Name: Nidhi Vijaynand Lavand

B.Tech AI, 4th Year

Mukesh Patel School of Technology Management & Engineering, Mumbai

Task 1: Make Interactive **US Trade Monitor: Tariffs & Trade Deficits**

Code:

```
import streamlit as st
import pandas as pd
import plotly.express as px
import requests
import datetime

# ---------------------------------------------------------
# 1. CONFIGURATION & PAGE SETUP
# ---------------------------------------------------------
st.set_page_config(layout="wide", page_title="US Trade & Tariff Tracker")

st.title("US Trade Monitor: Tariffs & Trade Deficits")
st.markdown("""
This dashboard visualizes **US Trade Deficits** (live data from US Census Bureau)
combined with **Tariff Rates**.
\n*Hover over a country to see the trade balance and tariff details.*
""")

# ---------------------------------------------------------
# 2. DATA FETCHING FUNCTIONS
# ---------------------------------------------------------

@st.cache_data(ttl=3600)
```

```
def get_census_trade_data():
    """
    Fetches the latest annual Import/Export data from the US Census Bureau API.
    Uses separate endpoints for Imports and Exports and merges them.
    """
    # 1. Dynamic Year Selection (Previous Year for full dataset)
    current_year = datetime.datetime.now().year
    target_year = current_year - 1

    # 2. Define Endpoints
    # We use the Harmonized System (HS) endpoint which allows country-level aggregation
    url_exports = "https://api.census.gov/data/timeseries/intltrade/exports/hs"
    url_imports = "https://api.census.gov/data/timeseries/intltrade/imports/hs"

    # 3. Define Parameters
    # CTY_CODE = Country Code, CTY_NAME = Country Name
    # ALL_VAL_YR = Total Export Value (Year to Date)
    # GEN_VAL_YR = General Import Value (Year to Date)

    params_exp = {
        'get': 'CTY_CODE,CTY_NAME,ALL_VAL_YR',
        'time': str(target_year)
    }

    params_imp = {
        'get': 'CTY_CODE,CTY_NAME,GEN_VAL_YR',
        'time': str(target_year)
    }
```

```
try:
    # --- Fetch Exports ---
    r_exp = requests.get(url_exports, params=params_exp)
    r_exp.raise_for_status()
    data_exp = r_exp.json()
    df_exp = pd.DataFrame(data_exp[1:], columns=data_exp[0]) # Skip header row

    # --- Fetch Imports ---
    r_imp = requests.get(url_imports, params=params_imp)
    r_imp.raise_for_status()
    data_imp = r_imp.json()
    df_imp = pd.DataFrame(data_imp[1:], columns=data_imp[0])

    # --- Process & Merge ---
    # Convert values to Billions (Census returns dollars)
    df_exp['Exports'] = pd.to_numeric(df_exp['ALL_VAL_YR']) / 1_000_000_000
    df_imp['Imports'] = pd.to_numeric(df_imp['GEN_VAL_YR']) / 1_000_000_000

    # Merge on Country Code (CTY_CODE is safer than Name)
    df_merged = pd.merge(df_exp[['CTY_CODE', 'CTY_NAME', 'Exports']],
                         df_imp[['CTY_CODE', 'Imports']],
                         on='CTY_CODE',
                         how='outer')

    # Fill NaN with 0
    df_merged.fillna(0, inplace=True)
```

```
        # Calculate Balance
        df_merged['US Trade Balance ($B)'] = df_merged['Exports'] - df_merged['Imports']

        # Clean up Country Names (Census names are ALL CAPS)
        df_merged['Country Name'] = df_merged['CTY_NAME'].str.title()

        # Fix specific names to match Plotly/Standard ISO mapping
        name_fixes = {
            'Korea, South': 'South Korea',
            'United Kingdom': 'United Kingdom',
            'China': 'China',
            'Russian Federation': 'Russia',
            'Vietnam': 'Vietnam',
            'Germany': 'Germany'
        }
        df_merged['Country Name'] = df_merged['Country Name'].replace(name_fixes)

        return df_merged

    except Exception as e:
        st.error(f"Error fetching data from Census Bureau: {e}")
        return pd.DataFrame()

def get_tariff_data():
    """
    Returns a DataFrame of Tariff policies.
    (Static data for demo purposes - update this list as policies change)
    """
```

```
    data = {
        'Country Name': ['Canada', 'Mexico', 'China', 'Germany', 'France', 'United Kingdom', 'India', 'Japan', 'South Korea', 'Brazil', 'Vietnam', 'Russia', 'Australia'],
        'Tariff Rate (%)': [0.0, 0.0, 19.3, 2.4, 2.6, 2.5, 3.2, 0.0, 0.0, 3.5, 2.8, 35.0, 0.0],
        'Category': ['FTA Partner', 'FTA Partner', 'Trade War', 'Normal Trade', 'Normal Trade', 'Normal Trade', 'Normal Trade', 'FTA Partner', 'FTA Partner', 'Normal Trade', 'Normal Trade', 'Sanctioned', 'FTA Partner'],
        'ISO_ALPHA_3': ['CAN', 'MEX', 'CHN', 'DEU', 'FRA', 'GBR', 'IND', 'JPN', 'KOR', 'BRA', 'VNM', 'RUS', 'AUS']
    }
    return pd.DataFrame(data)

# ----------------------------------------------------------
# 3. MAIN APPLICATION LOGIC
# ----------------------------------------------------------

# Load Data
with st.spinner('Fetching latest trade data from US Census Bureau...'):
    df_trade = get_census_trade_data()
    df_tariff = get_tariff_data()

if not df_trade.empty:
    # Merge Trade Data (Real-time) with Tariff Data (Static Policy)
    # We merge on 'Country Name' for this simple demo.
    # (In production, using ISO codes for everything is more robust)

    merged_df = pd.merge(df_tariff, df_trade, on="Country Name", how="left")

    # Handle missing trade data for countries in our tariff list
```

```
merged_df['US Trade Balance ($B)'] = merged_df['US Trade Balance ($B)'].fillna(0)
merged_df['Exports'] = merged_df['Exports'].fillna(0)
merged_df['Imports'] = merged_df['Imports'].fillna(0)

# ----------------------------------------------------------
# 4. PLOTLY MAP
# ----------------------------------------------------------
fig = px.choropleth(
    merged_df,
    locations="ISO_ALPHA_3",
    color="Tariff Rate (%)",
    hover_name="Country Name",
    color_continuous_scale=px.colors.sequential.Reds,
    projection="natural earth",
    title=f"<b>US Tariffs vs Trade Balance ({datetime.datetime.now().year - 1})</b>",
    hover_data={
        "ISO_ALPHA_3": False,
        "Category": True,
        "Tariff Rate (%)": ":.1f",
        "US Trade Balance ($B)": ":+$.2f", # Format: +$10.50 or -$20.00
        "Exports": ":$.1f",
        "Imports": ":$.1f"
    }
)

fig.update_layout(
    margin={"r":0,"t":40,"l":0,"b":0},
    height=600,
```

```
        geo=dict(
            showframe=False,
            showcoastlines=True,
            bgcolor='rgba(0,0,0,0)'
        ),
        coloraxis_colorbar=dict(
            title="Tariff Rate",
            ticksuffix="%"
        )
    )

    # Display in Streamlit
    st.plotly_chart(fig, use_container_width=True)

    # ---------------------------------------------------------
    # 5. DATA TABLE
    # ---------------------------------------------------------
    with st.expander("📊 View Detailed Data Table"):
        display_cols = ['Country Name', 'Category', 'Tariff Rate (%)', 'US Trade Balance ($B)',
'Exports', 'Imports']
        st.dataframe(merged_df[display_cols].style.format({
            "Tariff Rate (%)": "{:.1f}%",
            "US Trade Balance ($B)": "${:+,.2f}B",
            "Exports": "${:,.2f}B",
            "Imports": "${:,.2f}B"
        }))

else:
```

st.warning("Data currently unavailable. The Census API might be down or rate-limited.")

Steps to run the file in command prompt:

# 1. Install required Python packages

python -m pip install streamlit pandas plotly requests

# 2. Navigate to the project folder

Example:  cd C:\KPMG   # or your cloned repo path

# 3. Run the Streamlit application

python -m streamlit run Task_1.py

Output:


Task 2:

Create an infographic showing the quick takeaways of DPDPA act in India. Make it interesting to look at with colorful doodles and icons.

Python Code:

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import textwrap

def draw_professional_dashboard_v2():
    # 1. SETUP CANVAS
    fig, ax = plt.subplots(figsize=(16, 10), dpi=120)

    # Professional Corporate Background
    bg_color = '#f4f6f8'
    fig.patch.set_facecolor(bg_color)
    ax.set_facecolor(bg_color)
    ax.set_xlim(0, 100)
    ax.set_ylim(0, 100)
    ax.axis('off')

    # ------------------------------------------------------------
    # 2. HELPER: DRAW ICONS INSIDE BADGES
    # ------------------------------------------------------------
    def draw_badge_icon(ax, center_x, center_y, icon_type, color):
        radius = 3.5
        badge = patches.Circle((center_x, center_y), radius, fc='white', ec='#bdc3c7', lw=1,
zorder=5)
        ax.add_patch(badge)

        if icon_type == 'user':
            ax.add_patch(patches.Circle((center_x, center_y+0.5), 1.2, fc=color, zorder=6))
```

```
        ax.add_patch(patches.Wedge((center_x, center_y-2.5), 2.2, 0, 180, fc=color, zorder=6))

    elif icon_type == 'building':
        ax.add_patch(patches.Rectangle((center_x-1.5, center_y-2), 3, 3.5, fc=color, zorder=6))
        ax.add_patch(patches.Polygon([(center_x-2, center_y+1.5), (center_x+2, center_y+1.5), (center_x, center_y+3)], fc=color, zorder=6))
        ax.add_patch(patches.Rectangle((center_x-0.5, center_y-2), 1, 1.2, fc='white', zorder=7))

    elif icon_type == 'doc':
        ax.add_patch(patches.Rectangle((center_x-1.5, center_y-2), 3, 4, fc=color, zorder=6))
        for i in range(3):
            ax.add_patch(patches.Rectangle((center_x-1, center_y-1 + (i*1)), 2, 0.3, fc='white', zorder=7))
        ax.add_patch(patches.Circle((center_x+1.5, center_y-1.5), 1, fc='#27ae60', zorder=8))

    elif icon_type == 'alert':
        ax.add_patch(patches.Polygon([(center_x, center_y+2), (center_x-2, center_y-2), (center_x+2, center_y-2)], fc=color, zorder=6))
        ax.add_patch(patches.Rectangle((center_x-0.2, center_y-0.5), 0.4, 1.5, fc='white', zorder=7))
        ax.add_patch(patches.Circle((center_x, center_y-1.2), 0.3, fc='white', zorder=7))

# ---------------------------------------------------------
# 3. HELPER: DRAW CLEAN UI CARDS
# ---------------------------------------------------------
def draw_ui_card(x, y, w, h, color, title, subtitle, points, icon_type):
    # Shadow
```

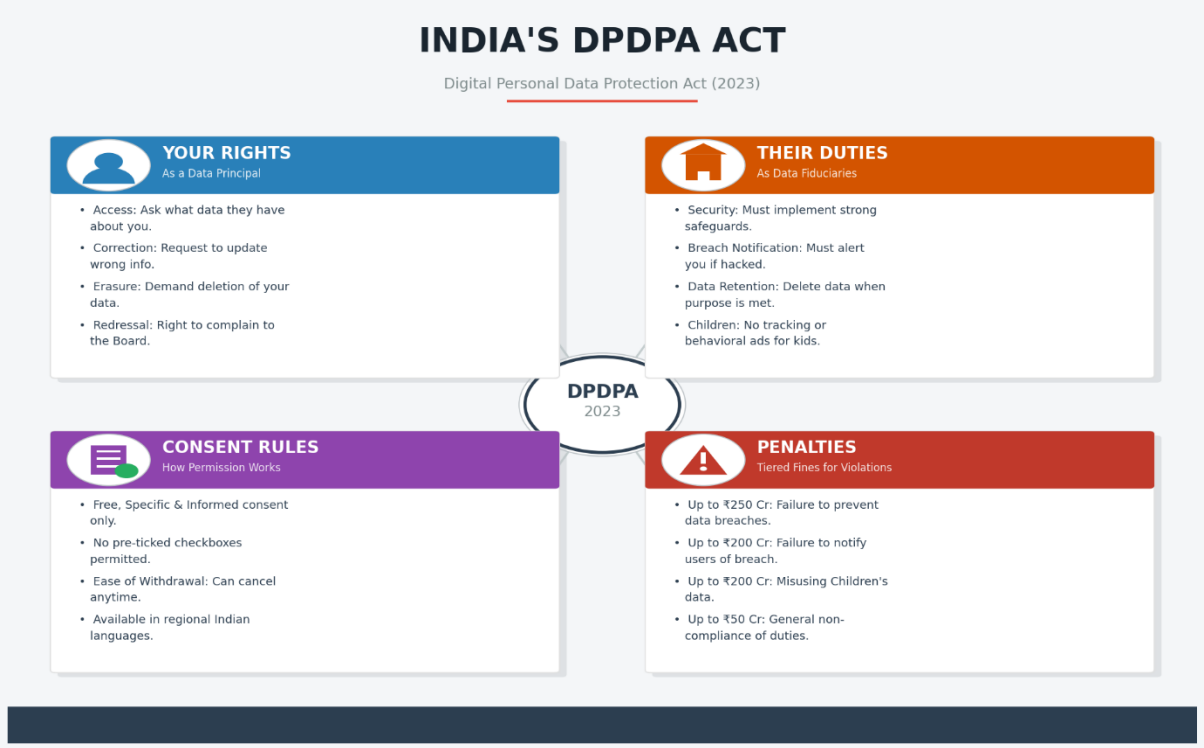
```
    shadow = patches.FancyBboxPatch((x+0.6, y-0.6), w, h, boxstyle="round,pad=0.4",
                                    fc='#bdc3c7', ec='none', alpha=0.4, zorder=2)
    ax.add_patch(shadow)

    # Main Box
    card = patches.FancyBboxPatch((x, y), w, h, boxstyle="round,pad=0.4",
                                  fc='white', ec='#e0e0e0', lw=1, zorder=3)
    ax.add_patch(card)

    # Header Strip
    header_h = 7
    header = patches.FancyBboxPatch((x, y + h - header_h), w, header_h,
                                    boxstyle="round,pad=0.4",
                                    fc=color, ec='none', zorder=4)
    ax.add_patch(header)
    ax.add_patch(patches.Rectangle((x, y + h - header_h), w, 3, fc=color, zorder=4))

    # Titles
    plt.text(x + 9, y + h - 2.5, title, fontsize=16, weight='bold', color='white', zorder=6, fontname="DejaVu Sans")
    plt.text(x + 9, y + h - 5, subtitle, fontsize=10, color='white', alpha=0.9, zorder=6, fontname="DejaVu Sans")

    # Icon
    draw_badge_icon(ax, x + 4.5, y + h - 3.5, icon_type, color)

    # Content Text
    y_pos = y + h - 10
    for point in points:
```

```
        wrapper = textwrap.TextWrapper(width=34)
        lines = wrapper.wrap(text=point)
        for i, line in enumerate(lines):
            bullet = "•" if i == 0 else " "
            plt.text(x + 2, y_pos, f"{bullet}  {line}",
                     fontsize=11, color='#2c3e50', zorder=6, fontname="DejaVu Sans")
            y_pos -= 2.2
        y_pos -= 0.8

    # ------------------------------------------------------
    # 4. DRAW LAYOUT
    # ------------------------------------------------------

    # HEADER
    plt.text(50, 94, "INDIA'S DPDPA ACT", ha='center', fontsize=32, weight='bold',
color='#1a252f', fontname="DejaVu Sans")
    plt.text(50, 89, "Digital Personal Data Protection Act (2023)", ha='center', fontsize=14,
color='#7f8c8d', fontname="DejaVu Sans")
    ax.add_patch(patches.Rectangle((42, 87), 16, 0.4, fc='#e74c3c'))

    # CARD 1: RIGHTS (Blue)
    draw_ui_card(4, 50, 42, 32, '#2980b9', "YOUR RIGHTS", "As a Data Principal",
            [
                "Access: Ask what data they have about you.",
                "Correction: Request to update wrong info.",
                "Erasure: Demand deletion of your data.",
                "Redressal: Right to complain to the Board."
            ], 'user')
```

```
# CARD 2: DUTIES (Orange)
draw_ui_card(54, 50, 42, 32, '#d35400', "THEIR DUTIES", "As Data Fiduciaries",
        [
            "Security: Must implement strong safeguards.",
            "Breach Notification: Must alert you if hacked.",
            "Data Retention: Delete data when purpose is met.",
            "Children: No tracking or behavioral ads for kids."
        ], 'building')

# CARD 3: CONSENT (Purple)
draw_ui_card(4, 10, 42, 32, '#8e44ad', "CONSENT RULES", "How Permission Works",
        [
            "Free, Specific & Informed consent only.",
            "No pre-ticked checkboxes permitted.",
            "Ease of Withdrawal: Can cancel anytime.",
            "Available in regional Indian languages."
        ], 'doc')

# --- UPDATED PENALTIES CARD (Red) ---
# Now includes the Tiers: 250, 200, and 50.
draw_ui_card(54, 10, 42, 32, '#c0392b', "PENALTIES", "Tiered Fines for Violations",
        [
            "Up to ₹250 Cr: Failure to prevent data breaches.",
            "Up to ₹200 Cr: Failure to notify users of breach.",
            "Up to ₹200 Cr: Misusing Children's data.",
            "Up to ₹50 Cr: General non-compliance of duties."
        ], 'alert')
```

```
    # CENTER HUB
    center_bg = patches.Circle((50, 46), 7, fc='white', ec='#bdc3c7', lw=1, zorder=1)
    ax.add_patch(center_bg)
    center_ring = patches.Circle((50, 46), 6.5, fc='white', ec='#2c3e50', lw=3, zorder=2)
    ax.add_patch(center_ring)
    plt.text(50, 47, "DPDPA", ha='center', fontsize=18, weight='bold', color='#2c3e50',
fontname="DejaVu Sans")
    plt.text(50, 44.5, "2023", ha='center', fontsize=14, color='#7f8c8d', fontname="DejaVu
Sans")

    # Connectors
    conn_style = dict(arrowstyle="-", color="#95a5a6", lw=2, zorder=0, alpha=0.5)
    ax.add_patch(patches.FancyArrowPatch((50, 46), (46, 55), **conn_style))
    ax.add_patch(patches.FancyArrowPatch((50, 46), (54, 55), **conn_style))
    ax.add_patch(patches.FancyArrowPatch((50, 46), (46, 37), **conn_style))
    ax.add_patch(patches.FancyArrowPatch((50, 46), (54, 37), **conn_style))

    # FOOTER
    ax.add_patch(patches.Rectangle((0, 0), 100, 5, fc='#2c3e50', zorder=10))
    plt.text(50, 2.5, " ⚠️ EXEMPTION: Government agencies can process data without consent
for National Security & Public Order.",
             ha='center', va='center', fontsize=12, color='white', weight='bold', fontname="DejaVu
Sans")

    plt.tight_layout()
    plt.savefig('dpdpa_tiered_penalties.png', dpi=150)
    plt.show()

if __name__ == "__main__":
```

draw_professional_dashboard_v2()




Canva: