Final code

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Paragraph

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.graphics.shapes import Drawing

from reportlab.graphics.charts.barcharts import VerticalBarChart

from io import BytesIO

import pyodbc

import pandas as pd

# Function to send email with PDF attachment

def send\_email\_with\_pdf\_attachment(pdf\_content, receiver\_email):

# Email configuration

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

sender\_email = 'nallagarivishnuvardhanreddy@gmail.com'

sender\_password = 'cchk mwpc vuml iazz'

subject = 'PDF Attachment'

# Create a multipart message

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = receiver\_email

message['Subject'] = subject

# Attach PDF content

part = MIMEBase('application', 'octet-stream')

part.set\_payload(pdf\_content.getvalue())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', 'attachment', filename="combined\_tables.pdf")

message.attach(part)

# Connect to SMTP server and send email

with smtplib.SMTP(smtp\_server, smtp\_port) as server:

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, receiver\_email, message.as\_string())

print(f'Email sent successfully to {receiver\_email}')

# Establish connection to the SQL Server database

server = '10.0.1.71,4000'

database = 'HERITAGEBI'

username = 'HFLSQLReader'

password = 'HFL@12345'

driver = '{ODBC Driver 17 for SQL Server}'

conn\_str = f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

conn = pyodbc.connect(conn\_str)

# Create a cursor object to execute SQL queries

cursor = conn.cursor()

# Execute the first SQL query

query1 = '''SELECT state, city, SalesOfficeID, salesquantity, tops, plant\_name

FROM (SELECT m.state, m.city, s.SalesOfficeID, s.salesquantity,

RANK() OVER (ORDER BY s.salesquantity DESC) AS tops, m.plant\_name

FROM dw.fSales s INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE BillingDate = '2024-02-05') AS ranked\_sales

WHERE tops <= 10 ORDER BY salesquantity DESC'''

cursor.execute(query1)

rows1 = cursor.fetchall()

df1 = pd.DataFrame([list(row) for row in rows1], columns=[desc[0] for desc in cursor.description])

# Execute the second SQL query

query2 = '''SELECT state, city, SalesOfficeID, salesquantity, tops, plant\_name

FROM (SELECT m.state, m.city, s.SalesOfficeID, s.salesquantity,

RANK() OVER (ORDER BY s.salesquantity) AS tops, m.plant\_name

FROM dw.fSales s INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE BillingDate = '2024-02-05') AS ranked\_sales

WHERE tops <= 10 ORDER BY salesquantity DESC'''

cursor.execute(query2)

rows2 = cursor.fetchall()

df2 = pd.DataFrame([list(row) for row in rows2], columns=[desc[0] for desc in cursor.description])

# Close cursor and connection

cursor.close()

conn.close()

# Specify the output file path for the PDF

output\_file = "combined\_tables.pdf"

# Create a PDF document

pdf\_buffer = BytesIO()

pdf = SimpleDocTemplate(pdf\_buffer)

# Create styles for paragraphs

styles = getSampleStyleSheet()

title\_style = styles['Heading1']

normal\_style = styles['Normal']

# Define data for tables

data1 = [df1.columns.values.tolist()] + df1.values.tolist()

data2 = [df2.columns.values.tolist()] + df2.values.tolist()

# Add the tables and charts to the PDF content

content = []

# Add title and table 1

content.append(Paragraph("Top 10 Sales Report:", title\_style))

table1 = Table(data1)

table1.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

image\_path = "C:\\Users\\HP\\Downloads\\heritage.cms"

image = Image(image\_path, width=letter[0], height=150)

content.append(image)

# Add bar chart for table 1

top\_chart\_data = df1['tops'].tolist()

top\_category\_names = df1['SalesOfficeID'].astype(str).tolist()

top\_drawing = Drawing(400, 200)

top\_bc = VerticalBarChart()

top\_bc.x = 50

top\_bc.y = 40

top\_bc.height = 150

top\_bc.width = 300

top\_bc.data = [top\_chart\_data]

top\_bc.categoryAxis.labels.angle = 45

top\_bc.categoryAxis.categoryNames = top\_category\_names

top\_bc.bars[0].fillColor = colors.blue

for i, label in enumerate(top\_category\_names):

top\_bc.bars[i].name = label

top\_drawing.add(top\_bc)

content.append(table1)

content.append(top\_drawing)

# Add title and table 2

content.append(Paragraph("Bottom 10 Sales Report:", title\_style))

table2 = Table(data2)

table2.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

# Add bar chart for table 2

bottom\_chart\_data = df2['tops'].tolist()

bottom\_category\_names = df2['SalesOfficeID'].astype(str).tolist()

bottom\_drawing = Drawing(400, 200)

bottom\_bc = VerticalBarChart()

bottom\_bc.x = 50

bottom\_bc.y = 40

bottom\_bc.height = 150

bottom\_bc.width = 300

bottom\_bc.data = [bottom\_chart\_data]

bottom\_bc.categoryAxis.labels.angle = 45

bottom\_bc.categoryAxis.categoryNames = bottom\_category\_names

bottom\_bc.bars[0].fillColor = colors.green

for i, label in enumerate(bottom\_category\_names):

bottom\_bc.bars[i].name = label

bottom\_drawing.add(bottom\_bc)

content.append(table2)

content.append(bottom\_drawing)

# Build the PDF document

pdf.build(content)

# Rewind the buffer

pdf\_buffer.seek(0)

# Send email with PDF attachment

receiver\_email = 'vnallagari1919@gmail.com' # Update with recipient's email address

send\_email\_with\_pdf\_attachment(pdf\_buffer, receiver\_email)

print(f"Data exported to PDF and sent via email to {receiver\_email}")

9-2-24

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Paragraph

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.graphics.shapes import Drawing

from reportlab.graphics.charts.barcharts import VerticalBarChart

from io import BytesIO

import pyodbc

import pandas as pd

# Function to send email with PDF attachment

def send\_email\_with\_pdf\_attachment(pdf\_content, receiver\_email):

# Email configuration

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

sender\_email = 'nallagarivishnuvardhanreddy@gmail.com'

sender\_password = 'cchk mwpc vuml iazz'

subject = 'PDF Attachment'

# Create a multipart message

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = receiver\_email

message['Subject'] = subject

# Attach PDF content

part = MIMEBase('application', 'octet-stream')

part.set\_payload(pdf\_content.getvalue())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', 'attachment', filename="combined\_tables.pdf")

message.attach(part)

# Connect to SMTP server and send email

with smtplib.SMTP(smtp\_server, smtp\_port) as server:

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, receiver\_email, message.as\_string())

print(f'Email sent successfully to {receiver\_email}')

# Establish connection to the SQL Server database

server = '10.0.1.71,4000'

database = 'HERITAGEBI'

username = 'HFLSQLReader'

password = 'HFL@12345'

driver = '{ODBC Driver 17 for SQL Server}'

conn\_str = f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

conn = pyodbc.connect(conn\_str)

# Create a cursor object to execute SQL queries

cursor = conn.cursor()

# Execute the first SQL query

query1 = '''SELECT state, city, SalesOfficeID, plant\_name as salesoffice\_Name, cast(salesquantity as int) AS salesquantity , tops

FROM (SELECT m.state, m.city, s.SalesOfficeID, s.salesquantity,

RANK() OVER (ORDER BY s.salesquantity DESC) AS tops, m.plant\_name

FROM dw.fSales s INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE BillingDate = '2024-02-05') AS ranked\_sales

WHERE tops <= 10 ORDER BY salesquantity DESC'''

cursor.execute(query1)

rows1 = cursor.fetchall()

df1 = pd.DataFrame([list(row) for row in rows1], columns=[desc[0] for desc in cursor.description])

# Execute the second SQL query

query2 = '''SELECT state, city, SalesOfficeID,plant\_name as salesoffice\_Name, cast(salesquantity as int) AS salesquantity, tops

FROM (SELECT m.state, m.city, s.SalesOfficeID, s.salesquantity,

RANK() OVER (ORDER BY s.salesquantity) AS tops, m.plant\_name

FROM dw.fSales s INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE BillingDate = '2024-02-05') AS ranked\_sales

WHERE tops <= 10 ORDER BY salesquantity DESC'''

cursor.execute(query2)

rows2 = cursor.fetchall()

df2 = pd.DataFrame([list(row) for row in rows2], columns=[desc[0] for desc in cursor.description])

# Close cursor and connection

cursor.close()

conn.close()

# Specify the output file path for the PDF

output\_file = "combined\_tables.pdf"

# Create a PDF document

pdf\_buffer = BytesIO()

pdf = SimpleDocTemplate(pdf\_buffer)

# Create styles for paragraphs

styles = getSampleStyleSheet()

title\_style = styles['Heading1']

normal\_style = styles['Normal']

# Define data for tables

data1 = [df1.columns.values.tolist()] + df1.values.tolist()

data2 = [df2.columns.values.tolist()] + df2.values.tolist()

# Add the tables and charts to the PDF content

content = []

image\_path = "C:\\Users\\HP\\Downloads\\heritage.cms"

image = Image(image\_path, width=200, height=200)

content.append(image)

title\_style = getSampleStyleSheet()["Heading1"]

title\_style.fontSize = 18 # Adjust font size as needed

title\_style.textColor = colors.black # Change color to black or any other color you prefer

# Add the text with the modified style

content.append(Paragraph("HERITAGE FOOD Pvt.Ltd", title\_style))

# Add title and table 1

content.append(Paragraph("Top 10 Sales Report:", title\_style))

table1 = Table(data1)

table1.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

top\_chart\_data = sorted(df1['tops'].tolist(), reverse=True)

top\_category\_names = sorted(df1['SalesOfficeID'].astype(str).tolist(), reverse=True)

top\_drawing = Drawing(400, 200)

top\_bc = VerticalBarChart()

top\_bc.x = 50

top\_bc.y = 40

top\_bc.height = 150

top\_bc.width = 300

top\_bc.data = [top\_chart\_data]

top\_bc.categoryAxis.labels.angle =25

top\_bc.categoryAxis.labels.fontSize = 12

top\_bc.categoryAxis.categoryNames = top\_category\_names

top\_bc.bars[0].fillColor = colors.green

for i, label in enumerate(top\_category\_names):

top\_bc.bars[i].name = label

top\_drawing.add(top\_bc)

content.append(table1)

content.append(top\_drawing)

# Add title and table 2

content.append(Paragraph("Bottom 10 Sales Report:", title\_style))

table2 = Table(data2)

table2.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

# Add bar chart for table 2

bottom\_chart\_data = sorted(df2['tops'].tolist(),reverse=False)

bottom\_category\_names = sorted(df2['SalesOfficeID'].astype(str).tolist(),reverse=False)

bottom\_drawing = Drawing(400, 200)

bottom\_bc = VerticalBarChart()

bottom\_bc.x = 50

bottom\_bc.y = 40

bottom\_bc.height = 150

bottom\_bc.width = 300

bottom\_bc.data = [bottom\_chart\_data]

bottom\_bc.categoryAxis.labels.angle =25

bottom\_bc.categoryAxis.labels.fontSize = 12

bottom\_bc.categoryAxis.categoryNames = bottom\_category\_names

bottom\_bc.bars[0].fillColor = colors.blue

for i, label in enumerate(bottom\_category\_names):

bottom\_bc.bars[i].name = label

bottom\_drawing.add(bottom\_bc)

content.append(table2)

content.append(bottom\_drawing)

# Build the PDF document

pdf.build(content)

# Rewind the buffer

pdf\_buffer.seek(0)

# Send email with PDF attachment

receiver\_email = 'vnallagari1919@gmail.com' # Update with recipient's email address

send\_email\_with\_pdf\_attachment(pdf\_buffer, receiver\_email)

print(f"Data exported to PDF and sent via email to {receiver\_email}")

important final

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Paragraph

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.graphics.shapes import Drawing

from reportlab.graphics.charts.barcharts import VerticalBarChart

from io import BytesIO

from reportlab.graphics.shapes import Drawing, Line, String

import pyodbc

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Image

import pandas as pd

# Function to send email with PDF attachment

def send\_email\_with\_pdf\_attachment(pdf\_content, receiver\_email):

# Email configuration

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

sender\_email = 'nallagarivishnuvardhanreddy@gmail.com'

sender\_password = 'cchk mwpc vuml iazz'

subject = 'PDF Attachment'

# Create a multipart message

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = receiver\_email

message['Subject'] = subject

# Attach PDF content

part = MIMEBase('application', 'octet-stream')

part.set\_payload(pdf\_content.getvalue())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', 'attachment', filename="combined\_tables.pdf")

message.attach(part)

# Connect to SMTP server and send email

with smtplib.SMTP(smtp\_server, smtp\_port) as server:

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, receiver\_email, message.as\_string())

print(f'Email sent successfully to {receiver\_email}')

# Establish connection to the SQL Server database

server = '10.0.1.71,4000'

database = 'HERITAGEBI'

username = 'HFLSQLReader'

password = 'HFL@12345'

driver = '{ODBC Driver 17 for SQL Server}'

conn\_str = f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

conn = pyodbc.connect(conn\_str)

# Create a cursor object to execute SQL queries

cursor = conn.cursor()

import pandas as pd

# Your existing code

# Execute the first SQL query

query1 = '''SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales, tops

FROM (

SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales,

ROW\_NUMBER() OVER (ORDER BY total\_sales DESC) AS tops

FROM (

SELECT m.state, m.city, s.SalesOfficeID, m.plant\_name AS salesoffice\_Name, cast(SUM(s.salesquantity)as int) AS total\_sales

FROM dw.fSales s

INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE s.BillingDate = CAST(DATEADD(day, -1, GETDATE()) AS DATE)

GROUP BY m.state, m.city, s.SalesOfficeID, m.plant\_name

) AS ranked\_sales

) AS numbered\_sales

WHERE tops <= 10

ORDER BY total\_sales DESC;'''

cursor.execute(query1)

rows1 = cursor.fetchall()

df1 = pd.DataFrame([list(row) for row in rows1], columns=[desc[0] for desc in cursor.description])

# Execute the query to get average sales quantity

previousyear\_query = '''SELECT

SalesOfficeID,

cast(SUM(salesquantity)as int) /

DATEDIFF(day, DATEADD(year, -1, EOMONTH(GETDATE(), -1)), DATEADD(year, -1, EOMONTH(GETDATE(), 0))) AS AverageSalesQuantityPerDay

FROM

dw.fSales

WHERE

billingdate BETWEEN DATEADD(year, -1, DATEADD(day, 1, EOMONTH(GETDATE(), -1)))

AND DATEADD(year, -1, EOMONTH(GETDATE(), 0))

GROUP BY

SalesOfficeID;

'''

cursor.execute(previousyear\_query)

avg\_sales\_rows = cursor.fetchall()

avg\_sales\_quantity = pd.DataFrame([list(row) for row in avg\_sales\_rows], columns=['SalesOfficeID', 'AverageSalesQuantity'])

# Merge the average sales quantity column to df1 based on SalesOfficeID

df1 = pd.merge(df1, avg\_sales\_quantity, on='SalesOfficeID', how='left')

df1['sales\_difference'] = df1['total\_sales'] - df1['AverageSalesQuantity']

# Execute the second SQL query

query2 = '''SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales, tops

FROM (

SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales,

ROW\_NUMBER() OVER (ORDER BY total\_sales ) AS tops

FROM (

SELECT m.state, m.city, s.SalesOfficeID, m.plant\_name AS salesoffice\_Name, cast(SUM(s.salesquantity)as int) AS total\_sales

FROM dw.fSales s

INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE s.BillingDate = CAST(DATEADD(day, -1, GETDATE()) AS DATE)

GROUP BY m.state, m.city, s.SalesOfficeID, m.plant\_name

) AS ranked\_sales

) AS numbered\_sales

WHERE tops <= 10

ORDER BY total\_sales desc ;

'''

cursor.execute(query2)

rows2 = cursor.fetchall()

df2 = pd.DataFrame([list(row) for row in rows2], columns=[desc[0] for desc in cursor.description])

cursor.execute(previousyear\_query)

avg\_sales\_rows = cursor.fetchall()

avg\_sales\_quantity = pd.DataFrame([list(row) for row in avg\_sales\_rows], columns=['SalesOfficeID', 'AverageSalesQuantity'])

# Merge the average sales quantity column to df1 based on SalesOfficeID

df2 = pd.merge(df2, avg\_sales\_quantity, on='SalesOfficeID', how='left')

# Calculate the difference between total\_sales and average sales quantity

df2['sales\_difference'] = df2['total\_sales'] - df2['AverageSalesQuantity']

cursor.close()

conn.close()

# Specify the output file path for the PDF

output\_file = "combined\_tables.pdf"

custom\_page\_size = (800, 600) # Width x Height in points

# Create a PDF document

pdf\_buffer = BytesIO()

pdf = SimpleDocTemplate(pdf\_buffer, pagesize=custom\_page\_size)

# Create styles for paragraphs

styles = getSampleStyleSheet()

title\_style = styles['Heading1']

normal\_style = styles['Normal']

# Define data for tables

data1 = [df1.columns.values.tolist()] + df1.values.tolist()

data2 = [df2.columns.values.tolist()] + df2.values.tolist()

# Add the tables and charts to the PDF content

content = []

image\_path = "C:\\Users\\HP\\Downloads\\heritage.cms"

image = Image(image\_path)

content.append(image)

title\_style = getSampleStyleSheet()["Heading1"]

title\_style.fontSize = 18 # Adjust font size as needed

title\_style.textColor = colors.black # Change color to black or any other color you prefer

# Add the text with the modified style

content.append(Paragraph("HERITAGE FOODs Pvt.Ltd", title\_style))

# Add title and table 1

content.append(Paragraph("Top 10 Sales Report:", title\_style))

table1 = Table(data1)

table1.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

top\_chart\_data = sorted(df1['tops'].tolist(), reverse=True)

top\_category\_names =df1['SalesOfficeID'].astype(str).tolist()

top\_drawing = Drawing(400, 200)

top\_bc = VerticalBarChart()

top\_bc.x = 50

top\_bc.y = 40

top\_bc.height = 150

top\_bc.width = 300

top\_bc.data = [top\_chart\_data]

top\_bc.categoryAxis.labels.angle =25

top\_bc.categoryAxis.labels.fontSize = 12

top\_bc.categoryAxis.categoryNames = top\_category\_names

top\_bc.bars[0].fillColor = colors.green

for i, label in enumerate(top\_category\_names):

top\_bc.bars[i].name = label

top\_drawing.add(top\_bc)

content.append(table1)

content.append(top\_drawing)

content.append(Paragraph("Bottom 10 Sales Report:", title\_style))

table2 = Table(data2)

table2.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

# Add bar chart for table 2

bottom\_chart\_data =sorted(df2['tops'].tolist(),reverse=False)

bottom\_category\_names =sorted(df2['SalesOfficeID'].astype(str).tolist(),reverse=True)

bottom\_drawing = Drawing(400, 200)

bottom\_bc = VerticalBarChart()

bottom\_bc.x = 50

bottom\_bc.y = 40

bottom\_bc.height = 150

bottom\_bc.width = 300

bottom\_bc.data = [bottom\_chart\_data]

bottom\_bc.categoryAxis.labels.angle =25

bottom\_bc.categoryAxis.labels.fontSize = 12

bottom\_bc.categoryAxis.categoryNames = bottom\_category\_names

bottom\_bc.bars[0].fillColor = colors.blue

for i, label in enumerate(bottom\_category\_names):

bottom\_bc.bars[i].name = label

bottom\_drawing.add(bottom\_bc)

content.append(table2)

content.append(bottom\_drawing)

# Build the PDF document

pdf.build(content)

# Rewind the buffer

pdf\_buffer.seek(0)

# Send email with PDF attachment

receiver\_email = 'vnallagari1919@gmail.com' # Update with recipient's email address

send\_email\_with\_pdf\_attachment(pdf\_buffer, receiver\_email)

print(f"Data exported to PDF and sent via email to {receiver\_email}")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Paragraph

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.graphics.shapes import Drawing

from reportlab.graphics.charts.barcharts import VerticalBarChart

from io import BytesIO

from reportlab.graphics.shapes import Drawing, Line, String

import pyodbc

from reportlab.lib import fonts

from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

from reportlab.platypus import Paragraph

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Image

import pandas as pd

from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

# Function to send email with PDF attachment

def send\_email\_with\_pdf\_attachment(pdf\_content, receiver\_email):

# Email configuration

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

sender\_email = 'nallagarivishnuvardhanreddy@gmail.com'

sender\_password = 'cchk mwpc vuml iazz'

subject = 'PDF Attachment'

# Create a multipart message

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = receiver\_email

message['Subject'] = subject

# Attach PDF content

part = MIMEBase('application', 'octet-stream')

part.set\_payload(pdf\_content.getvalue())

encoders.encode\_base64(part)

part.add\_header('Content-Disposition', 'attachment', filename="combined\_tables.pdf")

message.attach(part)

# Connect to SMTP server and send email

with smtplib.SMTP(smtp\_server, smtp\_port) as server:

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, receiver\_email, message.as\_string())

print(f'Email sent successfully to {receiver\_email}')

# Establish connection to the SQL Server database

server = '10.0.1.71,4000'

database = 'HERITAGEBI'

username = 'HFLSQLReader'

password = 'HFL@12345'

driver = '{ODBC Driver 17 for SQL Server}'

conn\_str = f'DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}'

conn = pyodbc.connect(conn\_str)

# Create a cursor object to execute SQL queries

cursor = conn.cursor()

import pandas as pd

# Your existing code

# Execute the first SQL query

query1 = '''SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales, tops

FROM (

SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales,

ROW\_NUMBER() OVER (ORDER BY total\_sales DESC) AS tops

FROM (

SELECT m.state, m.city, s.SalesOfficeID, m.plant\_name AS salesoffice\_Name, cast(SUM(s.salesquantity)as int) AS total\_sales

FROM dw.fSales s

INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE s.BillingDate = CAST(DATEADD(day, -1, GETDATE()) AS DATE)

GROUP BY m.state, m.city, s.SalesOfficeID, m.plant\_name

) AS ranked\_sales

) AS numbered\_sales

WHERE tops <= 10

ORDER BY total\_sales DESC;'''

cursor.execute(query1)

rows1 = cursor.fetchall()

df1 = pd.DataFrame([list(row) for row in rows1], columns=[desc[0] for desc in cursor.description])

# Execute the query to get average sales quantity

previousyear\_query = '''SELECT

SalesOfficeID,

cast(SUM(salesquantity)as int) /

DATEDIFF(day, DATEADD(year, -1, EOMONTH(GETDATE(), -1)), DATEADD(year, -1, EOMONTH(GETDATE(), 0))) AS AverageSalesQuantityPerDay

FROM

dw.fSales

WHERE

billingdate BETWEEN DATEADD(year, -1, DATEADD(day, 1, EOMONTH(GETDATE(), -1)))

AND DATEADD(year, -1, EOMONTH(GETDATE(), 0))

GROUP BY

SalesOfficeID;'''

cursor.execute(previousyear\_query)

avg\_sales\_rows = cursor.fetchall()

avg\_sales\_quantity = pd.DataFrame([list(row) for row in avg\_sales\_rows], columns=['SalesOfficeID', 'AverageSalesQuantity'])

# Merge the average sales quantity column to df1 based on SalesOfficeID

df1 = pd.merge(df1, avg\_sales\_quantity, on='SalesOfficeID', how='left')

df1['sales\_difference'] = df1['total\_sales'] - df1['AverageSalesQuantity']

df1['sales\_percentage\_difference'] = ((df1['sales\_difference'] / df1['AverageSalesQuantity']) \* 100).round(2).astype(str) + '%'

# Execute the second SQL query

query2 = '''SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales, tops

FROM (

SELECT state, city, SalesOfficeID, salesoffice\_Name, total\_sales,

ROW\_NUMBER() OVER (ORDER BY total\_sales ) AS tops

FROM (

SELECT m.state, m.city, s.SalesOfficeID, m.plant\_name AS salesoffice\_Name, cast(SUM(s.salesquantity)as int) AS total\_sales

FROM dw.fSales s

INNER JOIN dw.dSalesOfficeMaster m ON s.SalesOfficeID = m.PLANT

WHERE s.BillingDate = CAST(DATEADD(day, -1, GETDATE()) AS DATE)

GROUP BY m.state, m.city, s.SalesOfficeID, m.plant\_name

) AS ranked\_sales

) AS numbered\_sales

WHERE tops <= 10

ORDER BY total\_sales desc;'''

cursor.execute(query2)

rows2 = cursor.fetchall()

df2 = pd.DataFrame([list(row) for row in rows2], columns=[desc[0] for desc in cursor.description])

cursor.execute(previousyear\_query)

avg\_sales\_rows = cursor.fetchall()

avg\_sales\_quantity = pd.DataFrame([list(row) for row in avg\_sales\_rows], columns=['SalesOfficeID', 'AverageSalesQuantity'])

# Merge the average sales quantity column to df1 based on SalesOfficeID

df2 = pd.merge(df2, avg\_sales\_quantity, on='SalesOfficeID', how='left')

# Calculate the difference between total\_sales and average sales quantity

df2['sales\_difference'] = df2['total\_sales'] - df2['AverageSalesQuantity']

df2['sales\_percentage\_difference'] = ((df2['sales\_difference'] / df2['AverageSalesQuantity']) \* 100).round(2).astype(str) + '%'

cursor.close()

conn.close()

# Specify the output file path for the PDF

output\_file = "combined\_tables.pdf"

custom\_page\_size = (800,800) # Width x Height in points

# Create a PDF document

pdf\_buffer = BytesIO()

pdf = SimpleDocTemplate(pdf\_buffer, pagesize=custom\_page\_size)

# Create styles for paragraphs

styles = getSampleStyleSheet()

title\_style = styles['Heading1']

normal\_style = styles['Normal']

# Define data for tables

data1 = [df1.columns.values.tolist()] + df1.values.tolist()

data2 = [df2.columns.values.tolist()] + df2.values.tolist()

# Add the tables and charts to the PDF content

content = []

image\_path = "C:\\Users\\HP\\Downloads\\heritage.cms"

image = Image(image\_path)

content.append(image)

title\_style = getSampleStyleSheet()["Heading1"]

title\_style.fontSize = 18 # Adjust font size as needed

title\_style.textColor = colors.black # Change color to black or any other color you prefer

# Add the text with the modified style

content.append(Paragraph("HERITAGE FOODs Pvt.Ltd", title\_style))

# Add title and table 1

content.append(Paragraph("Top 10 Sales Report:", title\_style))

table1 = Table(data1)

table1.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

top\_chart\_data = sorted(df1['tops'].tolist(), reverse=True)

top\_category\_names =df1['SalesOfficeID'].astype(str).tolist()

top\_drawing = Drawing(400, 200)

top\_bc = VerticalBarChart()

top\_bc.x = 50

top\_bc.y = 40

top\_bc.height = 150

top\_bc.width = 300

top\_bc.data = [top\_chart\_data]

top\_bc.categoryAxis.labels.angle =25

top\_bc.categoryAxis.labels.fontSize = 12

top\_bc.categoryAxis.categoryNames = top\_category\_names

top\_bc.bars[0].fillColor = colors.green

for i, label in enumerate(top\_category\_names):

top\_bc.bars[i].name = label

top\_drawing.add(top\_bc)

content.append(table1)

content.append(top\_drawing)

content.append(Paragraph("Bottom 10 Sales Report:", title\_style))

table2 = Table(data2)

table2.setStyle(TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

('GRID', (0, 0), (-1, -1), 1, colors.black)]))

bottom\_chart\_data = sorted(df2['tops'].tolist(), reverse=False) # Use total\_sales column for chart data

bottom\_category\_names = sorted(df2['SalesOfficeID'].astype(str).tolist(),reverse=True)

bottom\_drawing = Drawing(400, 200)

bottom\_bc = VerticalBarChart()

bottom\_bc.x = 50

bottom\_bc.y = 40

bottom\_bc.height = 150

bottom\_bc.width = 300

bottom\_bc.data = [bottom\_chart\_data]

bottom\_bc.categoryAxis.labels.angle = 25

bottom\_bc.categoryAxis.labels.fontSize = 12

bottom\_bc.categoryAxis.categoryNames = bottom\_category\_names

bottom\_bc.bars[0].fillColor = colors.blue

# Add the bars to the drawing

bottom\_drawing.add(bottom\_bc)

content.append(table2)

content.append(bottom\_drawing)

pdf.build(content)

# Rewind the buffer

pdf\_buffer.seek(0)

# Send email with PDF attachment

receiver\_email = 'vnallagari1919@gmail.com' # Update with recipient's email address

send\_email\_with\_pdf\_attachment(pdf\_buffer, receiver\_email)

print(f"Data exported to PDF and sent via email to {receiver\_email}")