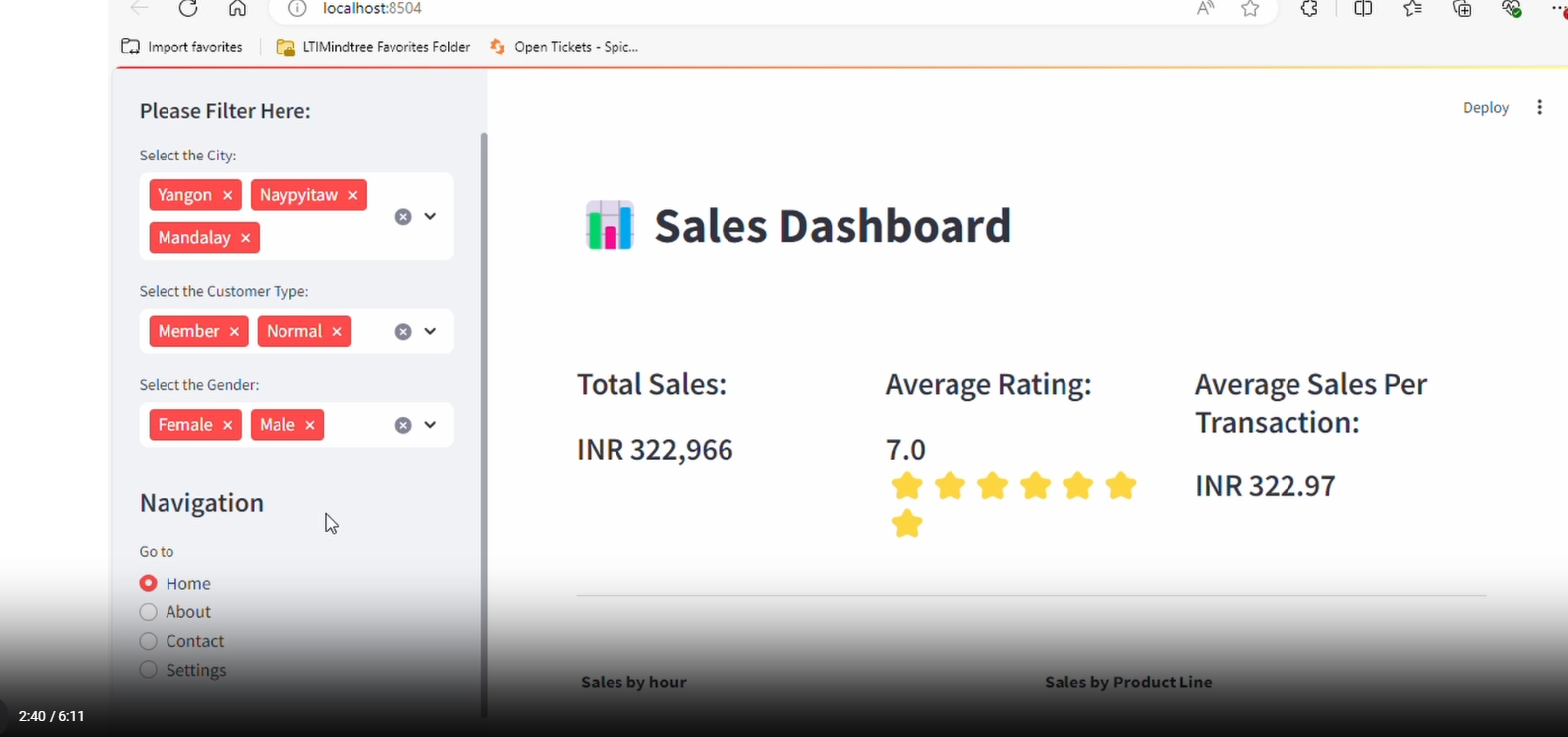
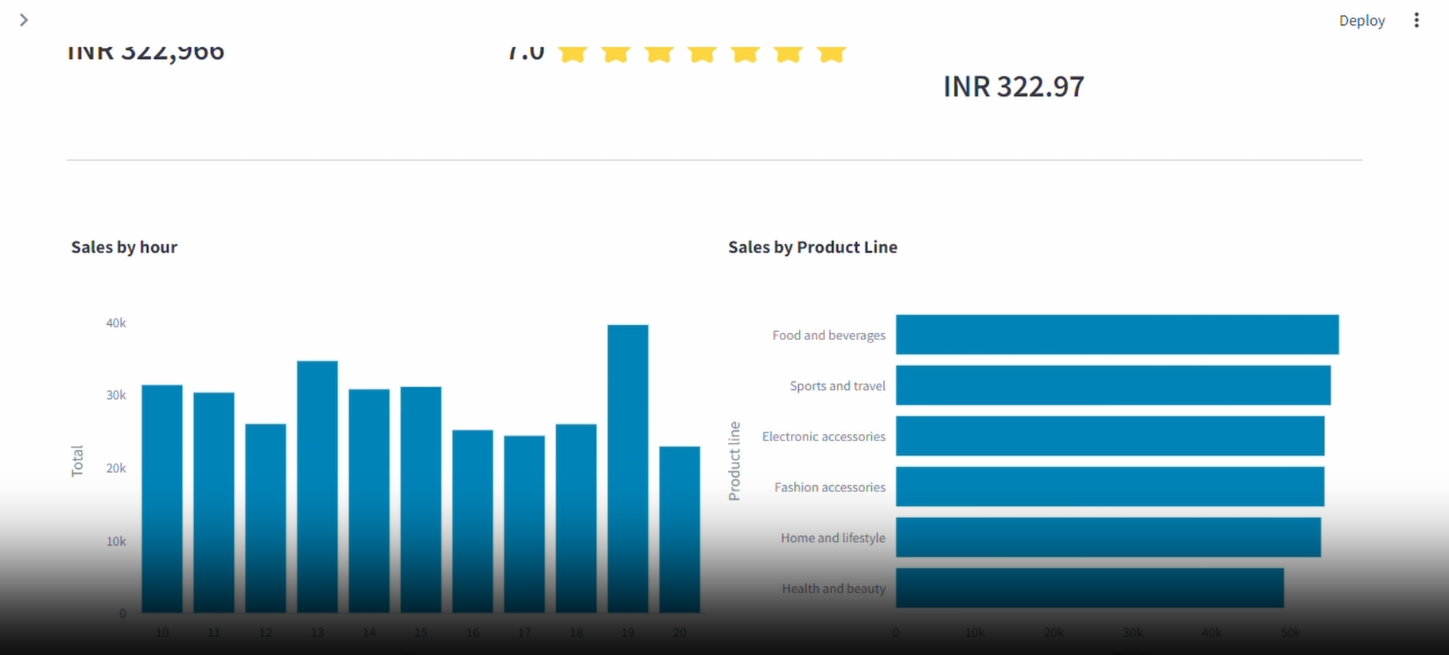
**Practice Question – 4**

**Streamlit with Multipage Dashboard**

03.09.2025

Consider the following code for Supermarket.xlsx dataset and create a multipage dashboard using streamlit:





import pandas as pd

import plotly.express as px

import streamlit as st

# Set Streamlit page configuration

st.set\_page\_config(page\_title="Sales Dashboard", page\_icon=":bar\_chart:", layout="wide")

# Function to load data from Excel

@st.cache(allow\_output\_mutation=True)

def get\_data\_from\_excel():

try:

df = pd.read\_excel(

io="supermarkt\_sales.xlsx",

engine="openpyxl",

sheet\_name="Sales",

skiprows=0, # Adjust skiprows if needed

nrows=100, # Adjust nrows if needed

)

# Check if 'Time' column exists in the dataframe

if 'Time' in df.columns:

df["Time"] = pd.to\_datetime(df["Time"], format="%H:%M:%S").dt.time

else:

st.error("Error: 'Time' column not found in the Excel data.")

st.stop() # Stop execution if 'Time' column is missing

return df

except FileNotFoundError:

st.error("Error: File 'supermarkt\_sales.xlsx' not found.")

st.stop() # Stop execution if file is not found

except Exception as e:

st.error(f"An error occurred: {str(e)}")

st.stop() # Stop execution for any other unexpected errors

# Load data from Excel

df = get\_data\_from\_excel()

# Sidebar filters

st.sidebar.header("Please Filter Here:")

city = st.sidebar.multiselect(

"Select the City:",

options=df["City"].unique(),

default=df["City"].unique()

)

customer\_type = st.sidebar.multiselect(

"Select the Customer Type:",

options=df["Customer type"].unique(),

default=df["Customer type"].unique(),

)

gender = st.sidebar.multiselect(

"Select the Gender:",

options=df["Gender"].unique(),

default=df["Gender"].unique()

)

st.sidebar.title("Navigation")

page = st.sidebar.radio("Go to", ["Home", "About", "Contact", "Settings"])

# Filter dataframe based on sidebar selections

# Ensure correct column names and string formatting in query

query\_string = f"City == {city} and Customer type == {customer\_type} and Gender == {gender}"

df\_selection = df.query(query\_string)

# Check if filtered dataframe is empty

if df\_selection.empty:

st.warning("No data available based on the current filter settings!")

st.stop() # Stop execution if filtered dataframe is empty

# Main page content

st.title(":bar\_chart: Sales Dashboard")

st.markdown("##")

# Top KPIs

total\_sales = int(df\_selection["Total"].sum())

average\_rating = round(df\_selection["Rating"].mean(), 1)

star\_rating = ":star:" \* int(round(average\_rating, 0))

average\_sale\_by\_transaction = round(df\_selection["Total"].mean(), 2)

# Display top KPIs in columns

left\_column, middle\_column, right\_column = st.columns(3)

with left\_column:

st.subheader("Total Sales:")

st.subheader(f"INR {total\_sales:,}")

with middle\_column:

st.subheader("Average Rating:")

st.subheader(f"{average\_rating} {star\_rating}")

with right\_column:

st.subheader("Average Sales Per Transaction:")

st.subheader(f"INR {average\_sale\_by\_transaction}")

# Separator line

st.markdown("""---""")

# Sales by Product Line (Bar chart)

sales\_by\_product\_line = df\_selection.groupby(by=["Product line"])[["Total"]].sum().sort\_values(by="Total")

fig\_product\_sales = px.bar(

sales\_by\_product\_line,

x="Total",

y=sales\_by\_product\_line.index,

orientation="h",

title="<b>Sales by Product Line</b>",

color\_discrete\_sequence=["#0083B8"] \* len(sales\_by\_product\_line),

template="plotly\_white",

)

fig\_product\_sales.update\_layout(

plot\_bgcolor="rgba(0,0,0,0)",

xaxis=dict(showgrid=False)

)

# Sales by Hour (Bar chart)

sales\_by\_hour = df\_selection.groupby(by=["Time"])[["Total"]].sum()

fig\_hourly\_sales = px.bar(

sales\_by\_hour,

x=sales\_by\_hour.index,

y="Total",

title="<b>Sales by Hour</b>",

color\_discrete\_sequence=["#0083B8"] \* len(sales\_by\_hour),

template="plotly\_white",

)

fig\_hourly\_sales.update\_layout(

xaxis=dict(tickmode="linear"),

plot\_bgcolor="rgba(0,0,0,0)",

yaxis=dict(showgrid=False),

)

# Display charts in two columns

left\_column, right\_column = st.columns(2)

left\_column.plotly\_chart(fig\_hourly\_sales, use\_container\_width=True)

right\_column.plotly\_chart(fig\_product\_sales, use\_container\_width=True)

# Hide Streamlit style (optional)

# hide\_st\_style = """

# <style>

# #MainMenu {visibility: hidden;}

# footer {visibility: hidden;}

# header {visibility: show;}

# </style>

# """

# st.markdown(hide\_st\_style, unsafe\_allow\_html=True)