```
#Import libraries
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import requests
import time
from IPython.display import display, HTML
```

```
# 1. Fetch all networks
def fetch_networks():
    url = "http://api.citybik.es/v2/networks"
    response = requests.get(url)
    return response.json()["networks"]

# 2. Fetch station details for a given network
def fetch_network_details(network_id):
    url = f"http://api.citybik.es/v2/networks/{network_id}"
    response = requests.get(url)
    return response.json().get("network", {})
```

```
#Fetch network list from City Bike API
API_BASE = "http://api.citybik.es/v2/networks"

def fetch_networks():
    response = requests.get(API_BASE)
    response.raise_for_status()
    return response.json().get("networks", [])

networks = fetch_networks()
networks[:2]  # Show a sample
```

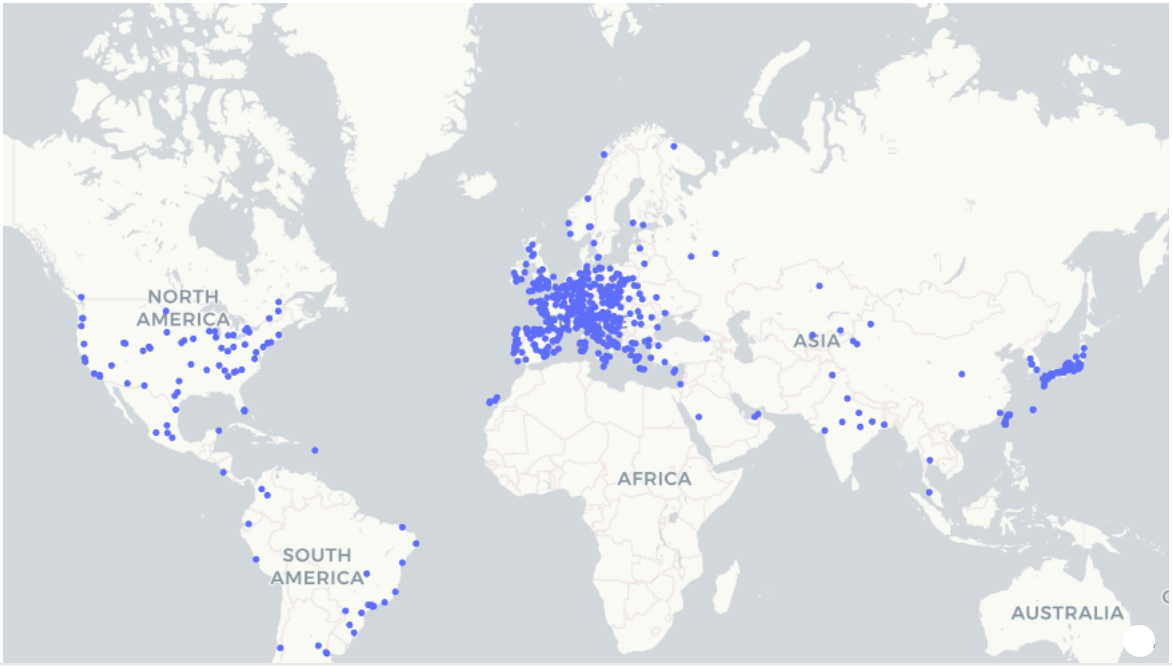
```
[{'id': 'abu-dhabi-careem-bike',
  'name': 'Abu Dhabi Careem BIKE',
  'location': {'latitude': 24.4866,
   'longitude': 54.3728,
   'city': 'Abu Dhabi',
   'country': 'AE'},
  'href': '/v2/networks/abu-dhabi-careem-bike',
  'company': ['Careem'],
  'gbfs_href': 'https://dubai.publicbikesystem.net/customer/gbfs/v2/en/gbfs.json'},
 {'id': 'acces-velo-saguenay',
  'name': 'Accès Vélo',
  'location': {'latitude': 48.433333,
   'longitude': -71.083333,
   'city': 'Saguenay',
   'country': 'CA'},
  'href': '/v2/networks/acces-velo-saguenay',
  'company': ['PBSC Urban Solutions'],
  'gbfs_href': 'https://saguenay.publicbikesystem.net/customer/gbfs/v2/gbfs.json'}]
```

```
# Load and preprocess networks
networks = fetch_networks()
df = pd.json_normalize(networks)
df = df.rename(columns={
    "id": "network_id",
    "location.city": "city",
    "location.country": "country",
    "location.latitude": "latitude",
    "location.longitude": "longitude"
})
df = df.dropna(subset=["latitude", "longitude"])
df.head()
```

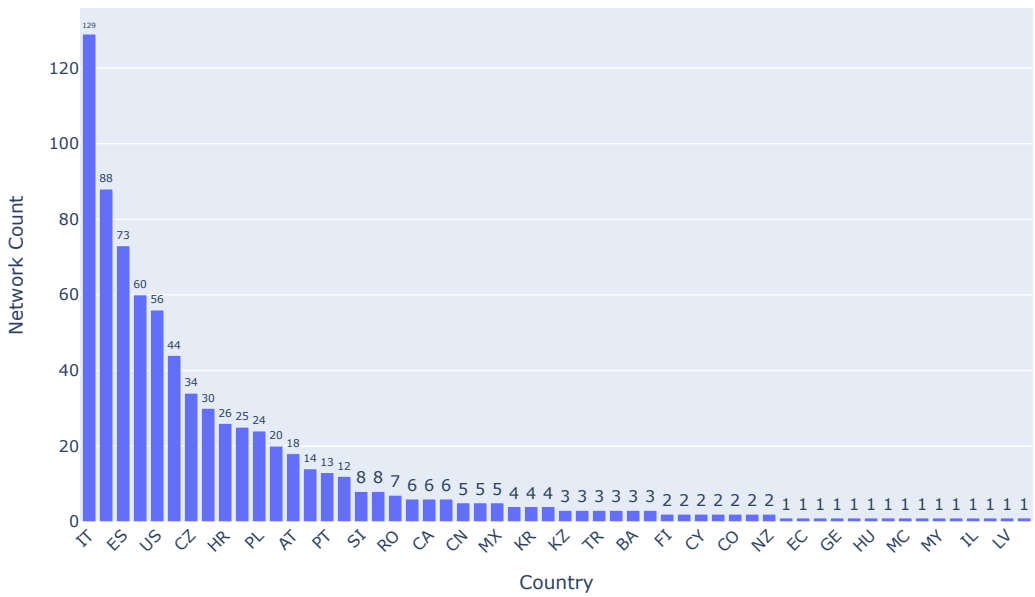
	network_id	name	href	company	gbfs_href	latitude	longitude	city	countr
0	abu-dhabi-careem-bike	Abu Dhabi Careem BIKE	/v2/networks/abu-dhabi-careem-bike	[Careem]	https://dubai.publicbikesystem.net/customer/gb...	24.486600	54.372800	Abu Dhabi	A
1	acces-velo-saguenay	Accès Vélo	/v2/networks/acces-velo-saguenay	[PBSC Urban Solutions]	https://saguenay.publicbikesystem.net/customer...	48.433333	-71.083333	Saguenay	C
2	aksu	Aksu	/v2/networks/aksu	[ ]	NaN	41.166400	80.261700	(Aksu City)	C
3	alba	Alba	/v2/networks/alba	[Comunicare S.r.l.]	NaN	44.716667	8.083333	Alba	
4	albabici	AlbaBici	/v2/networks/albabici	[Instituto Tecnológico de Castilla y León (ITCL)]	NaN	38.994300	-1.860200	Albacete	E

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Plot World Station Map
def plot_world_station_map(df, filters_applied=False):
    if df.empty or not {"latitude", "longitude"}.issubset(df.columns):
        return None

    df_map = df.dropna(subset=["latitude", "longitude"]).copy()
    df_map["name"] = df_map.get("name", "Unknown")

    hover_data = {
        "latitude": True,
        "longitude": True
    }

    if filters_applied:
        df_map["free_bikes"] = pd.to_numeric(df_map.get("free_bikes", 0), errors="coerce").fillna(0).astype(int)
        df_map["empty_slots"] = pd.to_numeric(df_map.get("empty_slots", 0), errors="coerce").fillna(0).astype(int)
        hover_data["free_bikes"] = True
        hover_data["empty_slots"] = True

    fig = px.scatter_mapbox(
        df_map,
        lat="latitude",
        lon="longitude",
        hover_name="name",
        hover_data=hover_data,
        zoom=1,
        height=500
    )
    fig.update_layout(mapbox_style="carto-positron", margin=dict(r=0, t=0, l=0, b=0))
    return fig

fig = plot_world_station_map(df)
fig.show()
```

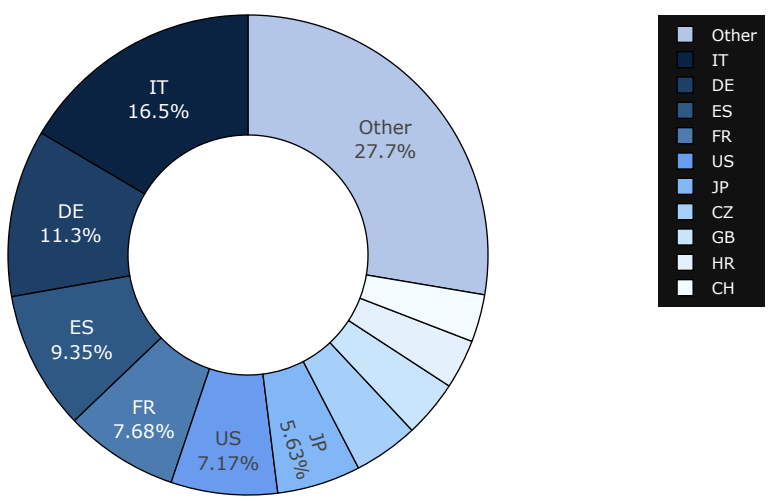



```
#Bar chart: Number of Networks per Country
country_counts = df["country"].value_counts().reset_index()
country_counts.columns = ["Country", "Network Count"]

fig_bar = px.bar(country_counts, x="Country", y="Network Count", text="Network Count")
fig_bar.update_traces(textposition='outside')
fig_bar.update_layout(xaxis_tickangle=-45)
fig_bar.show()
```



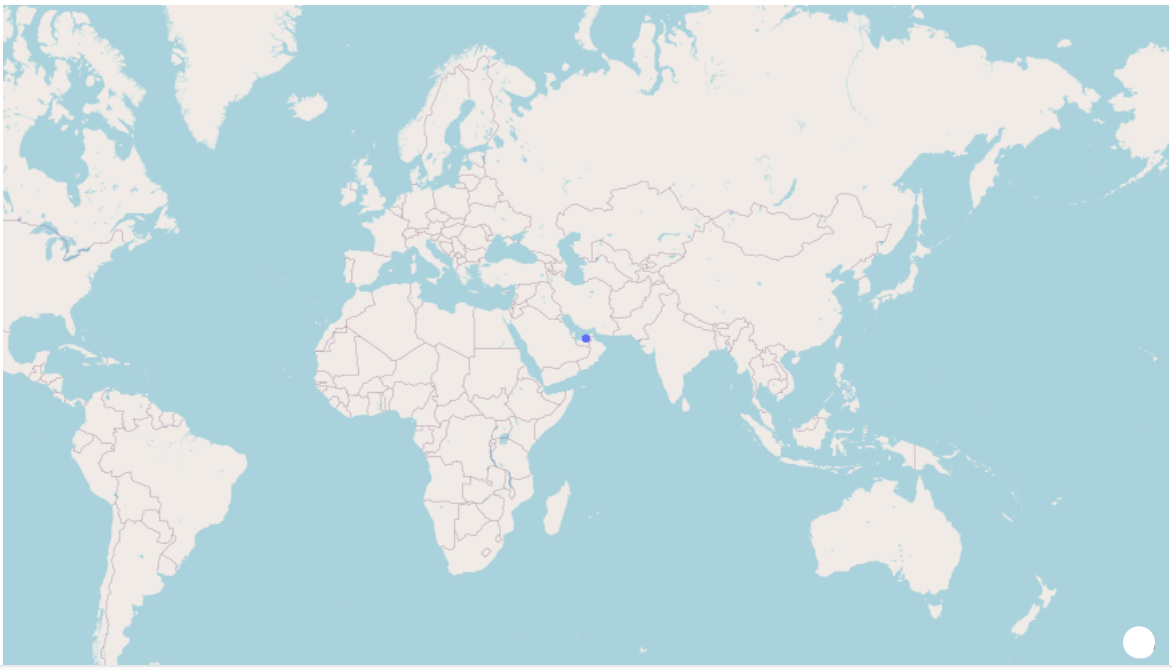

```
#Pie Chart of Network Distribution by Country
def plot_network_distribution_styled(df, top_n=10):
    # Count networks per country
    counts = df["country"].value_counts().reset_index()
    counts.columns = ["country", "network_count"]

    # Top N + group others
    top = counts.head(top_n)
    others_sum = counts["network_count"][top_n:].sum()

    final_df = top.copy()
```

```
    if others_sum > 0:
        final_df = pd.concat([
            top,
            pd.DataFrame([{"country": "Other", "network_count": others_sum}])
        ])

    # Cool blue shades
    cool_blues = [
        "#0A2342", "#1E3F66", "#2E5984", "#4B7BAF", "#699BEF",
        "#82B7F7", "#A6D0FA", "#C9E5FC", "#E4F1FD", "#F4FBFF", "#B4C6E7"
    ][:len(final_df)]  # Trim or adjust to match count

    fig = px.pie(
        final_df,
        names="country",
        values="network_count",
        hole=0.5,
    )

    fig.update_traces(
        textinfo="percent+label",
        textfont_size=14,
        marker=dict(colors=cool_blues, line=dict(color='#000000', width=1))
    )

    fig.update_layout(
        title_text="🌍 Networks Distribution by Country (Top 10)",
        showlegend=True,
        paper_bgcolor="#111",
        plot_bgcolor="#111",
        font=dict(color="white"),
        height=500
    )

    return fig


fig = plot_network_distribution_styled(df, top_n=10)
fig.show()
```




```
#Station Map for Selected Network
def plot_station_map(network, selected_station_name=None):
    if not network or 'stations' not in network:
        return None

    df = pd.DataFrame(network['stations']).dropna(subset=['latitude', 'longitude'])

    if selected_station_name:
        df = df[df["name"] == selected_station_name]
```

```
        zoom = 13
    else:
        zoom = 1

    fig = px.scatter_mapbox(
        df,
        lat="latitude",
        lon="longitude",
        hover_name="name",
        hover_data={"free_bikes": True, "empty_slots": True},
        zoom=zoom,
        height=500
    )
    fig.update_layout(mapbox_style="open-street-map", margin=dict(r=0, t=0, l=0, b=0))
    return fig
# Pick a specific network (first one as example)
network_id = df.iloc[0]["network_id"]
network_data = fetch_network_details(network_id)

fig = plot_station_map(network_data)
fig.show()
```



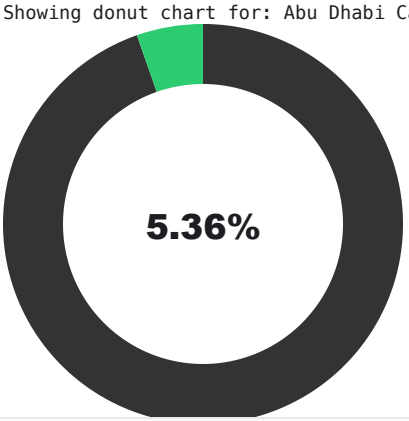
```
#Get station details for a selected network

def fetch_network_details(network_id):
    url = f"{API_BASE}/{network_id}"
    response = requests.get(url)
    response.raise_for_status()
    return response.json().get("network", {})

# Example: show details for a specific network
selected_network_id = df.iloc[0]["network_id"]  # Or use any ID
details = fetch_network_details(selected_network_id)
stations = details.get("stations", [])
pd.DataFrame(stations)[["name", "empty_slots", "free_bikes"]].head()
```

	name	empty_slots	free_bikes
0	AUH - Marasy	4	3
1	AUH - Al Zeina	14	1
2	AUH - Yas Plaza	5	0
3	AUH - ADIA HQ	8	2
4	AUH - Al Muneera North	3	0

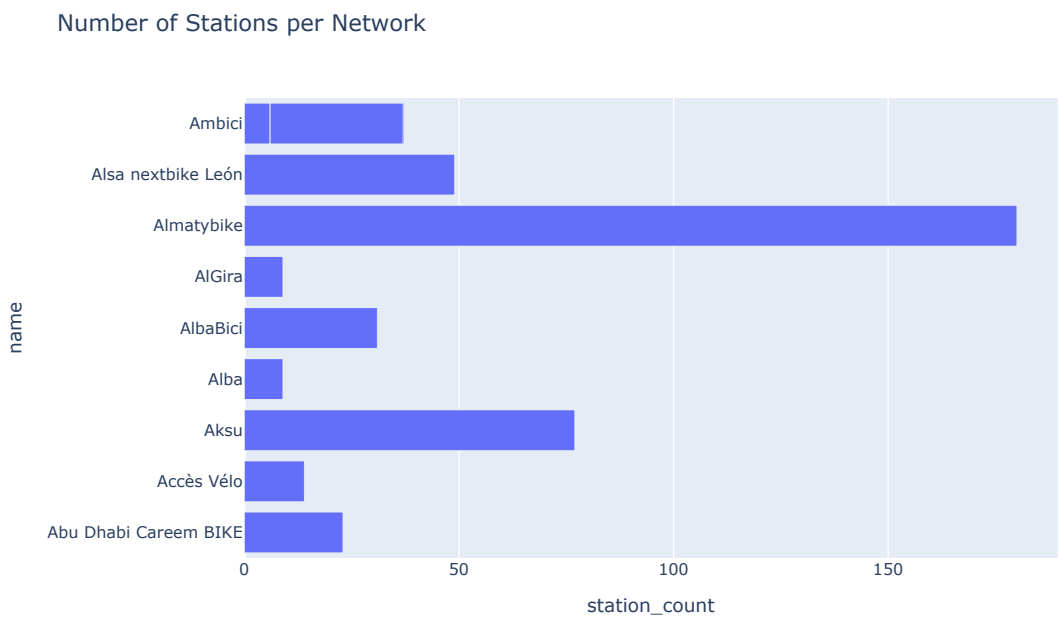
```
# --- Load basic network info ---
networks = fetch_networks()
```

```
df = pd.json_normalize(networks)
df = df.rename(columns={
    "id": "network_id",
    "location.city": "city",
    "location.country": "country",
    "location.latitude": "latitude",
    "location.longitude": "longitude"
})
df = df.dropna(subset=["latitude", "longitude"])

# --- Enrich with station_count (limited to avoid API bans) ---
import time

def enrich_with_station_counts(df, fetch_func, max_networks=10, sleep_sec=1.5):
    station_counts = []
    for i, network_id in enumerate(df["network_id"][:max_networks]):
        try:
            details = fetch_func(network_id)
            count = len(details.get("stations", []))
        except Exception as e:
            print(f"Error: {e}")
            count = 0
        station_counts.append(count)
        print(f"{i+1}/{max_networks}: {network_id} → {count} stations")
        time.sleep(sleep_sec)
    enriched = df[:max_networks].copy()
    enriched["station_count"] = station_counts
    return enriched

# First enrich, then save
enriched_df = enrich_with_station_counts(df, fetch_network_details)
enriched_df.to_csv("enriched_station_data.csv", index=False)

# --- Clean donut chart style ---
def render_clean_fixed_donut(selected_network, enriched_df, full_df):
    selected = enriched_df[enriched_df["name"] == selected_network]
    full_total = full_df["station_count"].sum()
    selected_count = selected["station_count"].sum()
    percent = round((selected_count / full_total) * 100, 2) if full_total > 0 else 0

    fig = go.Figure(data=[go.Pie(
        labels=["Selected Network", "Others"],
        values=[selected_count, full_total - selected_count],
        hole=0.7,
        marker_colors=["#2ecc71", "#333333"],
        textinfo="none",
        hoverinfo="label+value"
    )])

    fig.add_annotation(
        text=f"<b>{percent:.2f}%</b>",
        font=dict(size=26, color="#1c1e21", family="Arial Black"),
        showarrow=False,
        x=0.5, y=0.5
    )

    fig.update_layout(
        showlegend=False,
        paper_bgcolor="white",
        plot_bgcolor="white",
        margin=dict(t=0, b=0, l=0, r=0),
        height=300,
        width=300
    )

    return fig

# --- Show the donut chart for first network ---
network_name = enriched_df.iloc[0]["name"]
print(f"Showing donut chart for: {network_name}")
fig = render_clean_fixed_donut(network_name, enriched_df, enriched_df)
fig.show()
```

```
1/10: abu-dhabi-careem-bike → 23 stations
2/10: acces-velo-saguenay → 14 stations
3/10: aksu → 77 stations
4/10: alba → 9 stations
5/10: albabici → 31 stations
6/10: algira → 9 stations
7/10: almatybike → 180 stations
8/10: alsa-nextbike-leon → 49 stations
9/10: ambici-amb → 6 stations
10/10: ambici-badalona → 31 stations
Showing donut chart for: Abu Dhabi Careem BIKE
```




```
# Bar Chart (Horizontal)
fig = px.bar(
    enriched_df,
    x="station_count",
    y="name",
    orientation="h",
    title="Number of Stations per Network"
)
fig.show()
```




```
# Summary Table (Grouped by Country & City)
country_summary = enriched_df.groupby("country")["station_count"].mean().reset_index().sort_values(by="station_count", ascending
print("Average stations per country:")
display(country_summary)

city_summary = enriched_df.groupby("city")["station_count"].mean().reset_index().sort_values(by="station_count", ascending=False
print("Average stations per city:")
display(city_summary)
```

```
Average stations per country:
```

	country	station_count
5	KZ	180.00
2	CN	77.00
3	ES	29.25
0	AE	23.00
1	CA	14.00
4	IT	9.00
6	PT	9.00

```
Average stations per city:
```

	city	station_count
3	Almaty	180.0
9	(Aksu City)	77.0
7	León	49.0
2	Albacete	31.0
5	Badalona	31.0
0	Abu Dhabi	23.0
8	Saguenay	14.0
1	Alba	9.0
4	Almeirim	9.0
6	Barcelona	6.0

Next steps: Generate code with country_summary | View recommended plots | New interactive sheet | Generate code with city_summary | View recom

```
# Global Insights

print("Total number of networks:", len(enriched_df))

top_country = enriched_df.groupby("country")["station_count"].sum().idxmax()
print("Country with the most stations:", top_country)

top_network = enriched_df.loc[enriched_df["station_count"].idxmax()]
print("Network with the most stations:", top_network['name'], "–", top_network['station_count'])
```

```
Total number of networks: 10
Country with the most stations: KZ
Network with the most stations: Almatybike – 180
```

```
# === Load Cached CSV ===
df_cached = pd.read_csv("cached_station_data.csv")
```

```
# Sum station counts across same network names
network_totals = df_cached.groupby("name")["station_count"].sum().reset_index()

# Find top network by total station count
top_network_name = network_totals.sort_values(by="station_count", ascending=False).iloc[0]["name"]
top_station_count = network_totals.sort_values(by="station_count", ascending=False).iloc[0]["station_count"]

print("Top Network (aggregated):", top_network_name)
print("Total Stations:", top_station_count)


network_totals = df_cached.groupby("name")["station_count"].sum().reset_index()
top_network_name = network_totals.sort_values(by="station_count", ascending=False).iloc[0]["name"]
```

```
Top Network (aggregated): YouBike
Total Stations: 8888
```

```
# === Step 1: Summary Metrics (Fixed) ===
total_networks = len(df_cached)
```

```
total_stations = df_cached["station_count"].sum()
top_country = df_cached.groupby("country")["station_count"].sum().idxmax()

# Aggregate total stations by network name
network_totals = df_cached.groupby("name")["station_count"].sum().reset_index()
top_network_name = network_totals.sort_values(by="station_count", ascending=False).iloc[0]["name"]

# === Step 2: Render HTML Summary Cards in 2x2 Grid ===
def render_card(label, value, bold=False):
    return f"""
    <div style="
        background-color:#1a1a1a;
        color:white;
        border-radius:12px;
        padding:20px;
        width:230px;
        text-align:center;
        font-family:Arial;
        font-size:18px;
        box-shadow: 0 0 10px rgba(255,255,255,0.05);
    ">
        <div style="margin-bottom:10px; font-weight:normal;">{label}</div>
        <div style="font-size:26px; font-weight:{'bold' if bold else 'normal'};">{value}</div>
    </div>
    """

# Arrange cards in 2x2 grid using flexbox
html_output = f"""
<div style="display:flex; flex-wrap:wrap; gap:20px; justify-content:flex-start;">
    {render_card('Global Network Count', total_networks)}
    {render_card('Leading Country by Total Stations', top_country, bold=True)}
    {render_card('Top Network by Station Count', top_network_name, bold=True)}
    {render_card('Total Stations Worldwide', total_stations)}
</div>
"""

display(HTML(html_output))
```

Global Network Count

781

Leading Country by Total Stations

JP

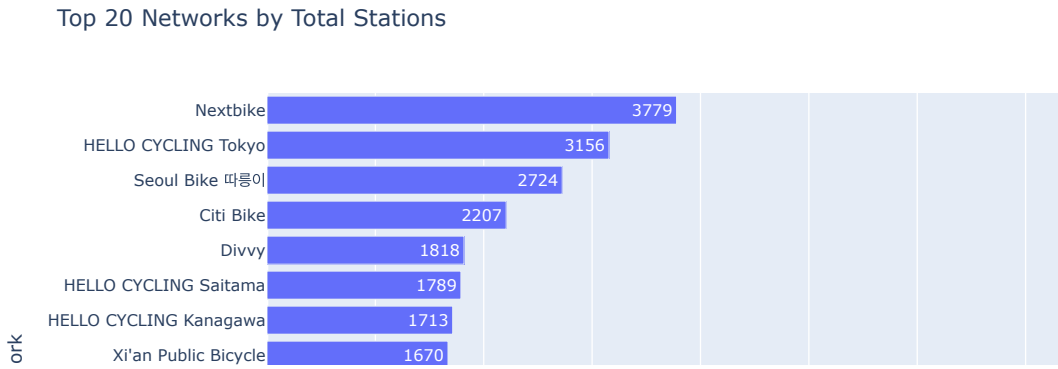
Top Network by Station Count

YouBike

Total Stations Worldwide

86862

```
import plotly.express as px

# Sort and select top 20 networks by station count
top_networks_df = df_cached.sort_values(by="station_count", ascending=False).head(20)

fig = px.bar(
    top_networks_df,
    x="station_count",
    y="name",
    orientation="h",
    text="station_count",
    title="Top 20 Networks by Total Stations",
    labels={"station_count": "Total Stations", "name": "Network"}
)

fig.update_layout(
    yaxis=dict(autorange="reversed"),
    height=600
)

fig.show()
```


Top 20 Networks by Total Stations
Nextbike
HELLO CYCLING Tokyo
Seoul Bike 따릉이
Citi Bike
Divvy
HELLO CYCLING Saitama
HELLO CYCLING Kanagawa
Xi'an Public Bicycle
3779
3156
2724
2207
1818
1789
1713
1670
ork