

SPRINT – 2

Data	05/10/2022
Team ID	PNT2022TMID32597
Project Name	Personal Expense Tracker

DATABASE ;

```
from __future__ import print_function
from datetime import datetime
from flask import Flask, request, json, jsonify
from flask_json import FlaskJSON, json_response
from flask_cors import CORS
import ibm_db
from template import *

# Initializing flask app
app = Flask(__name__)
jsonObj = FlaskJSON(app)
cors = CORS(app,resources={r'*':{'origins':'http://localhost:3000'}}})

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=125f9f61-9715-46f9-9399-
c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30426;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=mpr37327;PWD=k5iRh2q3KBmMzr7Z;","","")

# Route for seeing a data
@app.route('/login')
def login():
    email = request.args.get('email')
    password = request.args.get('password')
    sql = "SELECT * FROM login where email = '{}'.format(email)"
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    if document == False :
        response = json_response(value=0)
    elif document['PASSWORD'] == password :
        response = json_response(value=1)
    else :
        response = json_response(value=2)
    return response

@app.route('/register',methods=['POST'])
```

```

def register():
    if request.method == "POST":
        credentials = json.loads(request.data)
        sql = "INSERT INTO login
VALUES('{}','{}').format(credentials['email'],credentials['password'])
        out = ibm_db.exec_immediate(conn, sql)
        sql = "INSERT INTO personal_info(email,name)
VALUES('{}','{}').format(credentials['email'],credentials['name'])
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response

@app.route('/loadData')
def loadData():
    email = request.args.get('email')
    sql = "select sum(amount) as expense from expenses where email='{}' and
month(timestamp)=month(current_timestamp)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)

    totalExpense = document['EXPENSE']
    print(totalExpense)
    resultData = {
        'totalExpense' : document['EXPENSE'],
    }
    sql = "select walletlimit from personal_info where email =
'{}'.format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)

    # if resultData['totalExpense'] == None:
    #     resultData['balance'] = 0
    # else:
    #     resultData['balance'] = document['WALLETLIMIT'] - totalExpense
    sql = "select category, sum(amount) as expense from expenses where email='{}'
and month(timestamp)=month(current_timestamp) group by category".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    piegraphData = []
    piegraphLabel = []
    while document != False:
        piegraphLabel.append(document["CATEGORY"])
        piegraphData.append(document["EXPENSE"])
        document = ibm_db.fetch_assoc(out)
    resultData['piegraphdata'] = piegraphData

```

```

    resultData['piegraphlabel'] = piegraphLabel
    sql = "select dayname(cast(timestamp as date)) as day, sum(amount) as expense
from expenses,sysibm.sysdummy1 where email='{}' and
week(timestamp)=week(current_timestamp) group by cast(timestamp as
date)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    bargraphData = []
    bargraphLabel =[]
    while document != False:
        bargraphLabel.append(document['DAY'])
        bargraphData.append(document["EXPENSE"])
        document = ibm_db.fetch_assoc(out)
    resultData['bargraphdata'] = bargraphData
    resultData['bargraphlabel'] = bargraphLabel
    sql = "select sum(amount) as expense from expenses where email='{}' and
date(timestamp)=date(current_timestamp)".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    resultData['dailyExpense']=document['EXPENSE']
    response = json_response(resultData=resultData)
    return response

@app.route('/addExpense', methods=['POST'])
def addExpense():
    if request.method == "POST":
        expense = json.loads(request.data)
        sql = "INSERT INTO expenses(email,category,amount,timestamp)
VALUES('{}','{}',{},{})".format(expense['email'],expense['category'],expense['a
mount'], datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response

# @app.route('/limitExceed')
# def limitExceed():
#     email = request.args.get('email')
#     SendDynamic()
#     return json_response(200)

@app.route('/personalData')
def personalData():
    email = request.args.get('email')
    sql = "select * from personal_info where email='{}'".format(email)
    out = ibm_db.exec_immediate(conn, sql)

```

```

document = ibm_db.fetch_assoc(out)
resultData = {
    'name' : document['NAME'],
    'email' : email,
    'walletlimit':document['WALLETLIMIT'],
    'gender': document['GENDER'],
    'location': document['LOCATION'],
    'phone' : document['PHONE'],

}
sql = "select * from login where email='{}'".format(email)
out = ibm_db.exec_immediate(conn, sql)
document = ibm_db.fetch_assoc(out)
resultData['password'] = document['PASSWORD']
response = json_response(resultData=resultData)
return response

@app.route('/updateProfile',methods=['POST'])
def updateProfile():
    if request.method == "POST":
        credentials = json.loads(request.data)
        sql = "UPDATE login SET password='{}' where
email='{}'".format(credentials['password'],credentials['email'])
        out = ibm_db.exec_immediate(conn, sql)
        print("Cred:",credentials)
        sql = "UPDATE personal_info SET name='{}', walletlimit={}, gender='{}',
location='{}', phone='{}' where email='{}'
".format(credentials['name'],credentials['walletlimit'],credentials['gender'],cre
dentials['location'],credentials['phone'], credentials['email'])
        out = ibm_db.exec_immediate(conn, sql)
        response = json_response(200)
        return response
    print("hai out")

# Running app
if __name__ == '__main__':
    app.run(debug=True)

```

DATABASE SCREENSHOTS :

1 . LOGIN TABLE

The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is selected. A search bar at the top left contains the text 'Find schemas or tables'. On the left sidebar, the 'Schemas' section is expanded, showing a list of tables: EXPENSES, LOGIN, and PERSONAL_INFO, all under the schema MPR37327. The 'LOGIN' table is selected. The main panel displays the 'Table definition' for the 'LOGIN' table. It shows the table has approximately 5 rows (32.0 KB) and was updated on 2022-11-17 12:32:47. The table definition table is as follows:

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	Y	32	0
PASSWORD	VARCHAR	Y	32	0

A 'View data' button is located at the bottom right of the table definition panel.

2 . EXPENSES TABLE

The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is selected. A search bar at the top left contains the text 'Find schemas or tables'. On the left sidebar, the 'Schemas' section is expanded, showing a list of tables: EXPENSES, LOGIN, and PERSONAL_INFO, all under the schema MPR37327. The 'EXPENSES' table is selected. The main panel displays the 'Table definition' for the 'EXPENSES' table. It shows the table has approximately 5 rows (32.0 KB) and was updated on 2022-11-17 10:57:29. The table definition table is as follows:

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	Y	32	0
CATEGORY	VARCHAR	Y	32	0
AMOUNT	INTEGER	Y		0
TIMESTAMP	VARCHAR	Y	32	0

A 'View data' button is located at the bottom right of the table definition panel.

3 . PERSONAL INFO TABLE

Load Data

Load History

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

Find schemas or tables

Refresh

SQL

Business

Tables

New table

Filter

Sort

Columns

Close

Table definition

PERSONAL_INFO

Approximate 1 rows (32.0 KB)

Updated on 2022-11-17 10:57:28

Name

Schema

Properties

EXPENSES

MPR37327

...

LOGIN

MPR37327

...

PERSONAL_INFO

MPR37327

...

Total: 3, selected: 0

Name

Data type

Nullable

Length

Scale

NAME

VARCHAR

Y

32

0

EMAIL

VARCHAR

Y

32

0

WALLETLIMIT

DECIMAL

Y

5

0

GENDER

VARCHAR

Y

32

0

LOCATION

VARCHAR

Y

32

0

PHONE

VARCHAR

Y

32

0

View data