**Week 5 - Lab Exercise: Interactive Data Visualization**

**Objective**

In this lab, you will implement **interactive data visualizations** using Python. You will apply:

* **Filtering**: Allow users to refine the dataset using dropdown selections.
* **Drill-down**: Enable hierarchical navigation (e.g., Year → Month).
* **Interactive Charts**: Create **dynamic bar charts and line charts** that update based on user interactions.

This lab will help you understand how interactive visualizations **enhance data exploration and decision-making**.

**Dataset**

For this exercise, we will use a **Sales Transactions dataset**. If you do not have a dataset, you can download a sample dataset from:  
📌 **Kaggle - Sample Superstore Sales Data:** [**https://www.kaggle.com/datasets/bravehart101/sample-supermarket-dataset**](https://www.kaggle.com/datasets/bravehart101/sample-supermarket-dataset)

Alternatively, create a **sample dataset** using the following Python code:

#python

import pandas as pd

import numpy as np

# Generate a sample dataset

np.random.seed(42)

dates = pd.date\_range(start="2020-01-01", periods=300, freq="D")

regions = ["North America", "Europe", "Asia", "Australia"]

categories = ["Electronics", "Clothing", "Home Goods", "Toys"]

sales\_amount = np.random.randint(100, 10000, size=len(dates))

df = pd.DataFrame({

"Date": np.random.choice(dates, size=len(dates)),

"Region": np.random.choice(regions, size=len(dates)),

"Category": np.random.choice(categories, size=len(dates)),

"Sales": sales\_amount

})

df.to\_csv("sales\_data.csv", index=False)

print("Sample dataset saved as 'sales\_data.csv'.")

The Kaggle dataset contains **sales transactions** with four columns:

* **Date** (YYYY-MM-DD format)
* **Region** (Sales from different regions)
* **Category** (Product type)
* **Sales** (Revenue generated)

✅ **Task 1: Load the dataset**  
Run the following code to load the dataset and inspect the first few rows:

python

df = pd.read\_csv("sales\_data.csv")

print(df.head())

**Step 1: Create a Basic Visualization**

First, create a simple bar chart to visualize **total sales per category**.

python

import plotly.express as px

# Aggregate total sales per category

sales\_by\_category = df.groupby("Category")["Sales"].sum().reset\_index()

# Create a bar chart

fig = px.bar(sales\_by\_category, x="Category", y="Sales", title="Total Sales by Category")

# Show the chart

fig.show()

✅ **Task 2: Modify the chart to show sales per region instead of category.**

**Step 2: Implement Filtering (Dropdown Selection)**

Now, let’s allow users to **filter sales by region**. We will use Dash to create an interactive dashboard.

#python

import dash

from dash import dcc, html

from dash.dependencies import Input, Output

import plotly.express as px

# Initialize Dash app

app = dash.Dash(\_\_name\_\_)

# App Layout

app.layout = html.Div([

html.H1("Interactive Sales Dashboard"),

# Dropdown for region selection

dcc.Dropdown(

id="region-filter",

options=[{"label": r, "value": r} for r in df["Region"].unique()],

value="North America",

clearable=False,

),

# Graph output

dcc.Graph(id="sales-chart"),

])

# Callback to update chart based on selection

@app.callback(

Output("sales-chart", "figure"),

[Input("region-filter", "value")]

)

def update\_chart(selected\_region):

filtered\_df = df[df["Region"] == selected\_region]

sales\_by\_category = filtered\_df.groupby("Category")["Sales"].sum().reset\_index()

fig = px.bar(sales\_by\_category, x="Category", y="Sales", title=f"Sales in {selected\_region}")

return fig

# Run the app

if \_\_name\_\_ == "\_\_main\_\_":

app.run\_server(debug=True)

✅ **Task 3: Modify the dropdown to allow selecting multiple regions.**

**Step 3: Implement Drill-down (Year → Month)**

Let’s create a drill-down system where:

* The first chart shows **total sales per year**
* Clicking on a year updates a second chart to show **monthly sales**

#python

# Convert Date column to datetime

df["Date"] = pd.to\_datetime(df["Date"])

df["Year"] = df["Date"].dt.year

df["Month"] = df["Date"].dt.month

# Aggregate yearly sales

yearly\_sales = df.groupby("Year")["Sales"].sum().reset\_index()

# Create interactive figure

fig = px.bar(yearly\_sales, x="Year", y="Sales", title="Yearly Sales")

fig.show()

✅ **Task 4: Modify the code so that clicking on a year updates a second chart to display monthly sales.**

**What’s Next?**

* **Once you complete the exercise, move on to solving the assignment.**