---------- Python code------------

```python
import psycopg2
import pandas as pd
import sys
from datetime import datetime
import os
BASE = os.path.dirname(os.path.abspath(__file__))


def path(filename):
    return os.path.join(BASE, filename)



def getConn():
    pwFile = open("pw.txt", "r")
    pw = pwFile.read()
    pwFile.close()


    connStr = "host='cmpstudb-01.cmp.uea.ac.uk' dbname='cxh25psu' user='cxh25psu' password = " + pw
    conn=psycopg2.connect(connStr)
    return  conn

def clearOutput():
    with open(path("output.txt"), "w") as clearfile:
        clearfile.write('')



def writeOutput(output):
    with open(path("output.txt"), "a") as myfile:
        myfile.write(output + "\n")




def handle_transaction(cmd, data, conn, cur):
    output = f"TASK {cmd}: Processing with data {data}"
    writeOutput(output)

    try:

        if cmd == 'A':
            sql = "INSERT INTO spectator (sno, sname, semail) VALUES (%s, %s, %s);"
            cur.execute(sql, (data[0], data[1], data[2]))
            writeOutput(f"SUCCESS: Spectator {data[0]} inserted.")


        elif cmd == 'B':
            sql = "INSERT INTO event (ecode, edesc, elocation, edate, etime, emax) VALUES (%s, %s, %s, %s, %s, %s);"
            cur.execute(sql, (data[0], data[1], data[2], data[3], data[4], data[5]))
            writeOutput(f"SUCCESS: Event {data[0]} inserted.")

        elif cmd == 'C':
            sql = "DELETE FROM spectator WHERE sno = %s;"
            cur.execute(sql, (data[0],))
            writeOutput(f"SUCCESS: Spectator {data[0]} deleted (Tickets automatically cancelled and moved to audit log).")


        elif cmd == 'D':
            sql = "DELETE FROM event WHERE ecode = %s;"
            cur.execute(sql, (data[0],))
            writeOutput(f"SUCCESS: Event {data[0]} deleted (Tickets automatically cancelled and moved to audit log).")


        elif cmd == 'E':

            cur.execute("SELECT COALESCE(MAX(tno), 0) FROM ticket;")
            next_tno = cur.fetchone()[0] + 1


            sql = "INSERT INTO ticket (tno, ecode, sno) VALUES (%s, %s, %s);"
            cur.execute(sql, (next_tno, data[0], data[1]))
            writeOutput(f"SUCCESS: Ticket {next_tno} issued for event {data[0]}.")


        elif cmd == 'F':
            sql = """
                SELECT e.edate, e.elocation, COUNT(DISTINCT t.sno) AS total_spectators
                FROM event e JOIN ticket t ON e.ecode = t.ecode
                GROUP BY e.edate, e.elocation ORDER BY e.edate, e.elocation;
            """
            table_df=pd.read_sql_query(sql, conn)
            writeOutput("Report F: Spectators Liable to Travel:")
            writeOutput(table_df.to_string())


        elif cmd == 'G':
            sql = """
                SELECT e.edesc, e.ecode, COUNT(t.tno) AS total_tickets_issued
                FROM event e LEFT JOIN ticket t ON e.ecode = t.ecode
                GROUP BY e.edesc, e.ecode
                ORDER BY e.edesc;
            """
            table_df=pd.read_sql_query(sql, conn)
            writeOutput("Report G: Total Tickets Issued Per Event:")
            writeOutput(table_df.to_string())


        elif cmd == 'H':
            ecode = data[0]
            sql = "SELECT COUNT(tno) AS tickets_for_event FROM ticket WHERE ecode = %s;"
            cur.execute(sql, (ecode,))
            count = cur.fetchone()[0]
            writeOutput(f"Report H: Total active tickets for {ecode}: {count}")


        elif cmd == 'I':
            sno = data[0]

            sql = "SELECT sname, edate, elocation, etime, edesc FROM spectator_event_schedule WHERE sno = %s ORDER BY edate, etime;"
            table_df=pd.read_sql_query(sql, conn, params=(sno,))
            writeOutput(f"Report I: Schedule for Spectator {sno} (Active Tickets):")
            writeOutput(table_df.to_string())


        elif cmd == 'J':
            tno = data[0]
            sql = """
                SELECT t.tno, s.sname, t.ecode, 'Valid' AS status
                FROM ticket t JOIN spectator s ON t.sno = s.sno WHERE t.tno = %s
                UNION ALL
                SELECT c.tno, COALESCE(s.sname, 'Spectator Deleted (sno: ' || c.sno || ')'),
                       c.ecode, 'Cancelled' AS status
                FROM cancel c LEFT JOIN spectator s ON c.sno = s.sno WHERE c.tno = %s;
            """
            table_df=pd.read_sql_query(sql, conn, params=(tno, tno))
            writeOutput(f"Report J: Details for Ticket {tno}:")
            if table_df.empty:
                 writeOutput("Ticket not found (neither active nor cancelled).")
            else:
                 writeOutput(table_df.to_string())


        elif cmd == 'K':
            ecode = data[0]
            sql = "SELECT tno, sno, cdate, cuser FROM cancel WHERE ecode = %s;"
            table_df=pd.read_sql_query(sql, conn, params=(ecode,))
            writeOutput(f"Report K: Cancelled tickets for {ecode} (Audit Log):")
            writeOutput(table_df.to_string())


        elif cmd == 'L':
            cur.execute("DELETE FROM cancel;")
            cur.execute("DELETE FROM ticket;")
            cur.execute("DELETE FROM spectator;")
            cur.execute("DELETE FROM event;")
            writeOutput("SUCCESS: Database tables emptied.")

        conn.commit()

    except psycopg2.IntegrityError as e:
        conn.rollback()

        writeOutput(f"FAILURE: Integrity Error. Constraint Violated. Details: {e.pgerror.strip()}")
        writeOutput("---")
    except Exception as e:
        conn.rollback()
        writeOutput(f"FAILURE: General Application Error. Details: {e}")
        writeOutput("---")



try:
    conn=None
    conn=getConn()
    conn.autocommit=True
    cur = conn.cursor()

    cur.execute('SET search_path to Assessment_100536383')


    f = open("input.txt", "r")
    clearOutput()

    for x in f:
        line = x.strip()
        if not line:
            continue

        if(line[0] == 'X'):
            writeOutput("\nExit program!")
            break

        raw = line.split("#",1)
        cmd = raw[0].strip()

        if len(raw) > 1:
            raw[1]=raw[1].strip()
            data = raw[1].split("#")
        else:
            data = []

        handle_transaction(cmd, data, conn, cur)

except Exception as e:
    writeOutput(f"Critical System Error: {e}")
finally:
    if conn:
        conn.close()
```