**ENSF607 Assignment 3**

Balkarn Gill - 30202219 - balkarn.gill1@ucalgary.ca

Yajur Vashist - 30200252  
Satchytan Karalasingham - 30222555

Momin Muhammad - 30033100

SQL Scripts for table

**createdaatabase.sql:**

CREATE DATABASE ServiceTickets;

USE ServiceTickets;

CREATE TABLE EventActivity (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Activityname VARCHAR(20)

);

CREATE TABLE EventOrigin (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Origin VARCHAR(20)

);

CREATE TABLE EventStatus (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Status VARCHAR(20)

);

CREATE TABLE EventClass (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Class VARCHAR(20)

);

CREATE TABLE EventLog (

ID INT AUTO\_INCREMENT PRIMARY KEY,

CaseID VARCHAR(20) UNIQUE,

Activity VARCHAR(20),

Urgency VARCHAR(1),

Impact VARCHAR(1),

Priority VARCHAR(1),

StartDate DATE,

EndDate DATE,

TicketStatus VARCHAR(20),

UpdateDateTime DATETIME,

Duration INT,

Origin VARCHAR(20),

Class VARCHAR(20)

);

**authentication.sql**

CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';

GRANT ALL PRIVILEGES ON servicetickets.\* TO 'myuser'@'localhost';

FLUSH PRIVILEGES;

Source code for generator program

**generator.py:**

import mysql.connector

from mysql.connector import Error

import random

from datetime import datetime, timedelta

*# Sample data from your database tables*

event\_activities = ['Design', 'Construction', 'Test', 'Password Reset']

event\_origins = ['Joe S.', 'Bill B.', 'George E.', 'Achmed M.', 'Rona E.']

event\_statuses = ['Open', 'On Hold', 'In Process', 'Deployed', 'Deployed Failed']

event\_classes = ['Change', 'Incident', 'Problem', 'SR']

*# Input parameters*

num\_tickets = 100

time\_window\_start = datetime(2023, 1, 1)

time\_window\_end = datetime(2023, 6, 30)

unique\_random\_integers = random.sample(range(1, num\_tickets + 1), num\_tickets)

i = 0

*# Function to generate a random datetime within the specified time window*

def random\_datetime(start, end):

return start + timedelta(

seconds=random.randint(0, int((end - start).total\_seconds()))

)

*# Function to calculate priority from urgency and impact*

def calculate\_priority(urgency, impact):

if urgency == 'H' and impact == 'H':

return 'H'

elif urgency == 'H' and impact == 'M':

return 'M'

elif urgency == 'M' and impact == 'H':

return 'M'

elif urgency == 'H' and impact == 'L':

return 'M'

elif urgency == 'L' and impact == 'H':

return 'M'

else:

return 'L'

*# Function to insert randomly generated ticket into SQL database*

def insert\_ticket(case\_id, activity, origin, status, ticket\_class, start\_date, end\_date, urgency, impact, priority, duration, update\_datetime):

*# Define SQL statements for each table*

sql\_event\_log = "INSERT INTO EventLog (Caseid, Activity, Origin, TicketStatus, Class, StartDate, EndDate, Urgency, Impact, Priority, Duration, UpdateDateTime) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

sql\_event\_activity = "INSERT INTO EventActivity (Activityname) VALUES (%s)"

sql\_event\_origin = "INSERT INTO EventOrigin (Origin) VALUES (%s)"

sql\_event\_status = "INSERT INTO EventStatus (Status) VALUES (%s)"

sql\_event\_class = "INSERT INTO EventClass (Class) VALUES (%s)"

values\_event\_log = (case\_id, activity, origin, status, ticket\_class, start\_date, end\_date, urgency, impact, priority, duration, update\_datetime)

values\_event\_activity = (activity,)

values\_event\_origin = (origin,)

values\_event\_status = (status,)

values\_event\_class = (ticket\_class,)

*# Start a transaction*

connection.start\_transaction()

try:

*# Insert into EventLog*

cursor.execute(sql\_event\_log, values\_event\_log)

*# Insert into EventActivity*

cursor.execute(sql\_event\_activity, values\_event\_activity)

*# Insert into EventOrigin*

cursor.execute(sql\_event\_origin, values\_event\_origin)

*# Insert into EventStatus*

cursor.execute(sql\_event\_status, values\_event\_status)

*# Insert into EventClass*

cursor.execute(sql\_event\_class, values\_event\_class)

*# Commit the transaction*

connection.commit()

except Exception as e:

*# Rollback the transaction if an error occurs*

print("Error:", e)

connection.rollback()

connection.commit()

db\_config = {

'host': 'localhost',

'user': 'myuser',

'password': 'mypassword',

'database': 'ServiceTickets'

}

*# Establish a connection to the MySQL database*

connection = mysql.connector.connect(\*\*db\_config)

cursor = connection.cursor()

*# Generate and print tickets*

for \_ in range(num\_tickets):

activity = random.choice(event\_activities)

origin = random.choice(event\_origins)

status = random.choice(event\_statuses)

event\_class = random.choice(event\_classes)

start\_date = random\_datetime(time\_window\_start, time\_window\_end)

end\_date = random\_datetime(start\_date, time\_window\_end)

update\_datetime = random\_datetime(start\_date, end\_date)

*# Calculate urgency, impact, and priority (you can define your own logic here)*

urgency = random.choice(['H', 'M', 'L']) *# High, Medium, Low*

impact = random.choice(['H', 'M', 'L']) *# High, Medium, Low*

priority = calculate\_priority(urgency, impact)

id = unique\_random\_integers[i]

i += 1

case\_id = f'CS\_{id:05d}' *# Unique Case ID*

*# case\_id = f'CS\_{random.randint(1, 99999):05d}' # Unique Case ID*

*# Print or save the generated ticket data*

insert\_ticket(case\_id, activity, origin, status, event\_class, start\_date, end\_date, urgency, impact, priority, int((end\_date - start\_date).total\_seconds()), update\_datetime)

cursor.close()

connection.close()