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')
```

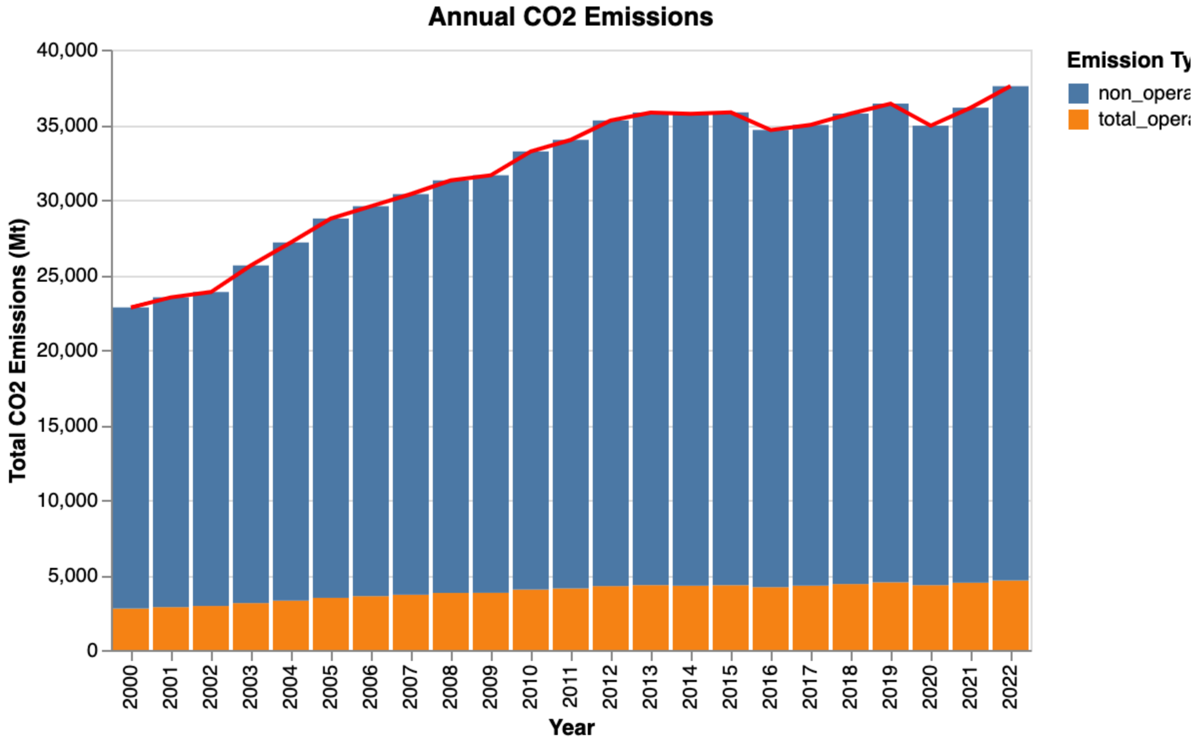
In [ ]:
```
#read in emissions data
emissions = pl.read_csv('data/emissions_high_granularity.csv', skip_rows = 1


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

1. Annual CO2 Emissions

The first data visualization is intended to provide a high level overview of annual emissions for the 122 organizations included in the report. I include the differentiation of the operational and non-operational emissions to contextualize the type of emissions. This visualization will likely need a description of the differences between the two.

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[ ]:

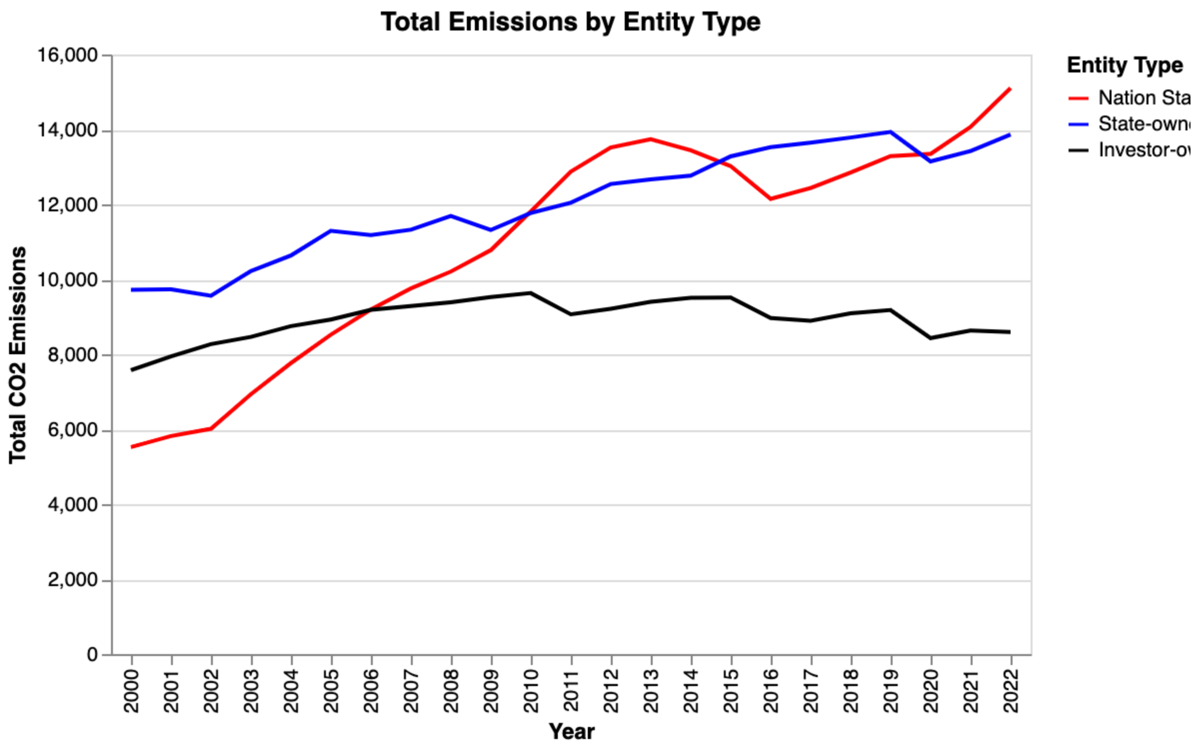


2. Annual Emissions by Entity Type

Here, we break down the emissions by entity type: nation state, state owned and investor owned. I think this differentiation is useful because it helps us understand whether the emissions are primarily government led or state led.

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

    # 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")
    )

    return total_emissions_by_types

emissions_by_entity_type(emissions)
```

Out[ ]:



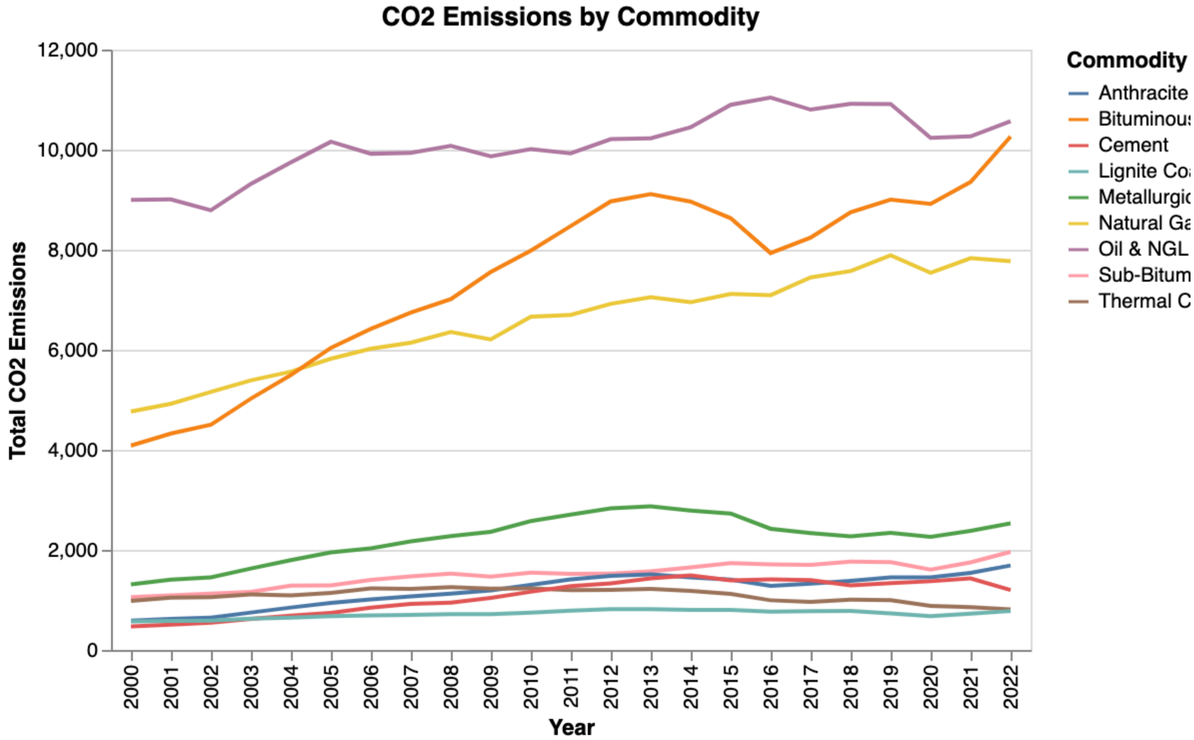
3. Emissions by Commodity

The below visualization breaks down emissions by commodity produced. This visualization is helpful because it highlights the narrrative that the commodity that contributes most to the emissions (of the ones examined) is coal. While this is perhaps predictable, I also think seeing the breakdown of various types of coal is informative and educational.

In [ ]:

```
def emissions_by_commodity(df):

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

    # 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[ ]:



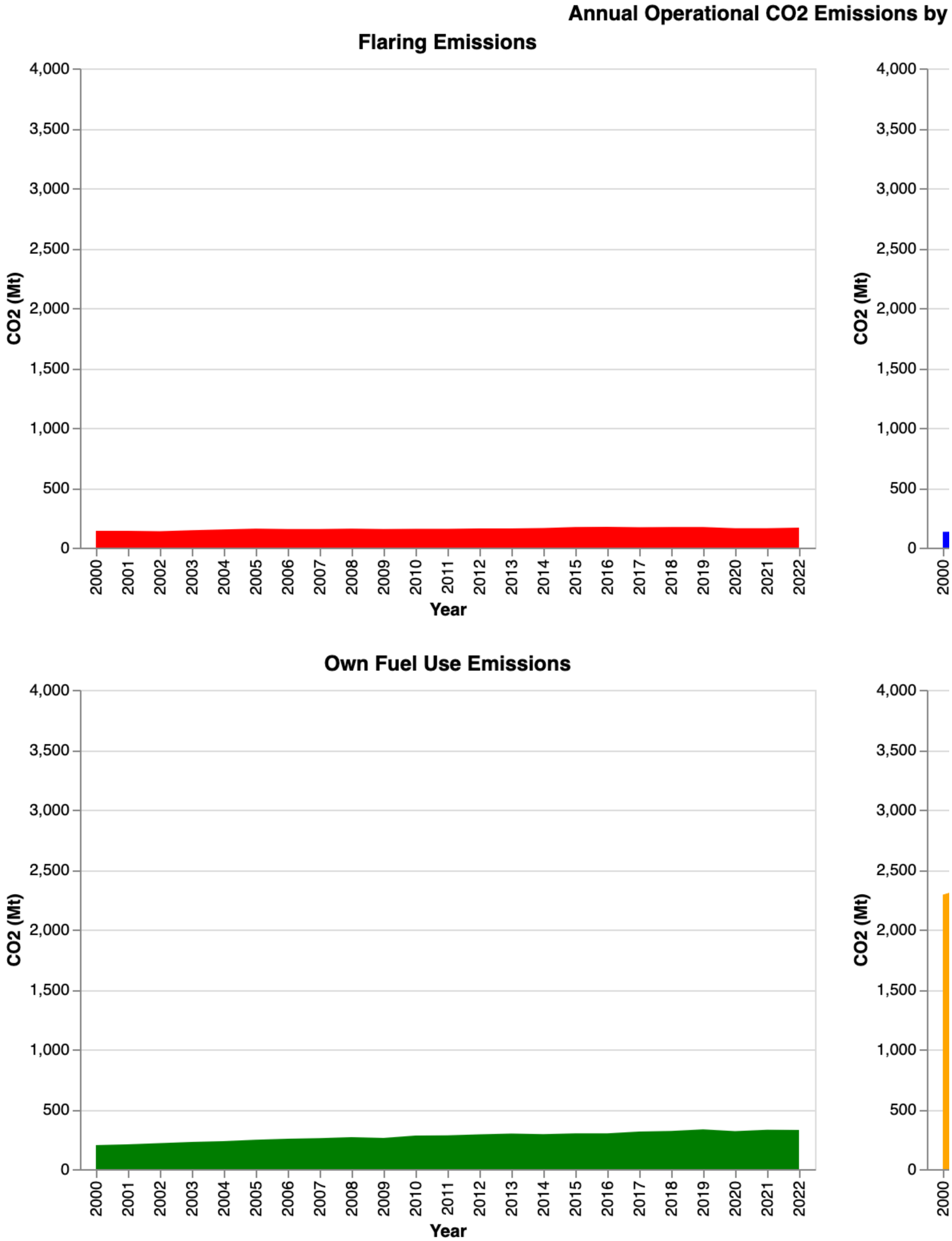
4. Operational Emissions by Type

It's important to note here that operational emissions are the minority (while non-operational emissions are the majority). However, we have breakout data on operational emissions and this visualization explores that. Notably, it demonstrates how abundant fugitive methane emissions are relative to the other types of operational emissions.

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)

    return chart

faceted_operational_emissions(emissions)
```

Out[ ]:



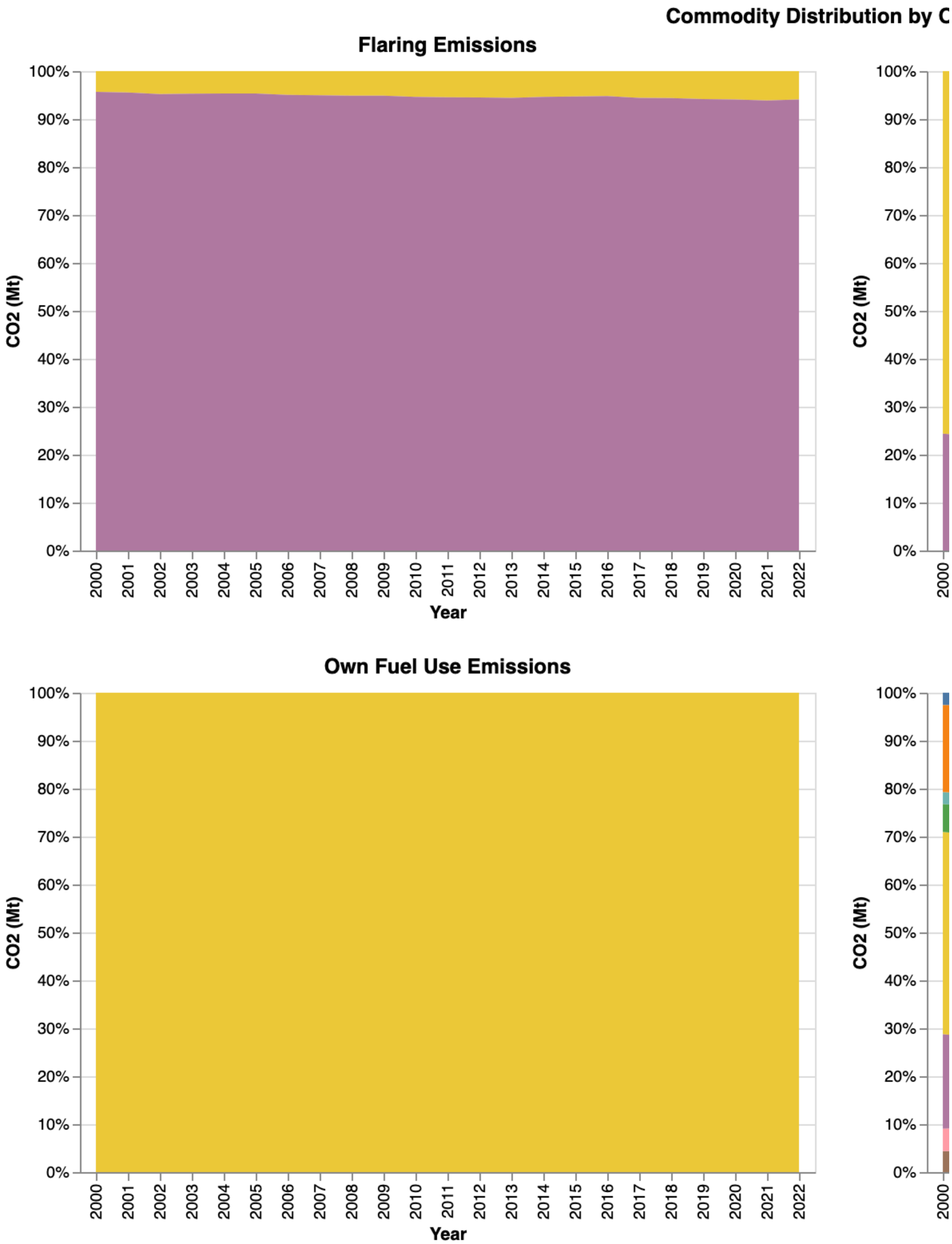
5. Operational Emissions by Commodity

Following the findings of the previous chart, this visualization demonstrates how diverse the fugitive methane emissions are in their sources relative to the other operational emission types. This diversity somewhat explains why fugitive methane emissions are so much higher than the rest; they are produced by nearly every commodity examined.

In [ ]:
```
def fugitive_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[ ]:

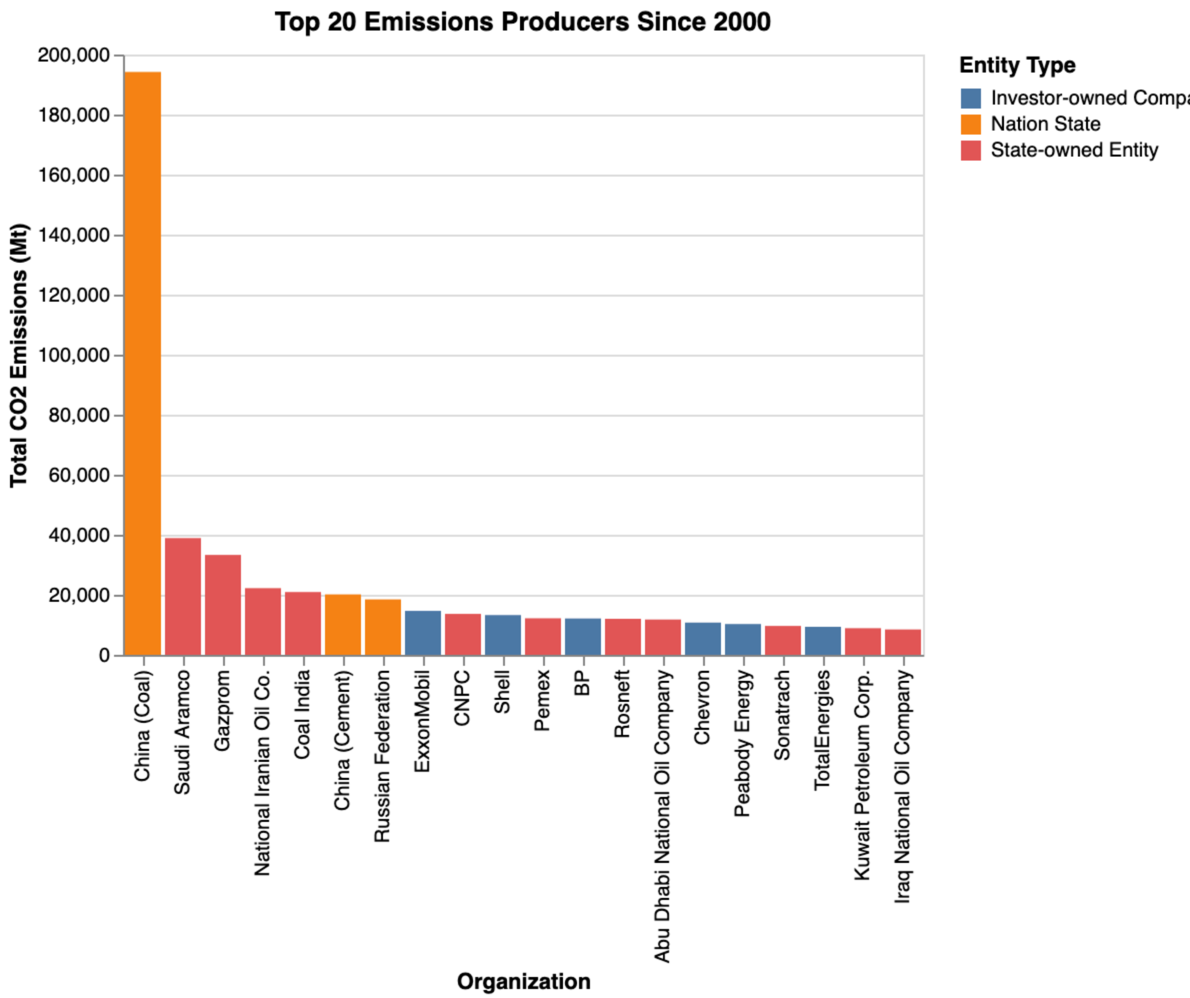


6. Top 20 Emissions Producers

Now that we have a somewhat stronger understanding of what the emissions of the past 20 years look like, we can take a closer look at who is proposing them. It might be a stronger narrative to put this above the operational emissions breakdown, but I have not committed to that choice yet. This visualization shows the astounding amount of emissions China produces through coal production relative to other top producers. It's also interesting to note that only 6 of the 20 top producers are investor-owned.

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[ ]:



7. Annual Emissions of Top 20 Producers

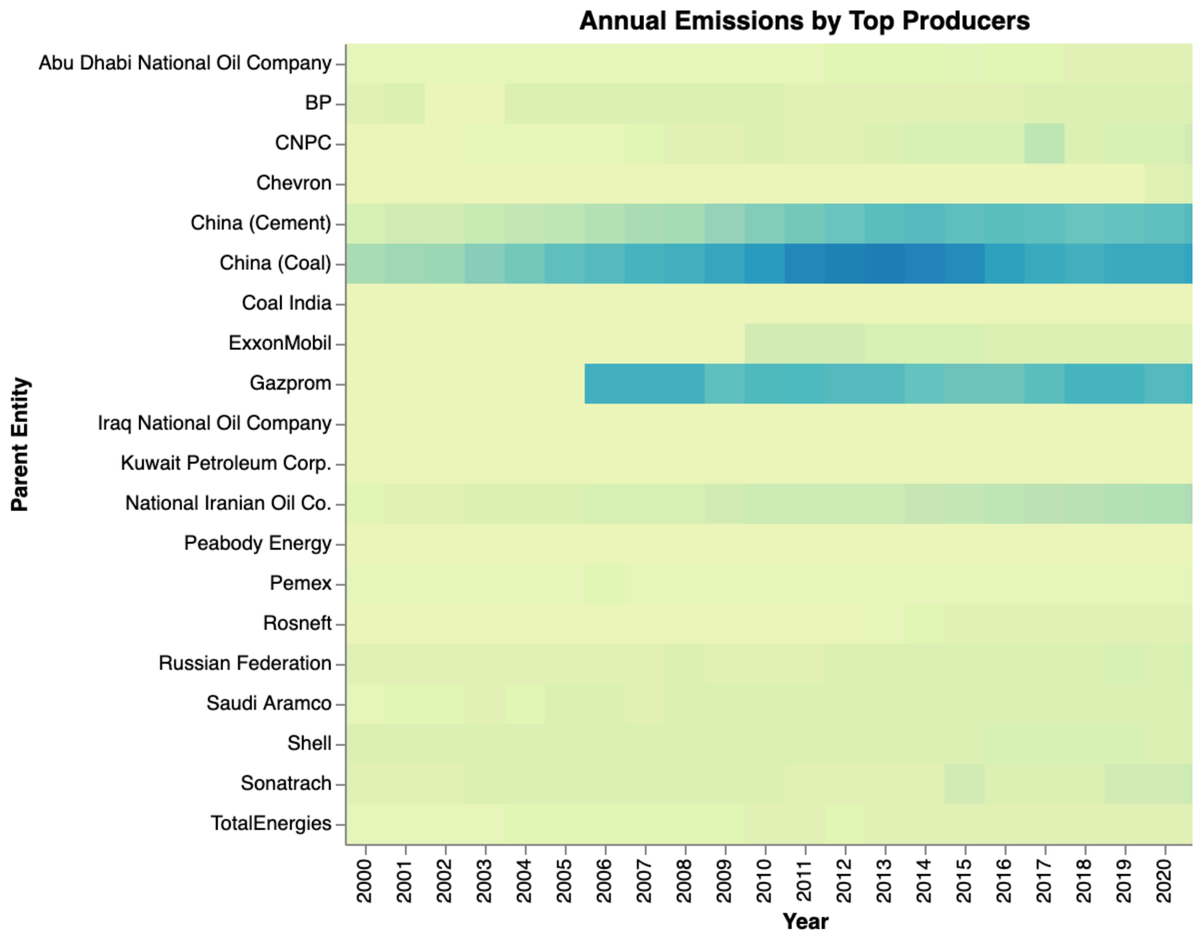
The below graph explores any annual trends in emissions by each of the top 20 producers. Unfortunately I do not know if this visualization offers much additional information other than the fact that China coal production related emissions intensified between 2010-2015 and that Gazprom was perhaps founded in 2005 but admittedly came to the scene with full force. I will have to think further about if this makes it into the final infographic.

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 Emissic
        )

        return chart


top_producers_by_year(emissions)
```

Out[ ]:



8. Commodity Distribution of Top 20 Producers

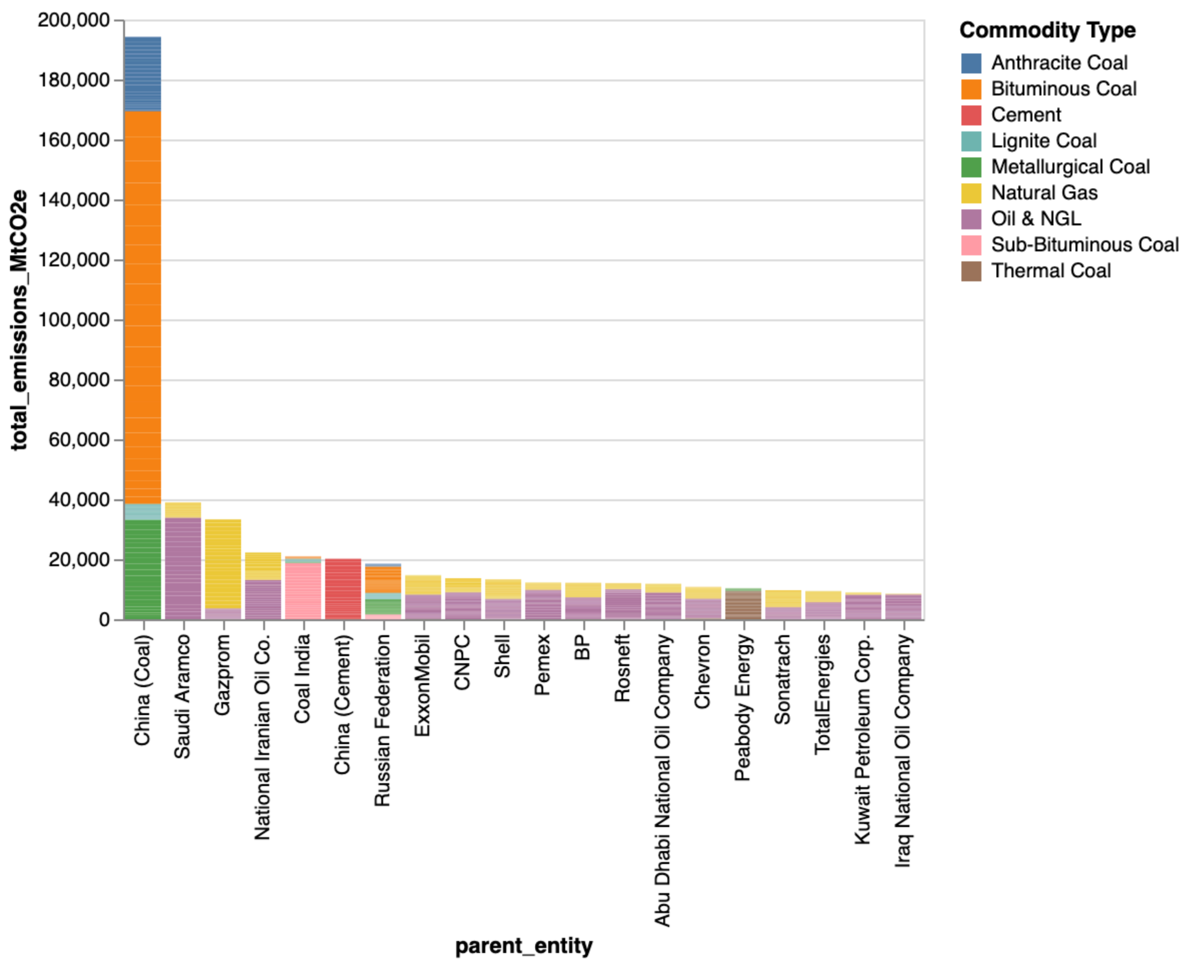
The below visualization examines the breakout of commodities by top producers. While China takes the lead with coal emissions, it's clear that the majority of entities that are generating outsized levels of CO2 emissions are producing oil and NGL.

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[ ]:



9. Geographic Origins of Top 20 Producers

I think this could be a valuable graphic to add, but I don't have a full grasp of how to do it yet and this is only generated through dummy data. This image would perhaps go first before identifying exactly who the entities are.

So far, I've only had the idea of manually looking up where each of the top producers are and manually inputting country codes/coordinates...

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[ ]: