.Scale(domain=['Nation State', 'State-owned Entity', 'I
                            range=['red', 'blue', 'black'])

    # develop chart by entities
    total_emissions_by_types = alt.Chart(total.sort("year"), title = "Total
        ['Nation State', 'State-owned Entity', 'Investor-owned Company'],
    ).mark_line().encode(
        alt.X("year:O").title("Year"),
        alt.Y("value:Q").title("Total CO2 Emissions"),
        alt.Color("key:N", scale = color_scale, title = "Entity Type")

    )
    op_emissions_by_types = alt.Chart(operational.sort("year"), title = "Ope
        ['Nation State', 'State-owned Entity', 'Investor-owned Company'],
    ).mark_line().encode(
        alt.X("year:O").title("Year"),
        alt.Y("value:Q", scale=alt.Scale(domain=[0, 14000])).title("Total CO
        alt.Color("key:N", scale = color_scale, title = "Entity Type")

    )

    non_op_emissions_by_types = alt.Chart(non_operational.sort("year"), titl
        ['Nation State', 'State-owned Entity', 'Investor-owned Company'],
    ).mark_line().encode(
        alt.X("year:O").title("Year"),
        alt.Y("value:Q", scale=alt.Scale(domain=[0, 14000])).title("Total CO
        alt.Color("key:N", scale = color_scale, title = "Entity Type")

    )

    all_emissions_by_type = total_emissions_by_types | op_emissions_by_types
    """ below code uses the layering approach, does not allow for use of a l
    # find total emissions by year & parent_type
    # df = df.group_by(["year", "parent_type"]).agg(pl.col("total_emissions_
    # df = df.filter(pl.col("year") > 2000)
    # print(df.sort("year"))

    # # create Nation State Chart
    # nation_states = alt.Chart(df.filter(pl.col("parent_type") == "Nation S
    # nation_states = nation_states.mark_line(color = "red").encode(
    #     alt.X("year:O", title = "Year"),
    #     alt.Y("total_emissions_MtCO2e:Q", title = "Total CO2 Emissions"),
    # )

    # # create State-Owned Entity Chart
    # state_owned = alt.Chart(df.filter(pl.col("parent_type") == "State-owne
    # state_owned = state_owned.mark_line(color = "blue").encode(
    #     alt.X("year:O", title = "Year"),
    #     alt.Y("total_emissions_MtCO2e:Q", title = "Total CO2 Emissions")
    # )

    # # create Investor-owned Company Chart
    # investor_owned = alt.Chart(df.filter(pl.col("parent_type") == "Investo
    # investor_owned = investor_owned.mark_line(color = "black").encode(
    #     alt.X("year:O", title = "Year"),
    #     alt.Y("total_emissions_MtCO2e:Q", title = "Total CO2 Emissions")
    # )

    # emissions_by_all_types = nation_states + state_owned + investor_owned

    return all_emissions_by_type

emissions_by_entity_type(emissions)
```

Out[ ]:

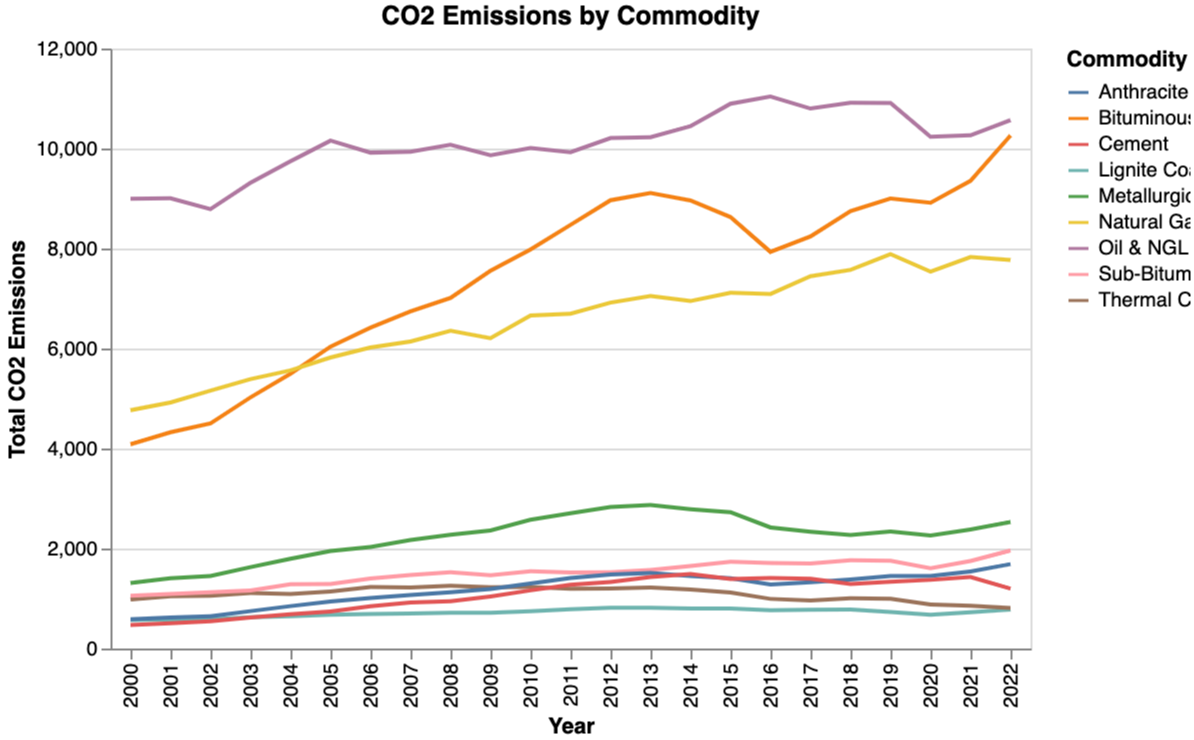


Emissions by Commodity

In [ ]:

```
def emissions_by_commodity(df):

    # aggregate Emissions by commodity
    df1 = df.group_by(["year", "commodity"]).agg(pl.col("total_emissions_MtC

    # set colors by commodity – NOT WORKING YET
    # color_scale = alt.Scale(domain=
    #                          ['Oil & NGL',
    #                           'Natural Gas',
    #                           'Anthracite Coal',
    #                           'Bituminous Coal',
    #                           'Lignite Coal',
    #                           'Metallurgical Coal',
    #                           'Sub– Bituminous Coal',
    #                           'Thermal Coal',
    #                           'Cement'],
    #                          range=['red', 'orange', 'yellow', 'blue', 'gre

    # develop chart
    chart = alt.Chart(df1, title = "CO2 Emissions by Commodity").mark_line()
        alt.X("year:O").title("Year"),
        alt.Y("total_emissions_MtCO2e:Q").title("Total CO2 Emissions"),
        alt.Color("commodity:N", title = "Commodity"),

    )

    return chart

emissions_by_commodity(emissions)
```

Out[ ]:

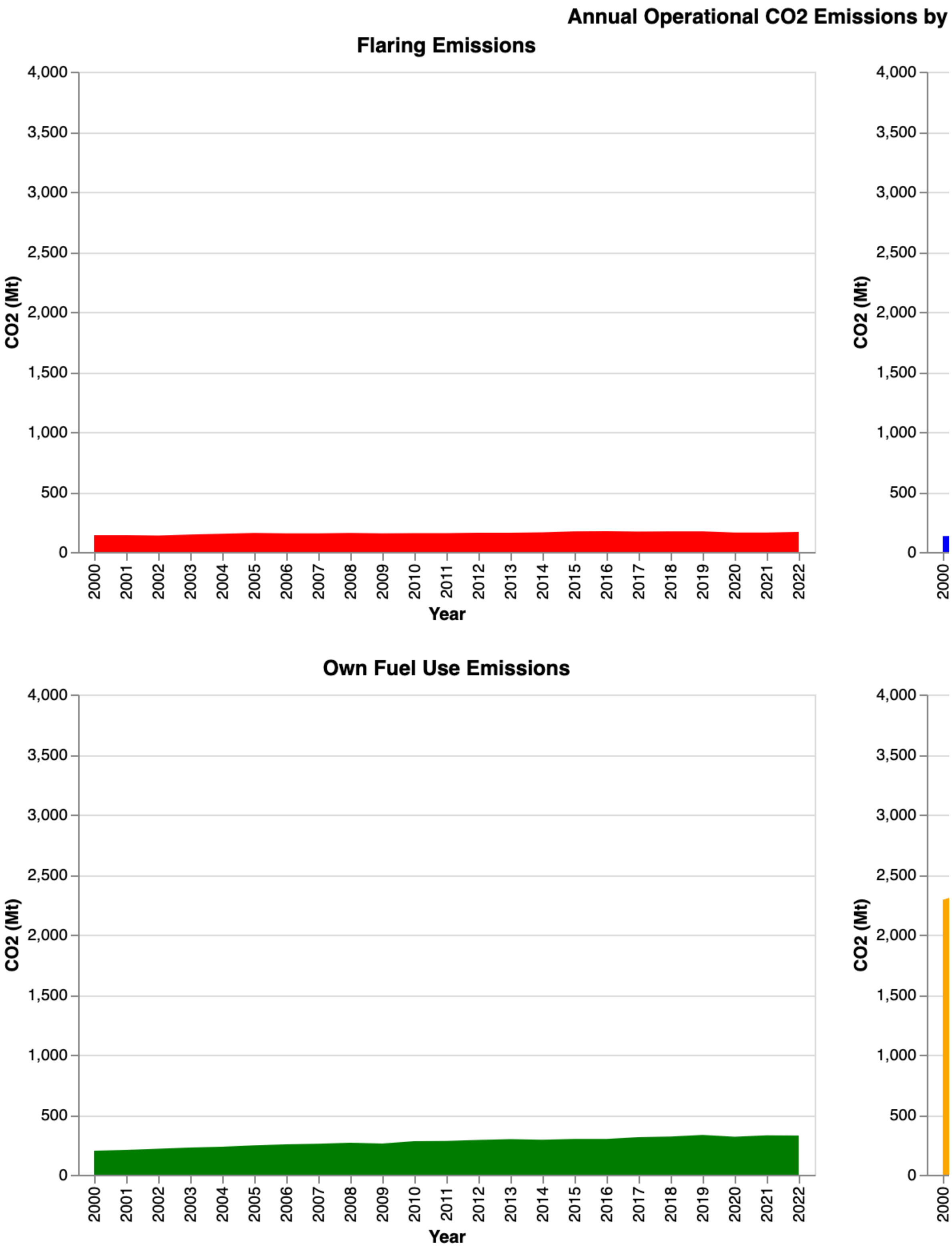


Operational Emissions by Type

In [ ]:

```
def faceted_operational_emissions(df):

    # calculate annual operational emissions by type
    df = df.group_by("year").agg(pl.col(
        ["flaring_emissions_MtCO2",
         "venting_emissions_MtCO2",
         "own_fuel_use_emissions_MtCO2",
         "fugitive_methane_emissions_MtCO2e"]).sum())

    # develop chart for each operational emission type
    flaring = alt.Chart(df, title = "Flaring Emissions").mark_area(color = "
        alt.X("year:N", title = "Year"),
        alt.Y("flaring_emissions_MtCO2:Q", title = "CO2 (Mt)", scale=alt.Sca
    )

    venting = alt.Chart(df, title = "Venting Emissions").mark_area(color = "
        alt.X("year:N", title = "Year"),
        alt.Y("venting_emissions_MtCO2:Q", title = "CO2 (Mt)", scale=alt.Sca
    )

    own_fuel_use = alt.Chart(df, title = "Own Fuel Use Emissions").mark_area
        alt.X("year:N", title = "Year"),
        alt.Y("own_fuel_use_emissions_MtCO2:Q", title = "CO2 (Mt)", scale=al
    )

    fugitive_methane = alt.Chart(df, title = "Fugitive Methane Emissions").m
        alt.X("year:N", title = "Year"),
        alt.Y("fugitive_methane_emissions_MtCO2e:Q", title = "CO2 (Mt)", sca
    )
    # concatenate charts into a grid
    custom_title = alt.TitleParams('Annual Operational CO2 Emissions by Emis
    upper = flaring | venting
    lower = own_fuel_use | fugitive_methane
    chart = alt.vconcat(upper, lower).properties(title = custom_title)

    """
    Below code uses chart repeat, but I found color and positioning customiz
    """

    # color_scale = alt.Scale(domain= emission_types,
    #                          range=['red', 'blue', 'orange', 'green'])


    # emission_types = ["flaring_emissions_MtCO2",
                      # "venting_emissions_MtCO2",
                      # "own_fuel_use_emissions_MtCO2",
                      # "fugitive_methane_emissions_MtCO2e"]

    # chart = alt.Chart(df).mark_area().encode(
    #     alt.X("year:N", title = "Year"),
    #     alt.Y(alt.repeat("row"), type='quantitative',  scale=alt.Scale(dom
    #     #alt.Color(["flaring_emissions_MtCO2:N",
    #                # "venting_emissions_MtCO2:N",
    #                # "own_fuel_use_emissions_MtCO2:N",
    #                # "fugitive_methane_emissions_MtCO2e:N"], scale = color
    # ).repeat(row= emission_types)

    return chart

faceted_operational_emissions(emissions)
```

Out[ ]:



In [ ]:

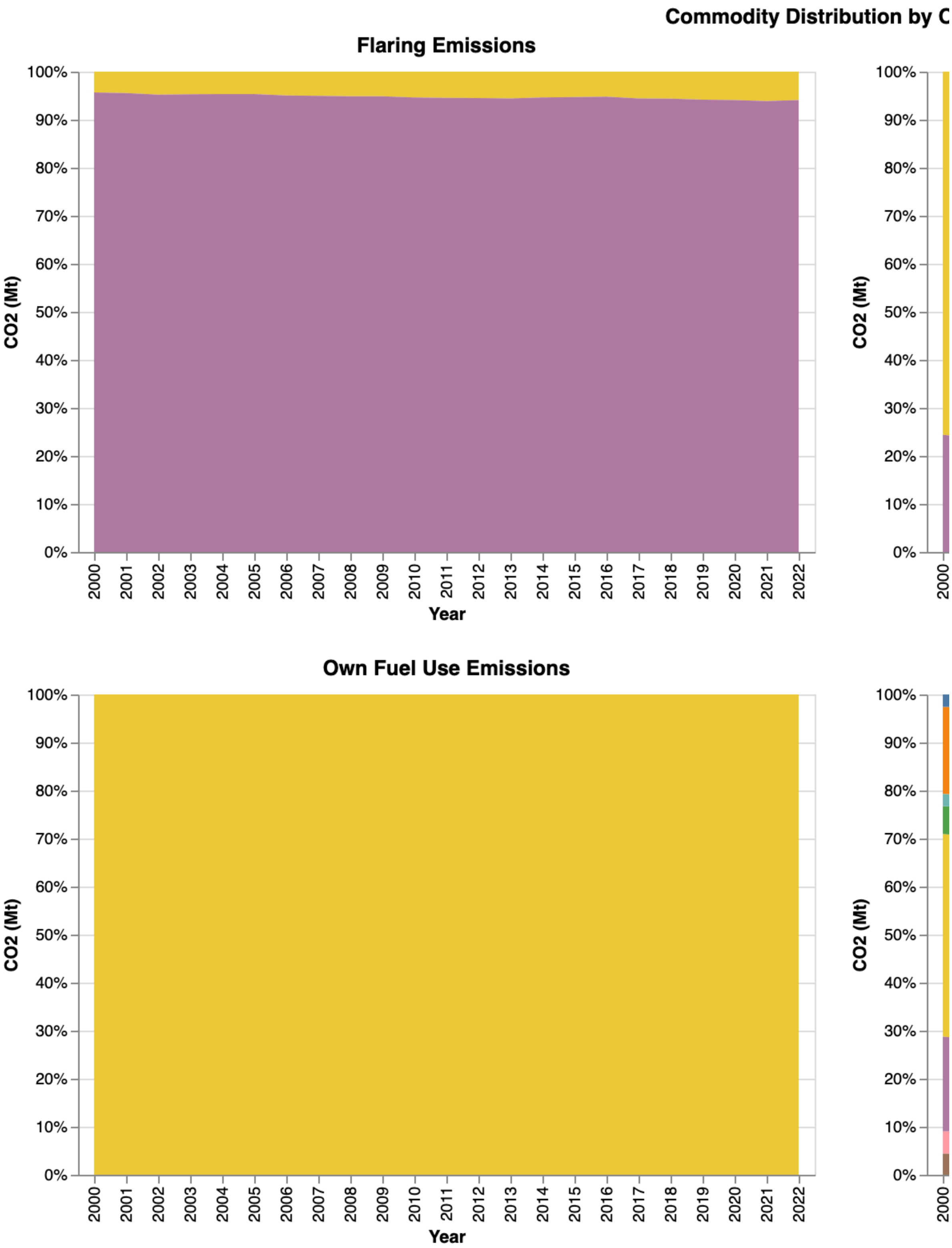
```
emissions.group_by("year").agg(pl.col(
         ["flaring_emissions_MtCO",
          "venting_emissions_MtCO2",
          "own_fuel_use_emissions_MtCO2",
          "fugitive_methane_emissions_MtC O2e"]).sum())
```

```
---------------------------------------------------------------------------
ColumnNotFoundError                       Traceback (most recent call last)
Cell In[7], line 1
----> 1 emissions.group_by("year").agg(pl.col(
      2         ["flaring_emissions_MtCO",
      3          "venting_emissions_MtCO2",
      4          "own_fuel_use_emissions_MtCO2",
      5          "fugitive_methane_emissions_MtC O2e"]).sum())

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/polars/dataframe/group_by.py:232, in GroupBy.agg(self, *aggs, **nam
ed_aggs)
    123 def agg(
    124     self,
    125     *aggs: IntoExpr | Iterable[IntoExpr],
    126     **named_aggs: IntoExpr,
    127 ) -> DataFrame:
    128     """
    129     Compute aggregations for each group of a group by operation.
    130
   (...)
    226     └──────┴────────┴─────────────────┘
    227     """
    228     return (
    229         self.df.lazy()
    230         .group_by(*self.by, **self.named_by, maintain_order=self.mai
ntain_order)
    231         .agg(*aggs, **named_aggs)
--> 232         .collect(no_optimization=True)
    233     )

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/polars/lazyframe/frame.py:2050, in LazyFrame.collect(self, type_coe
rcion, predicate_pushdown, projection_pushdown, simplify_expression, slice_p
ushdown, comm_subplan_elim, comm_subexpr_elim, cluster_with_columns, collaps
e_joins, no_optimization, streaming, engine, background, _eager, **_kwargs)
   2048 # Only for testing purposes
   2049 callback = _kwargs.get("post_opt_callback", callback)
-> 2050 return wrap_df(ldf.collect(callback))

ColumnNotFoundError: flaring_emissions_MtCO

Resolved plan until failure:

        ---> FAILED HERE RESOLVING 'group_by' <---
DF ["year", "parent_entity", "parent_type", "reporting_entity"]; PROJECT */1
7 COLUMNS; SELECTION: None
```

In [ ]:

```
def op_emissions_by_commodity(df):

    # aggregate Emissions by commodity
    df = df.group_by(["year", "commodity"]).agg(pl.col(["flaring_emissions_M
         "venting_emissions_MtCO2",
         "own_fuel_use_emissions_MtCO2",
         "fugitive_methane_emissions_MtCO2e"]).sum())
    # set colors by commodity - NOT WORKING YET
    # color_scale = alt.Scale(domain=
    #                         ['Oil & NGL',
    #                          'Natural Gas',
    #                          'Anthracite Coal',
    #                          'Bituminous Coal',
    #                          'Lignite Coal',
    #                          'Metallurgical Coal',
    #                          'Sub- Bituminous Coal',
    #                          'Thermal Coal',
    #                          'Cement'],
    #                         range=['red', 'orange', 'yellow', 'blue', 'gre

    # develop chart

    flaring = alt.Chart(df, title = "Flaring Emissions").mark_area().encode(
        alt.X("year:O").title("Year"),
        alt.Y("flaring_emissions_MtCO2:Q").title("CO2 (Mt)").stack("normaliz
        alt.Color("commodity", title = "Commodity"),
    )

    venting = alt.Chart(df, title = "Venting CO2 Emissions").mark_area().enc
        alt.X("year:O").title("Year"),
        alt.Y("venting_emissions_MtCO2:Q").title("CO2 (Mt)").stack("normaliz
        alt.Color("commodity", title = "Commodity"),
    )


    own_fuel_use = alt.Chart(df, title = "Own Fuel Use Emissions").mark_area
        alt.X("year:O").title("Year"),
        alt.Y("own_fuel_use_emissions_MtCO2:Q").title("CO2 (Mt)").stack("nor
        alt.Color("commodity", title = "Commodity"),
    )

    fugitive_methane = alt.Chart(df, title = "Fugitive Methane Emissions").m
        alt.X("year:O").title("Year"),
        alt.Y("fugitive_methane_emissions_MtCO2e:Q").title("CO2 (Mt)").stack
        alt.Color("commodity", title = "Commodity")
    )

    # concatenate charts into a grid
    custom_title = alt.TitleParams('Commodity Distribution by Operational Em
    upper = flaring | venting
    lower = own_fuel_use | fugitive_methane
    chart = alt.vconcat(upper, lower).properties(title = custom_title)

    return chart

fugitive_emissions_by_commodity(emissions)
```

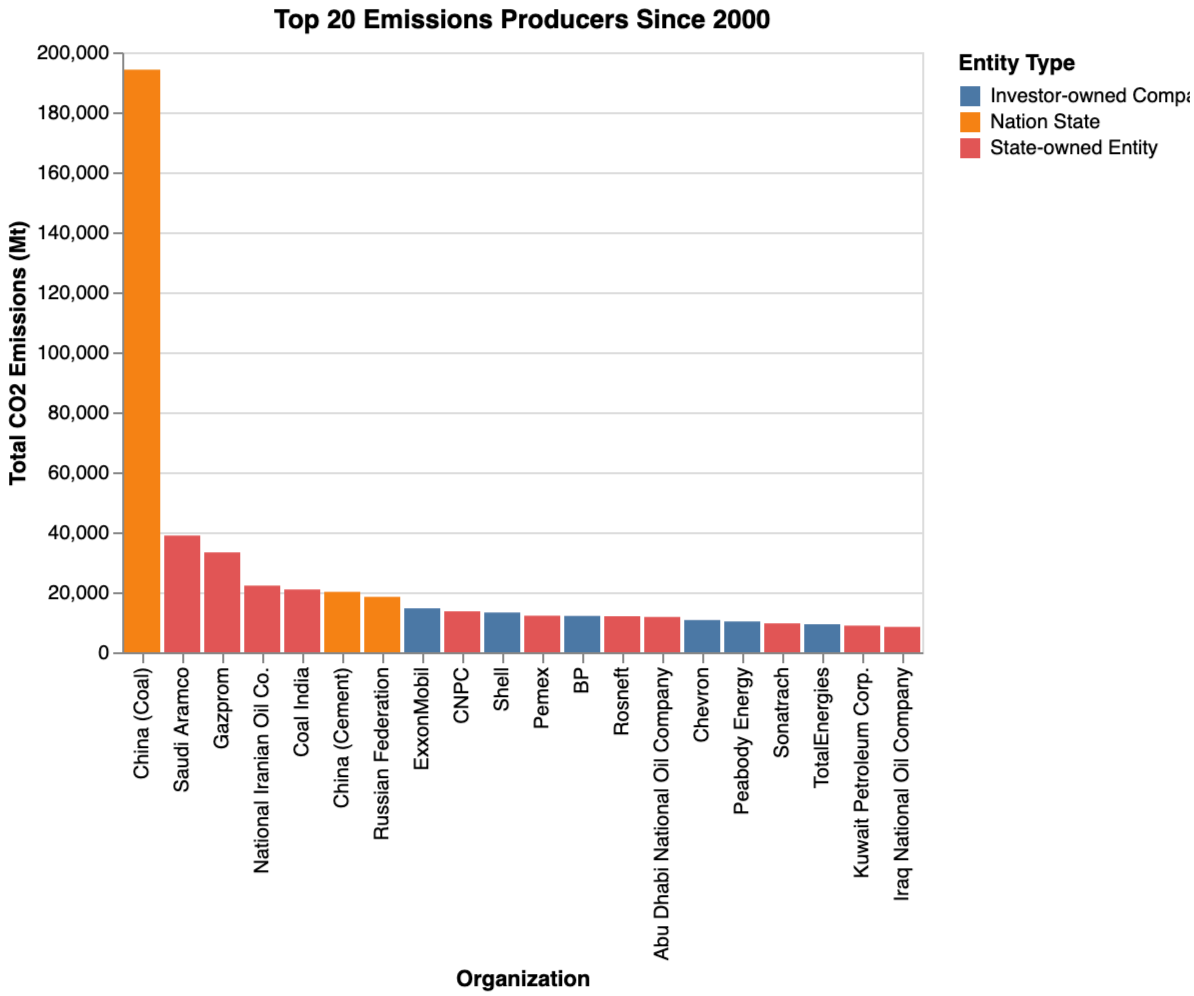
Out[ ]:



In [ ]:

```
def top_emissions_producers(df):

    # find top 20 emission producers of past 20 years
    df = df.group_by(["parent_entity","parent_type"]
                    ).agg(pl.col("total_emissions_MtCO2e").sum()
                         ).sort("total_emissions_MtCO2e", descending = True
                               ).top_k(20, by = "total_emissions_MtCO2e")
    chart = alt.Chart(df, title = "Top 20 Emissions Producers Since 2000").m
        alt.X("parent_entity:N").sort("-y").title("Organization"),
        alt.Y("total_emissions_MtCO2e:Q").title("Total CO2 Emissions (Mt)"),
        alt.Color("parent_type:N", title = "Entity Type"),
    )

    return chart

top_emissions_producers(emissions)
```

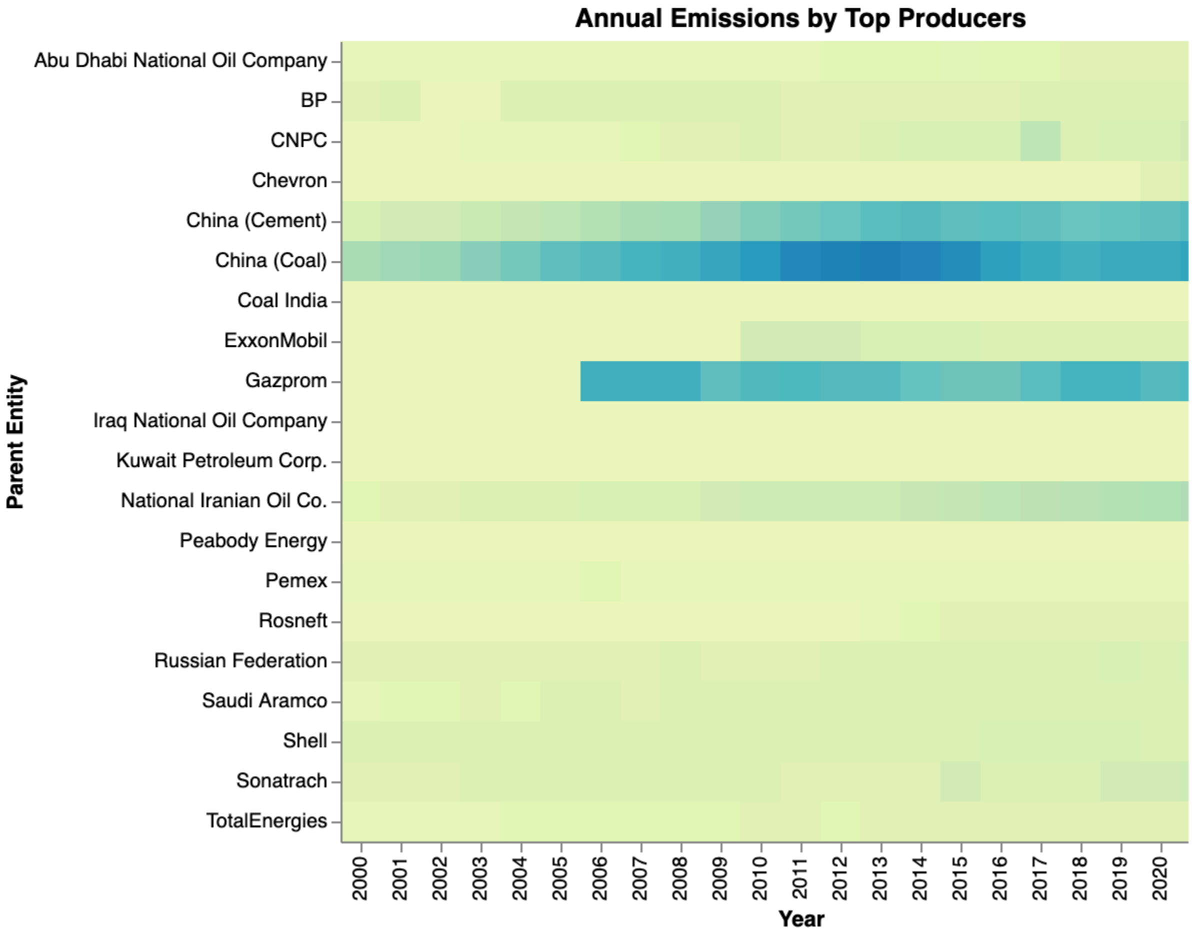
Out[ ]:



In [ ]:

```
def top_producers_by_year(df):

        # identify names of top producers
        top_producers = df.group_by(["parent_entity"]
                    ).agg(pl.col("total_emissions_MtCO2e").sum()
                         ).sort("total_emissions_MtCO2e", descending = True
                               ).top_k(20, by = "total_emissions_MtCO2e")

        top_producer_names = top_producers.select("parent_entity").to_series

        # find annual emissions of top producers

        top_producer_annual_emissions = df.filter(pl.col("parent_entity").is

        # map annual emissions of each top producer
        chart = alt.Chart(top_producer_annual_emissions, title = "Annual Emi
            alt.X("year:N", title = "Year"),
            alt.Y("parent_entity:N", title = "Parent Entity"),
            alt.Color("total_emissions_MtCO2e:Q", title = "Total CO2 Emissio
        )

        return chart


top_producers_by_year(emissions)
```

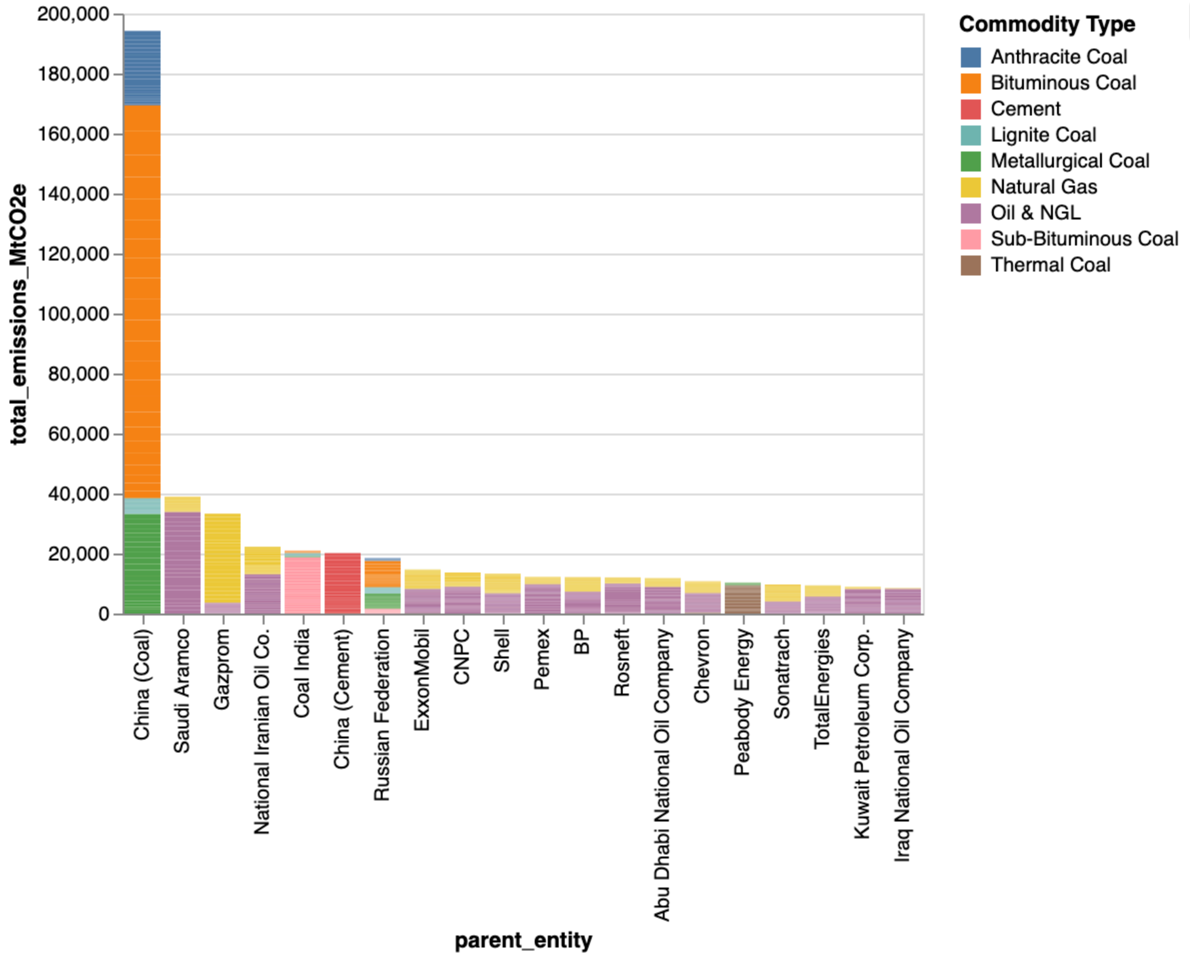
Out[ ]:



In [ ]:

```
def top_producers_commodity(df):
        # identify names of top producers
        top_producers = df.group_by(["parent_entity"]
                    ).agg(pl.col("total_emissions_MtCO2e").sum()
                         ).sort("total_emissions_MtCO2e", descending = True
                               ).top_k(20, by = "total_emissions_MtCO2e")

        top_producer_names = top_producers.select("parent_entity").to_series

        # find annual emissions of top producers

        top_producer_annual_emissions = df.filter(pl.col("parent_entity").is

        # map annual emissions of each top producer
        chart = alt.Chart(top_producer_annual_emissions).mark_bar().encode(
            alt.X("parent_entity:N").sort("-y"),
            alt.Y("total_emissions_MtCO2e:Q"),
            alt.Color("commodity:N", title = "Commodity Type"))

        return chart

top_producers_commodity(emissions)
```

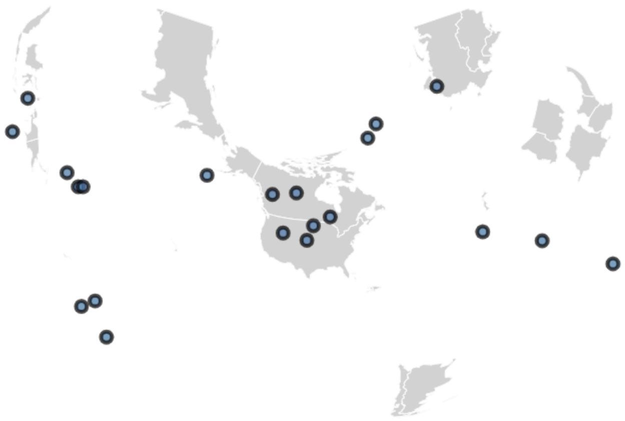
Out[ ]:



In [ ]:

```
import geopandas as gpd
def top_producers_location(df):
    url = "https://naciscdn.org/naturalearth/110m/cultural/ne_110m_admin_0_c
    gdf_ne = gpd.read_file(url)  # zipped shapefile
    gdf_ne = gdf_ne[["NAME", "CONTINENT", "POP_EST", 'geometry']][:21]

    basemap = alt.Chart(gdf_ne).mark_geoshape(
        fill='lightgrey', stroke='white', strokeWidth=0.5
    ).project(
    type='albers'
    )

    bubbles = alt.Chart(gdf_ne).transform_calculate(
    centroid=alt.expr.geoCentroid(None, alt.datum)).mark_circle(
    stroke='black').encode(
    longitude='centroid[0]:Q',
    latitude='centroid[1]:Q',
    # size="POP_EST:Q"
    )

    chart = (basemap + bubbles).project(type='identity', reflectY=True)

    return chart

top_producers_location(emissions)
```

Out[ ]:



In [ ]: