**CMP-7025A Database Manipulation**

**Reg No:**

100443924

**Marker:**

***Please follow the instructions below to complete the assessment template.***

**Part 1:**  For the first section, insert the SQL (DDL) statements with constraints, triggers, functions (if any function needs to be used by multiple transactions of interest, put that in Part 1), views, ER diagrams, and your own data.

**Part 2:**  Before executing the SQL (DML) commands for transactions of interest in this section, **ensure to delete all existing data** and populate the tables with the assessment data provided in the script (Assessment\_data.txt).

For individual SQL queries where output is expected (shown in red), insert the SQL statement you executed and present the output of running the statement from the PgAdmin (use screenshots).

**Notes:**

1. If a task involves using transaction-specific functions, present the function first and then call it.
2. Insert SQL statements in the order they were executed to accomplish the task.
3. Use screenshots from PgAdmin for tasks, ensuring readable text and visible table rows.
4. Capture any errors reported by the SQL environment (use screenshots).

If any SQL command encounters an error, record and paste the error message using a screenshot with the error shown. When a task is not implemented, use 'NOT DONE' as the output. Please refrain from removing the allocated marks (in blue) for each section, as they aid in assessment.

Refer to the appendix at the end of the document for an example on how to fill out the template for a transaction of interest.

**Part 3:**  Execute the Python program using the data from the input file (input.txt). Paste the output file (output.txt) resulting from running the program with input.txt in the "Python Program Outputs" section on this document.

***Submit all files as instructed in the specified format (Sample Submission Folder-Directory and File Structure), along with this document, in the appropriate directory.***

**Part 1: Database definition and loading**

Insert your solution for part 1

--Create event table

CREATE TABLE event (

ecode CHAR (4) PRIMARY KEY,

edesc VARCHAR (20) NOT NULL,

elocation VARCHAR (20) NOT NULL,

edate DATE NOT NULL CHECK (edate >= '2024-07-01' AND edate <= '2024-07-31'),

etime TIME NOT NULL CHECK (etime >= '09:00:00'),

emax SMALLINT NOT NULL CHECK (emax >= 1 AND emax <= 1000)

);

SELECT \* FROM event;

A screenshot of a computer

Description automatically generated

--Create spectator table

CREATE TABLE spectator (

sno INTEGER PRIMARY KEY,

sname VARCHAR (20) NOT NULL,

semail VARCHAR (20) NOT NULL

);

SELECT \* FROM spectator;

A screenshot of a computer

Description automatically generated

--Create ticket table

CREATE TABLE ticket (

tno INTEGER PRIMARY KEY,

ecode CHAR (4) NOT NULL,

sno INTEGER NOT NULL,

FOREIGN KEY (ecode) REFERENCES event(ecode),

FOREIGN KEY (sno) REFERENCES spectator(sno),

CHECK (tno >= 1)

);

SELECT \* FROM ticket;

A screenshot of a computer

Description automatically generated

--Create cancel table

CREATE TABLE cancel (

tno INTEGER PRIMARY KEY,

ecode CHAR (4) NOT NULL,

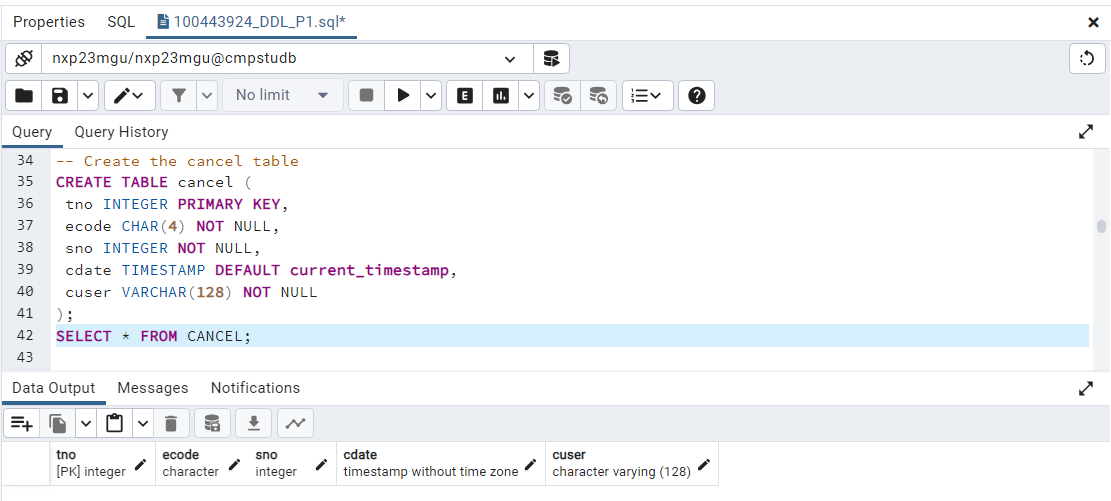
sno INTEGER NOT NULL,

cdate TIMESTAMP DEFAULT current\_timestamp,

cuser VARCHAR (128) NOT NULL

);

SELECT \* FROM CANCEL;



--Create Trigger Function – if event date, event location and event time change means update or cancel

1. Create function

CREATE OR REPLACE FUNCTION event\_cancel\_ticket()

RETURNS trigger

LANGUAGE 'plpgsql'

AS

$$

BEGIN

INSERT INTO cancel (tno, ecode, sno, cdate, cuser)

SELECT tno, ecode, sno, CURRENT\_DATE, 'Management1'

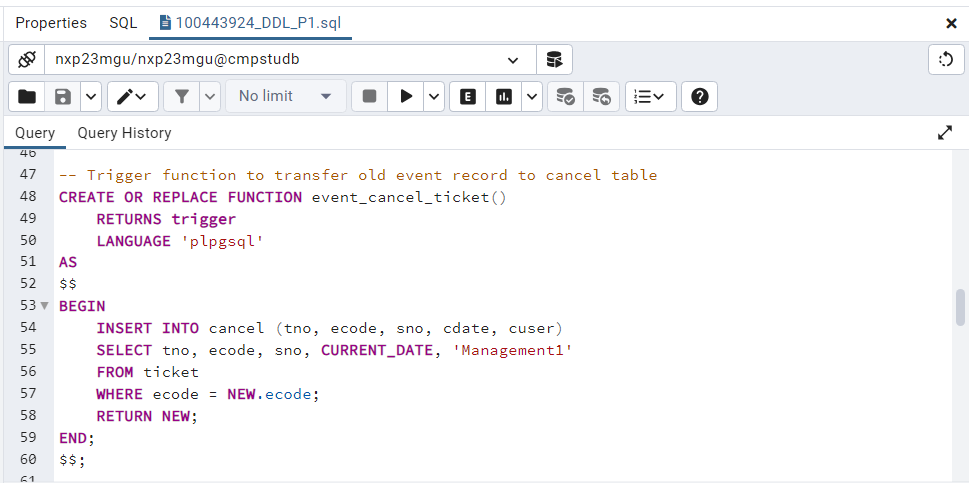
FROM ticket

WHERE ecode = NEW.ecode;

RETURN NEW;

END;

$$;



2. Create Trigger

CREATE TRIGGER trigger\_event\_update

AFTER UPDATE OF elocation, edate, etime

ON event

FOR EACH ROW

EXECUTE FUNCTION event\_cancel\_ticket();

A screenshot of a computer

Description automatically generated

3. Update Event description in event table

update event set elocation='USA' where ecode='P001'

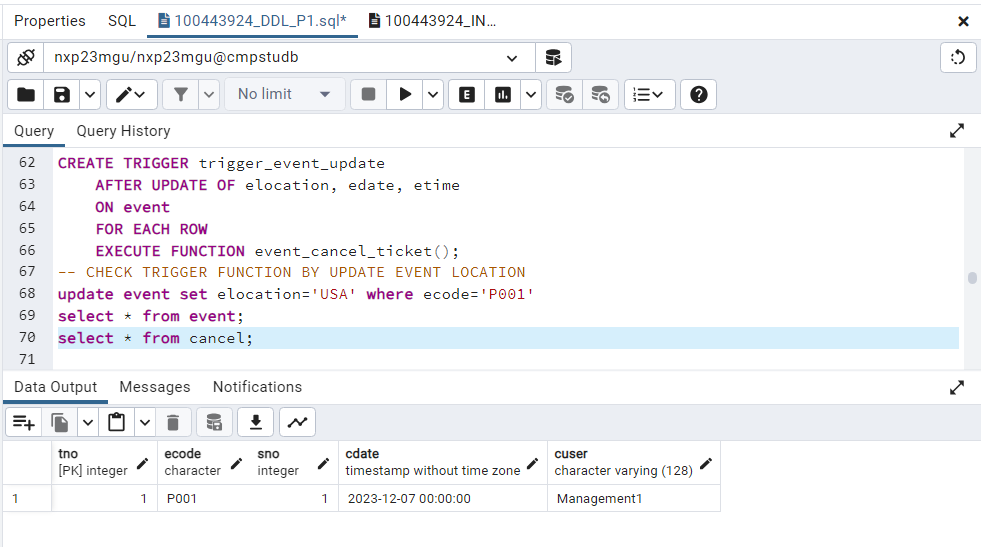
select \* from event;

A screenshot of a computer

Description automatically generated

4.After Update event location, old record of event transferred to cancel table

select \* from cancel;



--Create View Table to Check the cancel ticket details.

1. View function codes

CREATE OR REPLACE FUNCTION event\_cancellation\_details\_view()

RETURNS TABLE (

ticket\_number INTEGER,

spectator\_name VARCHAR (20),

event\_code CHAR (4),

event\_description VARCHAR (20),

event\_location VARCHAR (20),

event\_date DATE,

event\_time TIME,

cancellation\_date TIMESTAMP,

cancellation\_user VARCHAR (128)

) AS $$

A screenshot of a computer

Description automatically generated

BEGIN

RETURN QUERY

SELECT

t.tno AS ticket\_number,

s.sname AS spectator\_name,

e.ecode AS event\_code,

e.edesc AS event\_description,

e.elocation AS event\_location,

e.edate AS event\_date,

e.etime AS event\_time,

c.cdate AS cancellation\_date,

c.cuser AS cancellation\_user

FROM

ticket t

JOIN spectator s ON t.sno = s.sno

JOIN event e ON t.ecode = e.ecode

JOIN cancel c ON t.tno = c.tno;

END;

$$ LANGUAGE plpgsql;

A screenshot of a computer

Description automatically generated

1. View function output

CREATE OR REPLACE VIEW event\_cancellation AS

SELECT \* FROM event\_cancellation\_details\_view();

SELECT \* from event\_cancellation;

A screenshot of a computer

Description automatically generated

--ER Diagram

A screenshot of a computer

Description automatically generated

--Train data

1. Insert Event data with Output

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P001','CANDLE\_LIGHT','THE\_HALL','2024-07-03','09:30:00',200);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P002','MUSICAL\_NIGHT','ASSEMBLY\_HALL','2024-07-09','14:00:00',80);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P003','WINTER\_GAMES','LONG\_WATER','2024-07-14','10:0:00',800);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P004','SUMMER\_GAMES','LONDON','2024-07-18','11:0:00',500);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P005','SUMMER\_GAMES','LONG\_WATER','2024-07-05','13:0:00',250);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P006','CANDLE\_LIGHT','ASSEMBLY\_HALL','2024-07-06','16:0:00',150);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P007','MUSICAL\_NIGHT','THE\_HALL','2024-07-22','13:00:00',650);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P008','LIVE\_CONCERT','CITY\_CENTER','2024-07-25','17:00:00',250);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P009','TRSUREHUNT','PARK','2024-07-28','10:00:00',50);

INSERT INTO event (ecode, edesc, elocation, edate, etime, emax)

VALUES ('P010','QUIZ','CASTLE\_MDW','2024-07-11','12:00:00',170);

A screenshot of a computer

Description automatically generated

2.Insert spectator data with output

INSERT INTO spectator (sno, sname, semail)

VALUES (1,'ANDREW','andrew\_s1@gmail.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (2,'STEVE','steve\_s@outlook.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (3,'FLINTOFF','flintoff.1@yahoo.uk');

INSERT INTO spectator (sno, sname, semail)

VALUES (4,'CLEARK','cleark@outlook.ac.uk');

INSERT INTO spectator (sno, sname, semail)

VALUES (5,'JOHN','john\_32@gmail.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (6,'COOPER','copper\_s12@yahoo.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (7,'STIFEN','stifen\_h@gmail.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (8,'Georgey','georgey@hotmail.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (9,'CANNEY','caneey\_321@yahoo.com');

INSERT INTO spectator (sno, sname, semail)

VALUES (10,'PATRIC','pratick@outlook.com');

A screenshot of a computer

Description automatically generated

3.Insert ticket data with output

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P001', 1);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P002', 2);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P003', 3);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P004', 4);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P005', 5);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P006', 6);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P007', 7);

INSERT INTO ticket VALUES

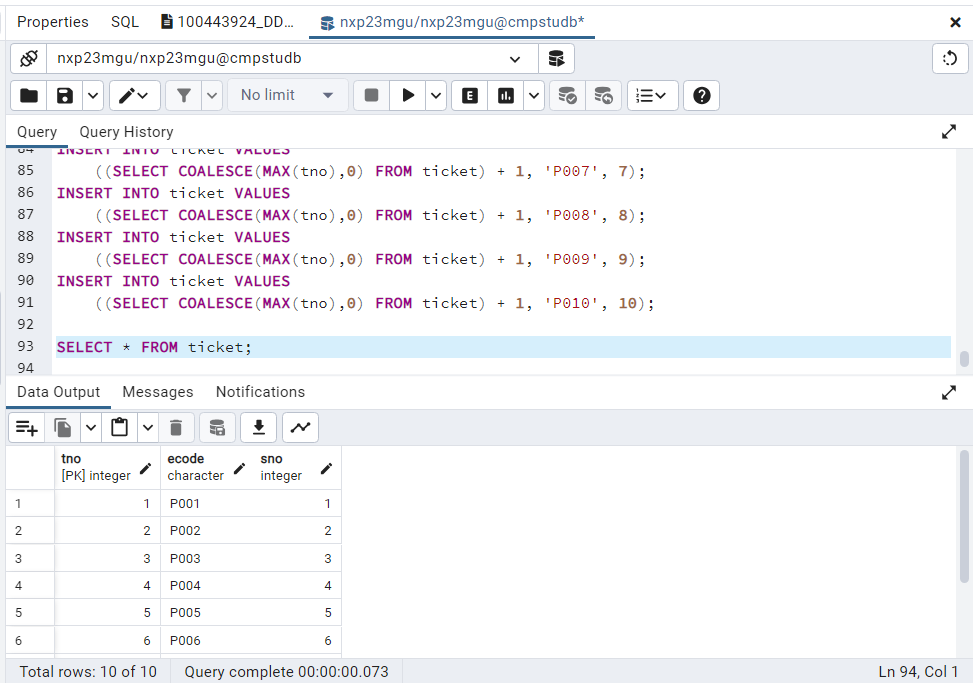
((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P008', 8);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P009', 9);

INSERT INTO ticket VALUES

((SELECT COALESCE(MAX(tno),0) FROM ticket) + 1, 'P010', 10);



Primary Keys /4

Check constraints /7

Foreign Keys /6

Triggers/functions /4

Others (Views, Indexes, ER diagram) /4

Train data /1

T0TAL /26

Marker’s Comments (If needed):

**Part 2. Interactive SQL version of the transactions**

*For each of the testing below, please ensure to provide adequate evidence demonstrating the completion of the testing. For instance, if you are performing an insert statement, display the relevant table (using a SELECT statement) after executing the insert operation.* ***Please execute the Testing Tasks sequentially, starting with (1) before proceeding to (2), and so forth.***

--------------------------------------------------------------------------------------------------------------------------------------

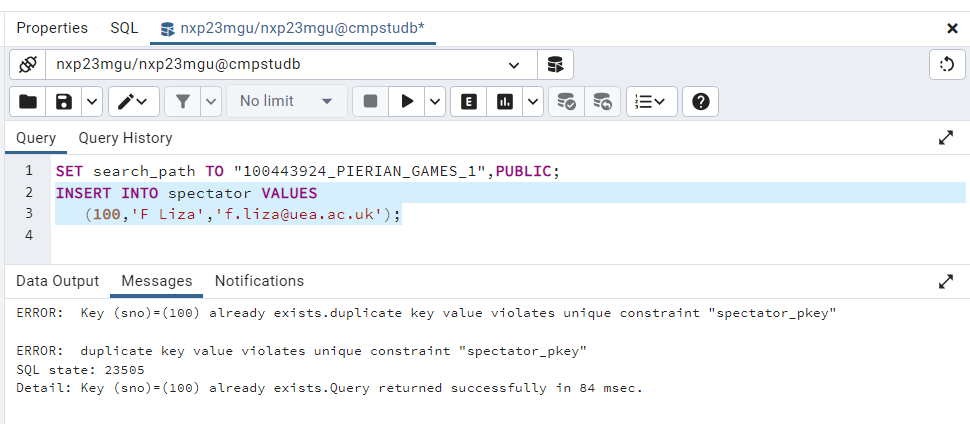
1. **Testing task A**

--------------------------------------------------------------------------------------------------------------------------------------

Insert a new spectator.

**sno = 100, sname = ‘F Liza’, semail = ‘**[**f.liza@uea.ac.uk**](mailto:f.liza@uea.ac.uk)**’**

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

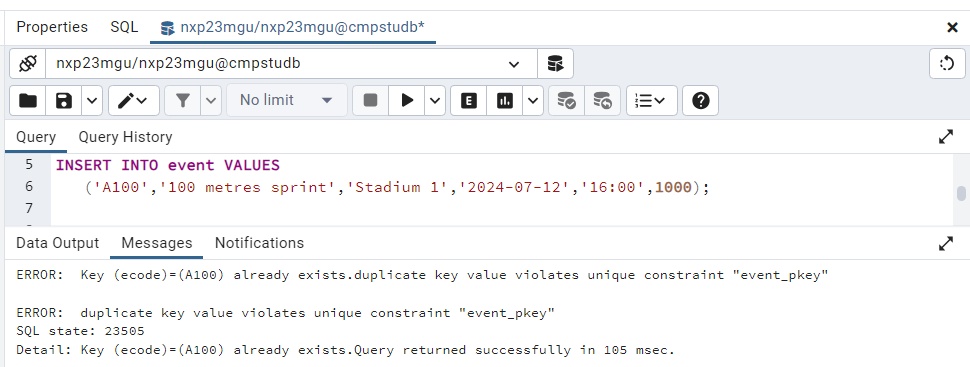
1. **Testing task B (i)**

--------------------------------------------------------------------------------------------------------------------------------------

Insert a new event.

ecode = 'A100', edesc = '100 metres sprint', elocation ='Stadium 1', edate = '2024-07-12', etime='16:00', emax = 1000

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

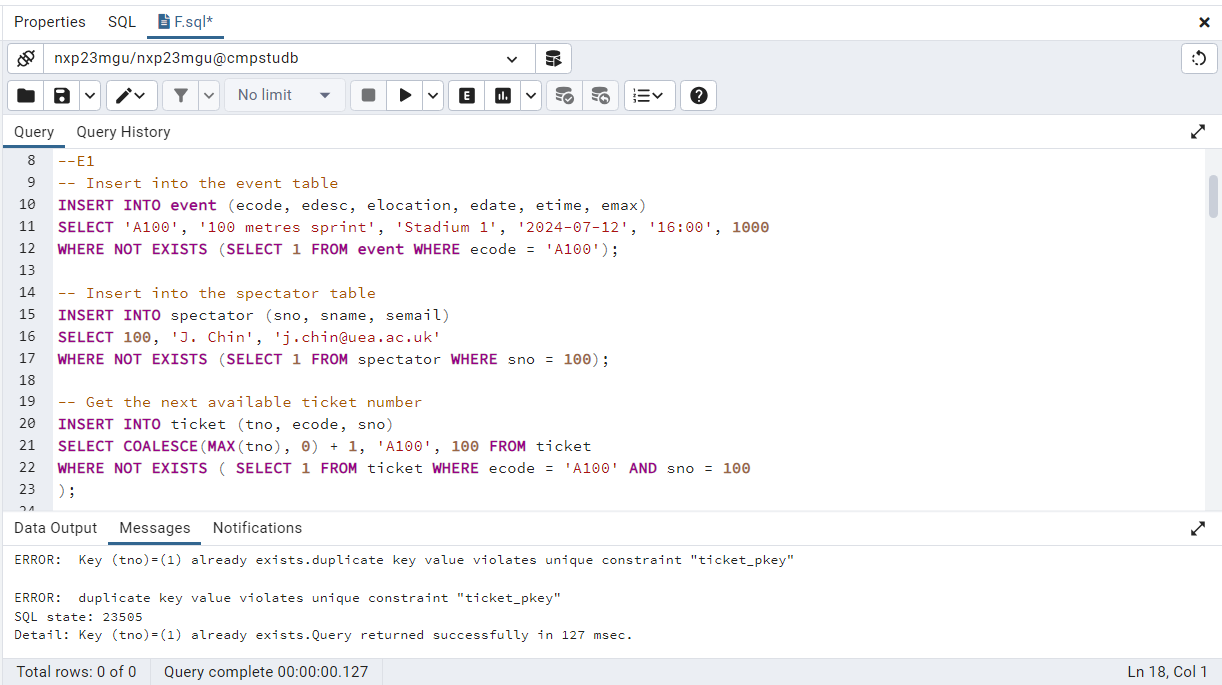
1. **Testing task E (i)**

--------------------------------------------------------------------------------------------------------------------------------------

Issue a ticket for an event.

ecode = 'A100' , sno = 100

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

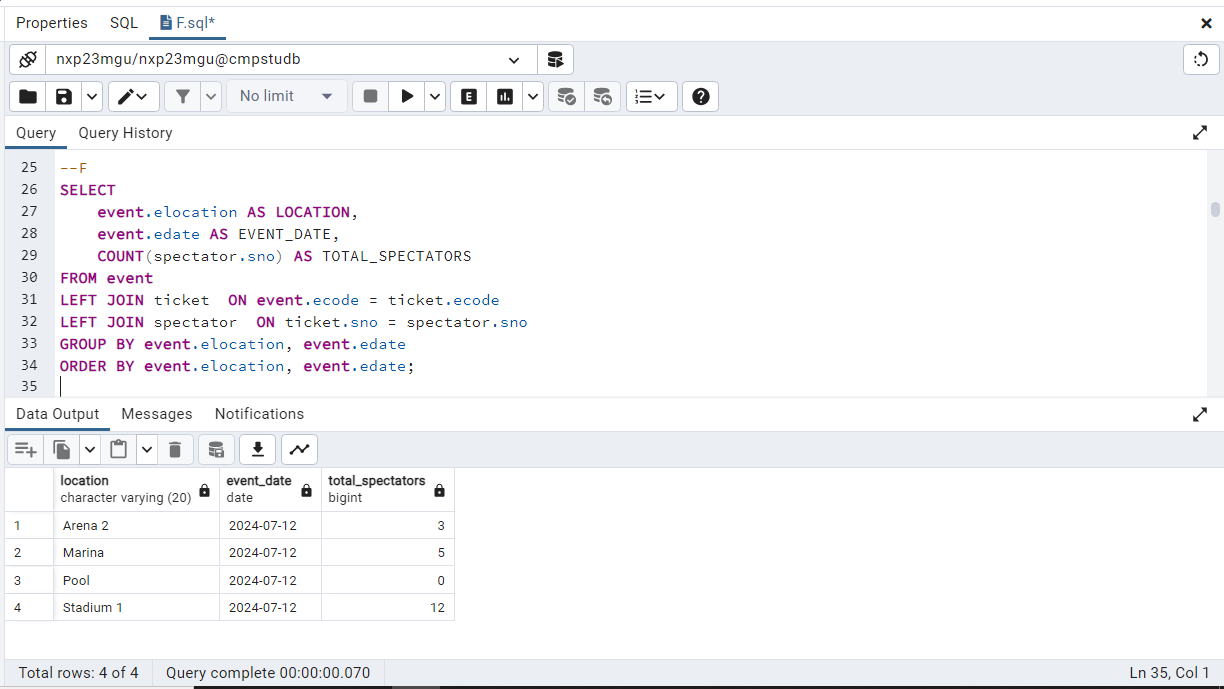
--------------------------------------------------------------------------------------------------------------------------------------

1. **Testing task F**

--------------------------------------------------------------------------------------------------------------------------------------

Produce a report showing the total number of spectators liable to travel to a location. The table should show the total number of spectators that could travel to a location on each date an event is held at a location.

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /3

Marker’s Comments (If needed):

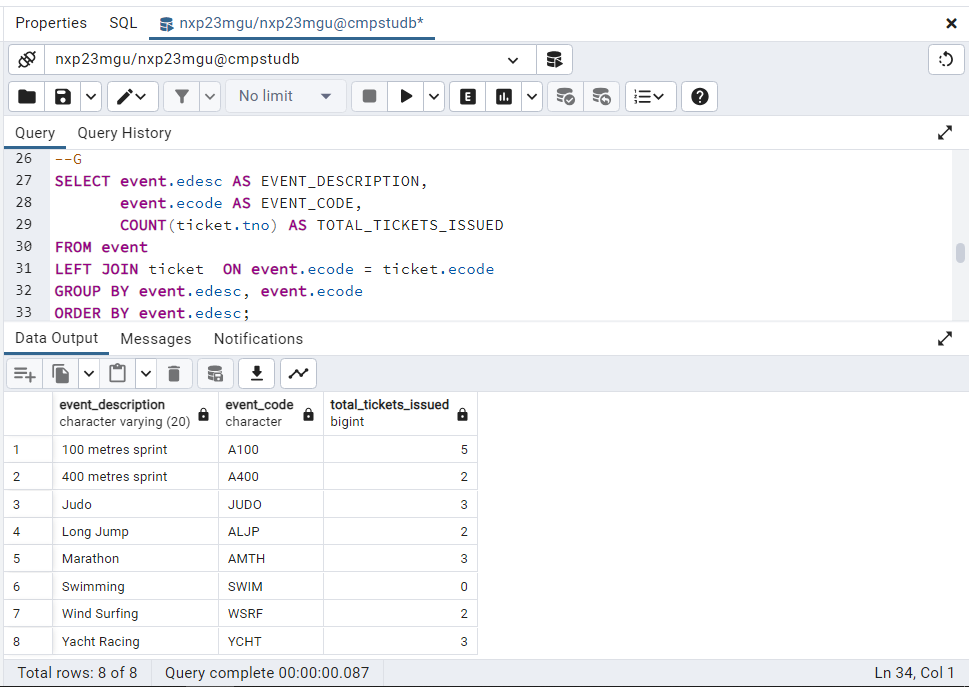
--------------------------------------------------------------------------------------------------------------------------------------

1. **Testing task G**

--------------------------------------------------------------------------------------------------------------------------------------

Produce a report showing the total number of tickets issued for each event. Present the data in event description sequence.

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /5

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

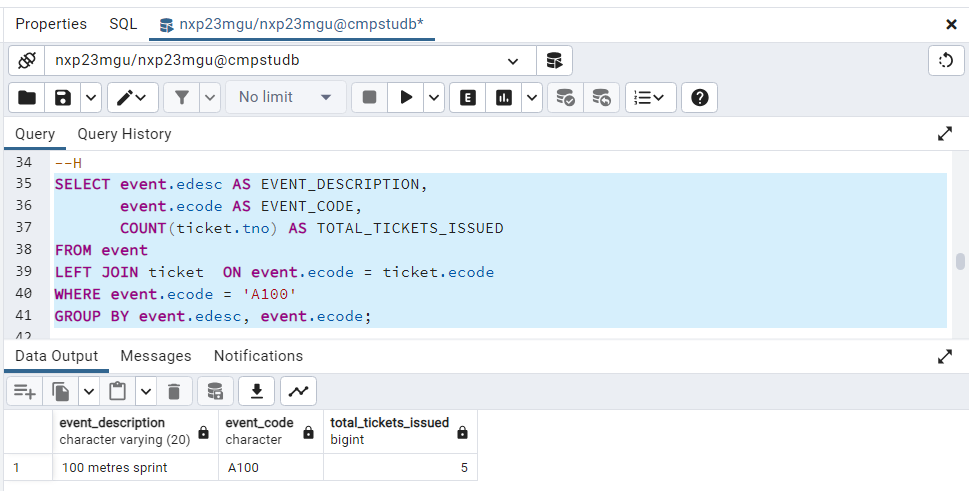
**(6) Testing task H**

--------------------------------------------------------------------------------------------------------------------------------------

As task **G** but only for a given event which is specified by the event code.

ecode = 'A100'

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /3

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

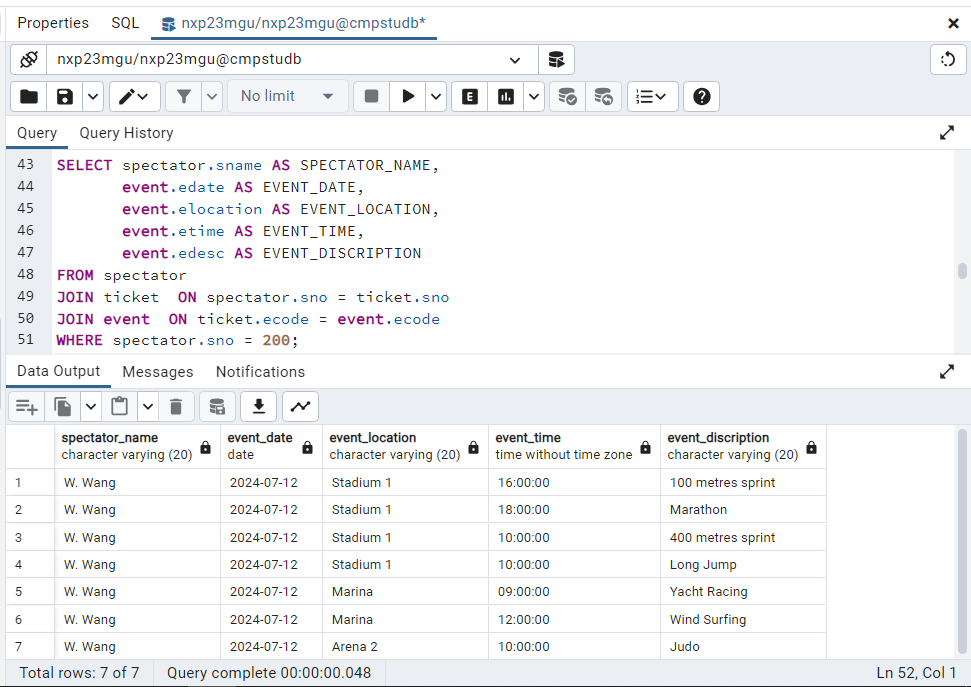
**(7) Testing task I**

--------------------------------------------------------------------------------------------------------------------------------------

Produce a report showing the schedule for a given spectator. The spectator is specified by his/her spectator number. The schedule should contain the spectator's name and the date, location, time and event description of each event for which the spectator has been issued a ticket.

sno = 200

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /5

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

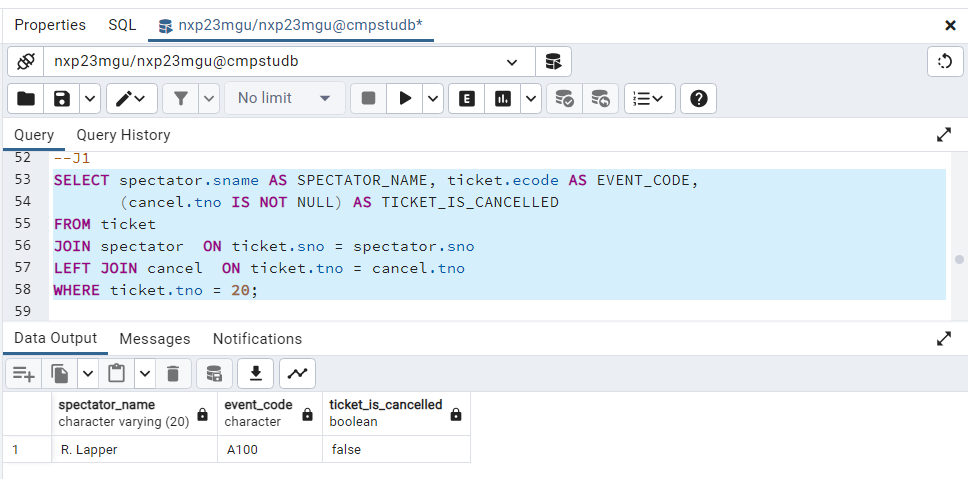
**(8) Testing task J(i)**

--------------------------------------------------------------------------------------------------------------------------------------

Given a specific ticket reference number, display the name of the spectator and the event code for the ticket and indicate if the ticket is valid or is cancelled.

tno = 20

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /4

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

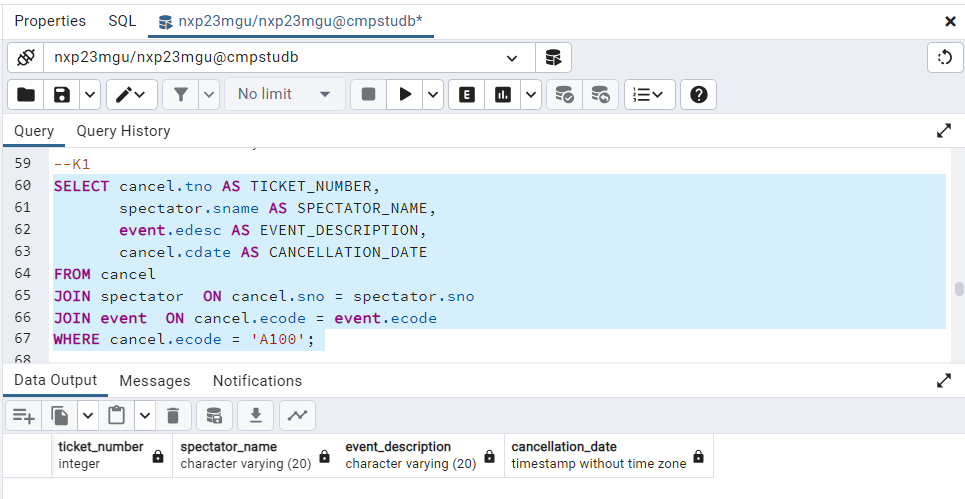
**(9) Testing task K (i)**

--------------------------------------------------------------------------------------------------------------------------------------

View the details of all cancelled tickets for a specific event.

ecode = 'A100'

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

**(10) Testing task D**

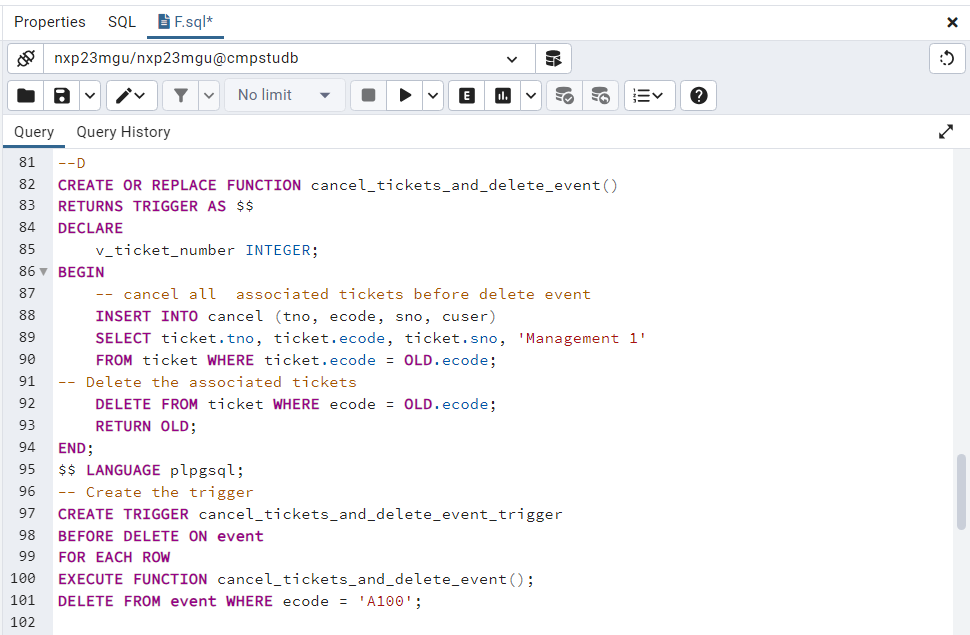
--------------------------------------------------------------------------------------------------------------------------------------

Delete an event. All the tickets for the event must be cancelled before an event can be deleted.

**ecode = 'A100'**

INSERT YOUR SQL QUERY AND OUTPUT HERE

--**Comment: Made Trigger function for cancel ticket record transfer to cancel table before delete event.** **Below code for delete event and trigger function executed and record of cancel ticket(output) in next task (Kii).**



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

**(11) Testing task K (ii)**

--------------------------------------------------------------------------------------------------------------------------------------

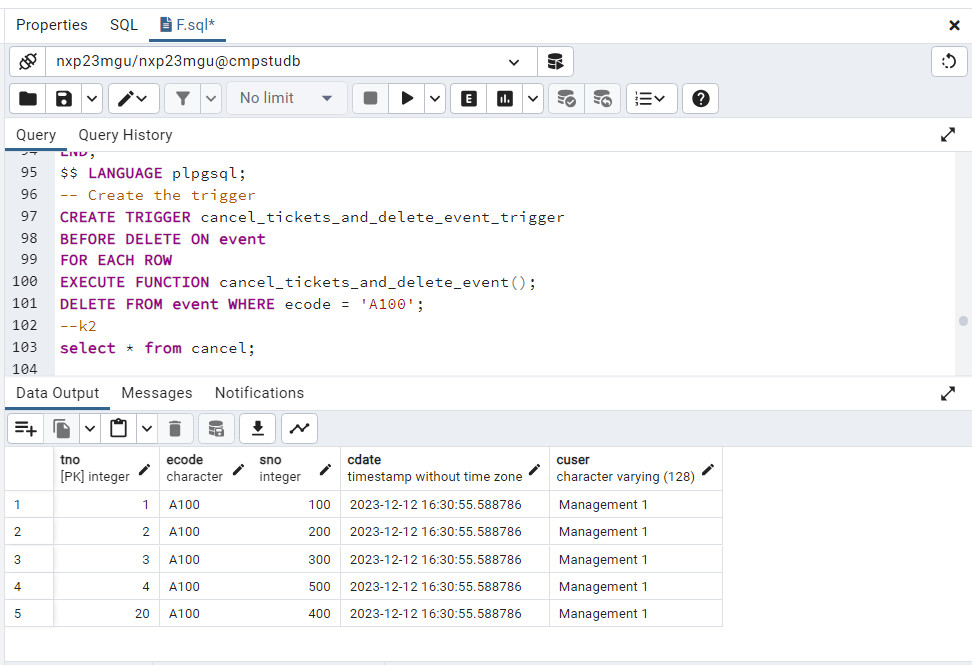
View the details of all cancelled tickets for a specific event.

ecode = 'A100'

INSERT YOUR SQL QUERY AND OUTPUT HERE

**--Comment: from above task D output reflect here as mentioned in previous task comment.**

**All canceled tickets records viewed.**



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /3

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

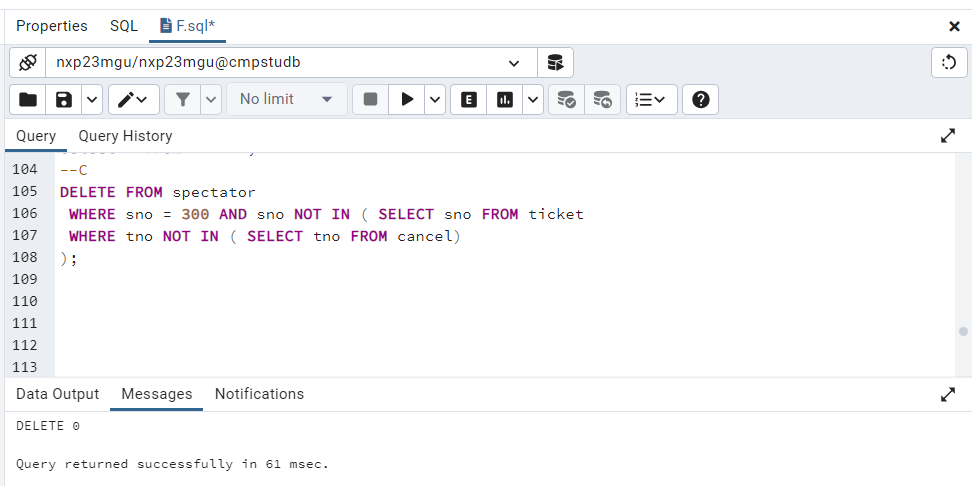
**(12) Testing task C (i)**

--------------------------------------------------------------------------------------------------------------------------------------

Delete a spectator. The spectator must not have any valid (i.e. not cancelled) tickets before it can be deleted.

**sno = 300;**

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

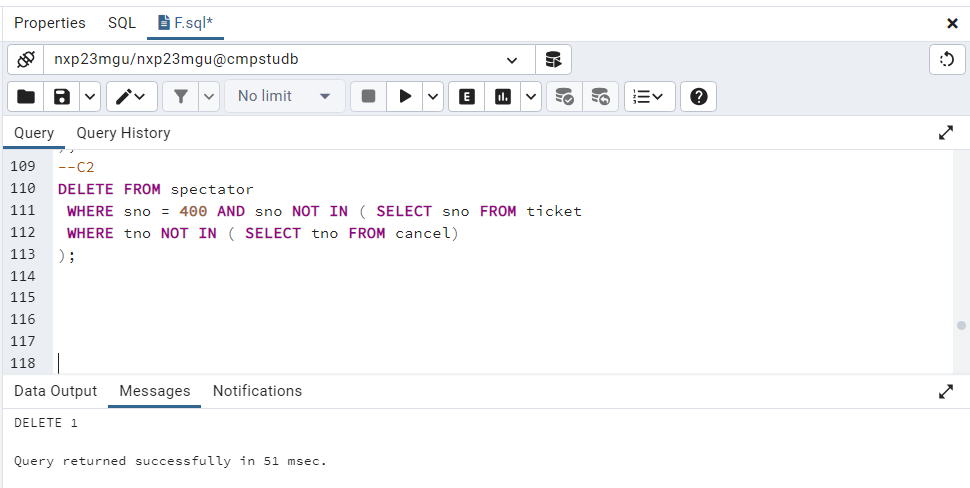
**(13) Testing task C (ii)**

--------------------------------------------------------------------------------------------------------------------------------------

Delete a spectator. The spectator must not have any valid (i.e. not cancelled) tickets before it can be deleted.

**sno = 400;**

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /2

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

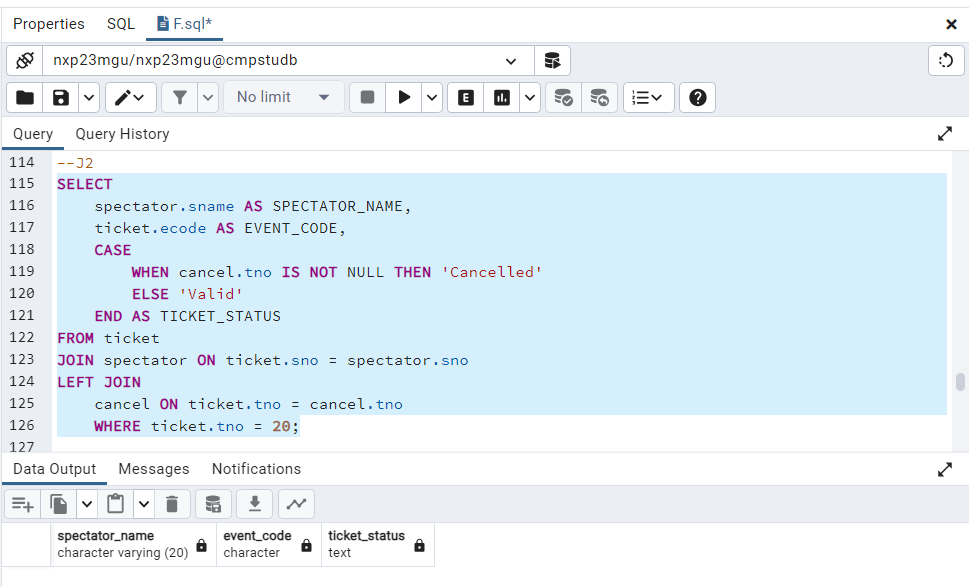
**(14) Testing task J (ii)**

--------------------------------------------------------------------------------------------------------------------------------------

Given a specific ticket reference number, display the name of the spectator and the event code for the ticket and indicate if the ticket is valid or is cancelled.

tno = 20

INSERT YOUR SQL QUERY AND OUTPUT HERE



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /4

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

**Checking the Constraints, Integrity, and Handling of the error cases:**

--------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------

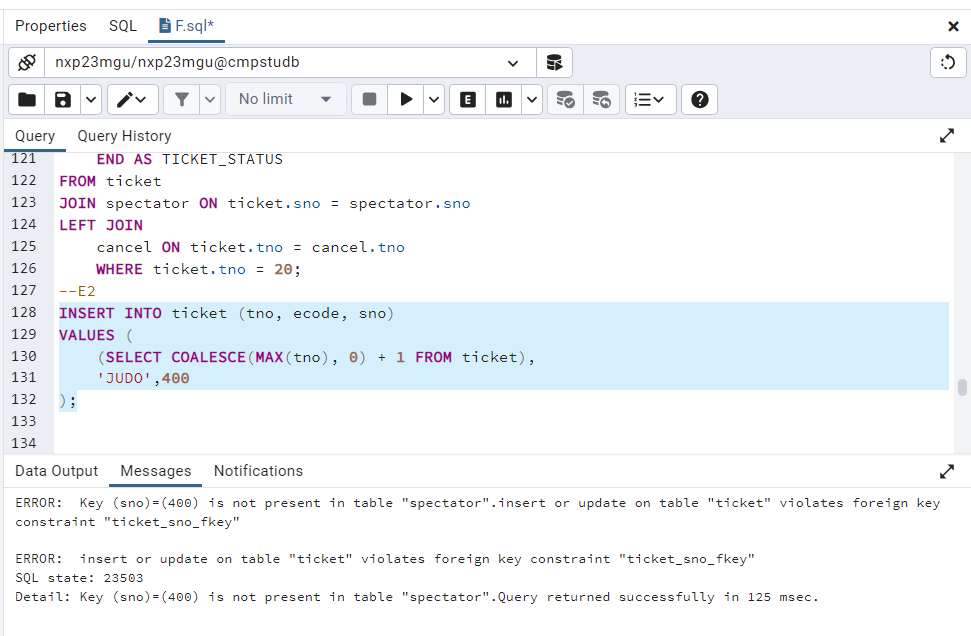
**(15) Testing task E (ii)**

--------------------------------------------------------------------------------------------------------------------------------------

Issue a ticket for an event.

ecode = 'JUDO' , sno = 400

INSERT YOUR SQL QUERY AND OUTPUT HERE



Marks: /1

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

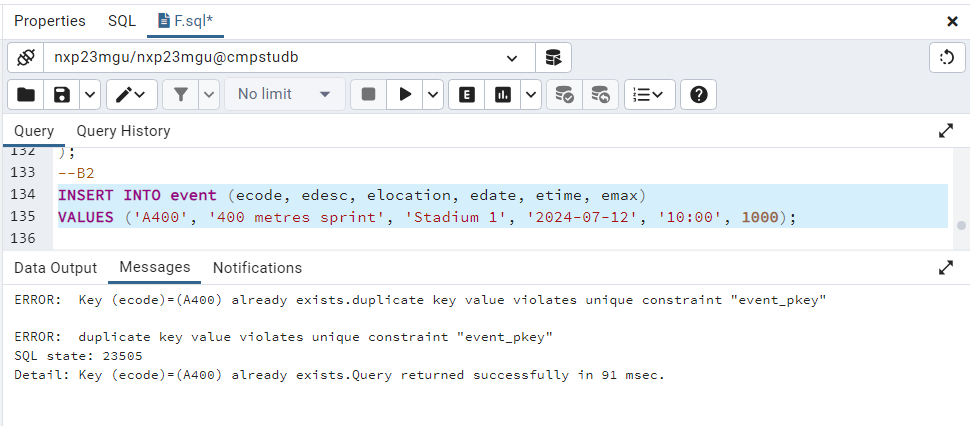
**(16) Testing task B (ii)**

--------------------------------------------------------------------------------------------------------------------------------------

Insert a new event.

ecode = 'A400', edesc = '400 metres sprint', elocation ='Stadium 1', edate = '2024-07-12', etime='10:00', emax = 1000

INSERT YOUR SQL QUERY AND OUTPUT HERE



Marks: /1

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

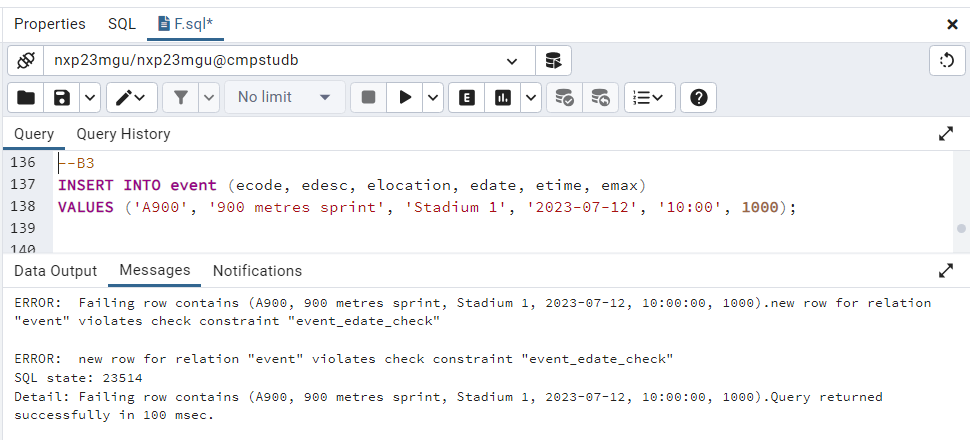
**(17) Testing task B (iii)**

--------------------------------------------------------------------------------------------------------------------------------------

Insert a new event.

ecode = 'A900', edesc = '900 metres sprint', elocation ='Stadium 1', edate = '2023-07-12', etime='10:00', emax = 1000

INSERT YOUR SQL QUERY AND OUTPUT HERE



Marks: /1

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

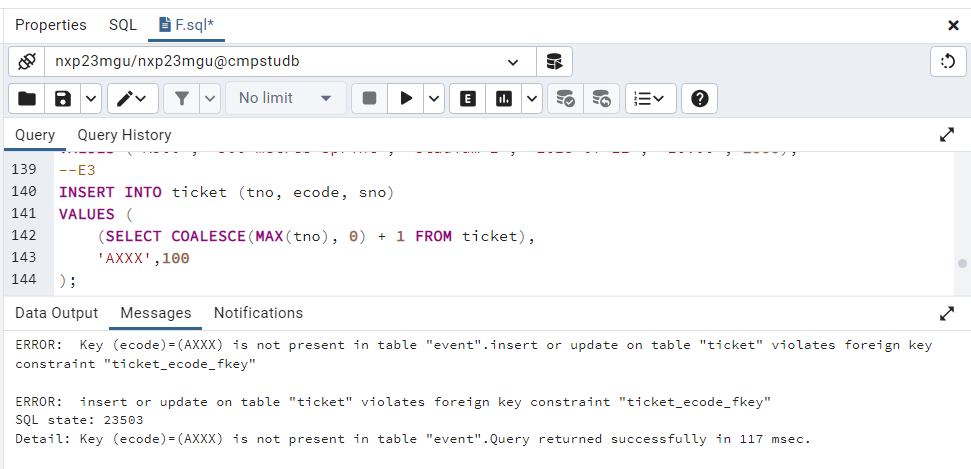
**(18) Testing task E (iii)**

--------------------------------------------------------------------------------------------------------------------------------------

Issue a ticket for an event.

ecode = 'AXXX' , sno = 100

INSERT YOUR SQL QUERY AND OUTPUT HERE



Marks: /1

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

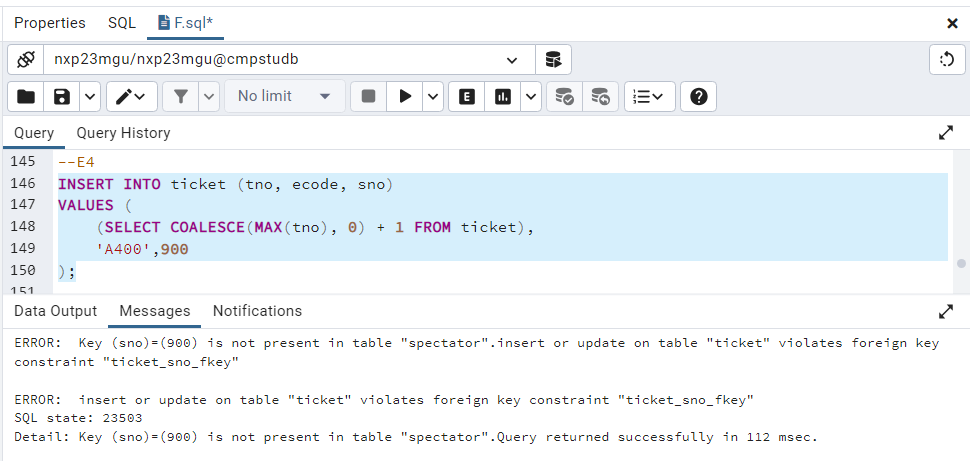
**(19) Testing task E (iv)**

--------------------------------------------------------------------------------------------------------------------------------------

Issue a ticket for an event.

ecode = 'A400' , sno = 900

INSERT YOUR SQL QUERY AND OUTPUT HERE



Marks: /1

Marker’s Comments (If needed):

--------------------------------------------------------------------------------------------------------------------------------------

**End of Constraint Check**

--------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------

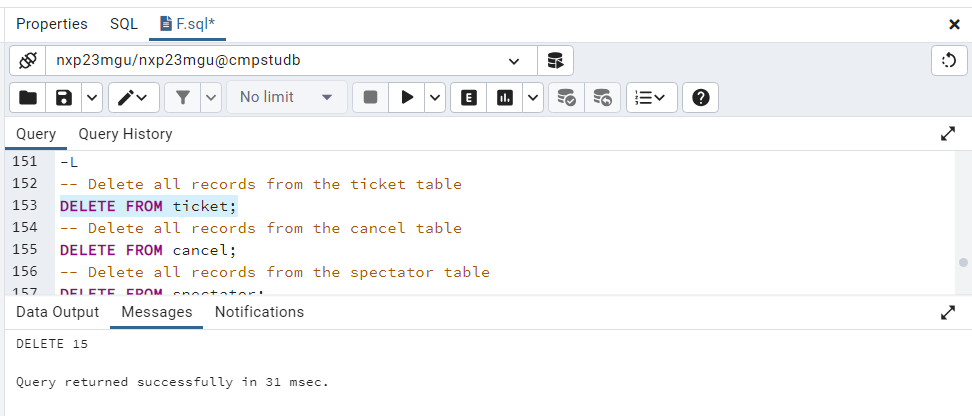
**(20) Testing task L**

--------------------------------------------------------------------------------------------------------------------------------------

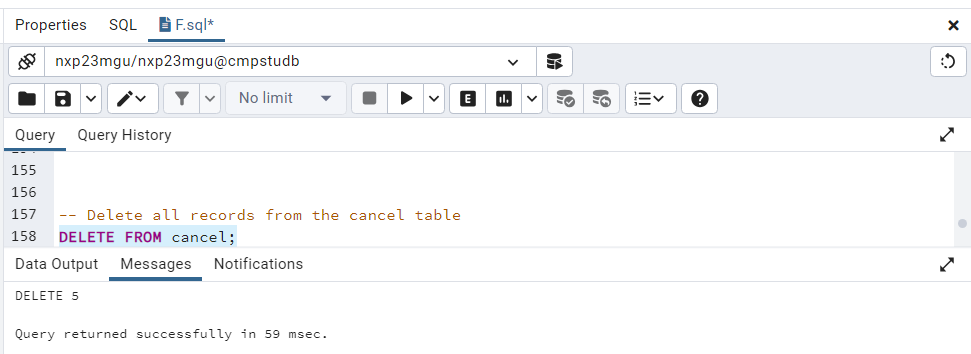
Delete the contents of the database tables.

INSERT YOUR SQL QUERY AND OUTPUT HERE

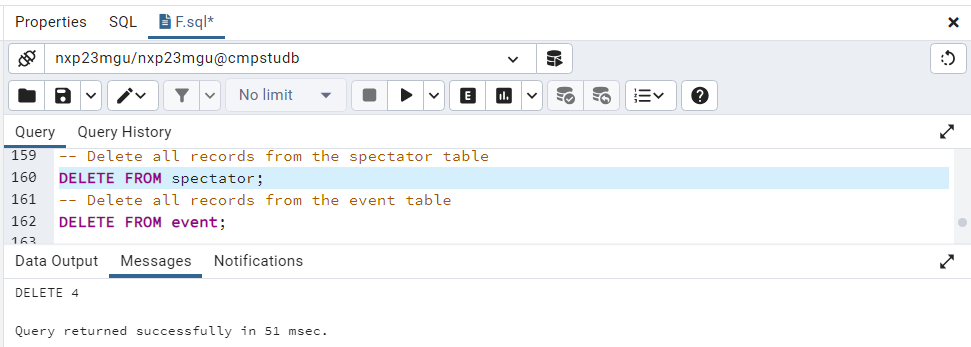
--Delete Ticket table record



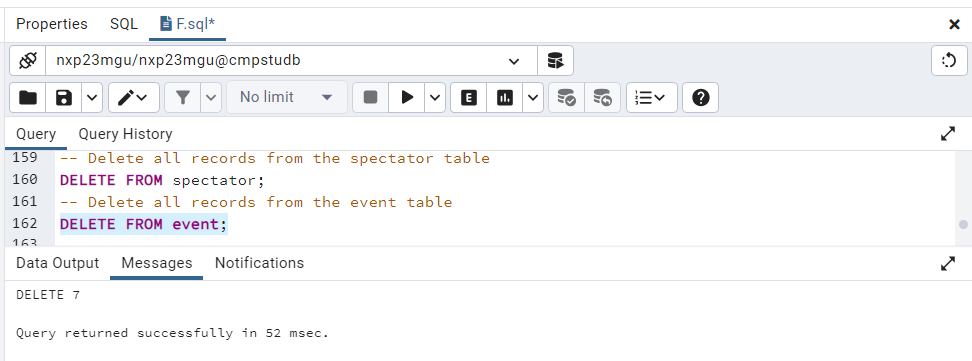
--Delete cancel table record



--Delete spectator table record



--Delete event table record



SHOW THE CONTENTS OF EACH AFFECTED TABLE (IF APPLICABLE) AT THE END OF THE TASK

Marks: /4

Marker’s Comments (If needed):

**Total marks for Part 2: /50**

**Part 3. Python application program**

Python application program communicates with the PostgreSQL server. Write the purpose of the main functions involved in the process.

**(Hint:** ﻿explain connection.cursor(), ﻿cursor.execute(), ﻿cursor.fetchall()**)**

Establish a Database Connection:

psycopg2.connect(): Creates a PostgreSQL connection.

Make a Cursor:

Using connection.cursor(), a cursor for running SQL queries is created.

Run SQL queries:

Executes SQL queries on the database using cursor.execute(sql\_query).

Retrieve Information:

Obtains query results with cursor.fetchone() and cursor.fetchall().

Engage in Transactions:

Changes are committed to the database using connection.commit().

Reverse Transactions:

Rollback modifications using connection.rollback() when an issue occurs.

Near Resources:

Resources are released via cursor.close() and connection.close().

Marks: /3

Marker’s Comments (If needed):

**Python Program Outputs**

**Enter/paste here the output from running input.txt.**

(Hint: the content of the **output.txt** file for the given **input.txt** file)

INSERT YOUR OUTPUT HERE

TASK A

SELECT 1

new spectator data verified successfully

(100, 'F Liza', 'f.liza@uea.ac.uk')

TASK B

SELECT 1

new event data inserted successfully

('A100', '100 metres sprint', 'Stadium 1', datetime.date(2024, 7, 12), datetime.time(16, 0), 1000)

TASK E

SELECT 1

TASK F

SELECT 1

Ticket report generated successfully

[('Stadium 1', datetime.date(2024, 7, 12), 1)]

TASK G

SELECT 1

Event report generated successfully

[('100 metres sprint', 1)]

TASK H

SELECT 1

For Event , Specific ticket report generated by event code successfully

[('100 metres sprint', 'A100', 1)]

TASK I

SELECT 1

spectator schedule report generated successfully

[('F Liza', datetime.date(2024, 7, 12), 'Stadium 1', datetime.time(16, 0), '100 metres sprint')]

TASK J

SELECT 1

Specific ticket ref number and name displayed successfully

[('F Liza', 'A100', False)]

TASK D

SELECT 0

[]

Event data deleted successful

TASK K

SELECT 1

Cancelled tickets report for event A100 generated successfully

[(1, 'A100', 100, datetime.datetime(2023, 12, 13, 14, 37, 26, 778494), 'Management 1')]

TASK C

SELECT 0

Spectator data deleted successful

[]

TASK L

SELECT 0

[]

Delete all data from event table successful

SELECT 0

[]

Delete all data from spectator table successful

SELECT 0

[]

Delete all data from ticket table successful

SELECT 0

[]

Delete all data from cancel table successful

Marks: /15

Marker’s Comments (If needed):

TOTAL PYTHON MARKS /18

Submission Quality:

The quality of the submission is according to the instructions: /6

**Total Mark Distribution:**

**MARKS DDL section /26**

**MARKS DML section /50**

**MARKS FOR Python Application /18**

**Marks for quality of submission /6**

**TOTAL MARKS /100**

**APPENDIX:**  EXAMPLE OF HOW TO FILL THIS FORM/DOCUMENT (using different indicative tasks).

--------------------------------------------------------------------------------------------------------------------------------------

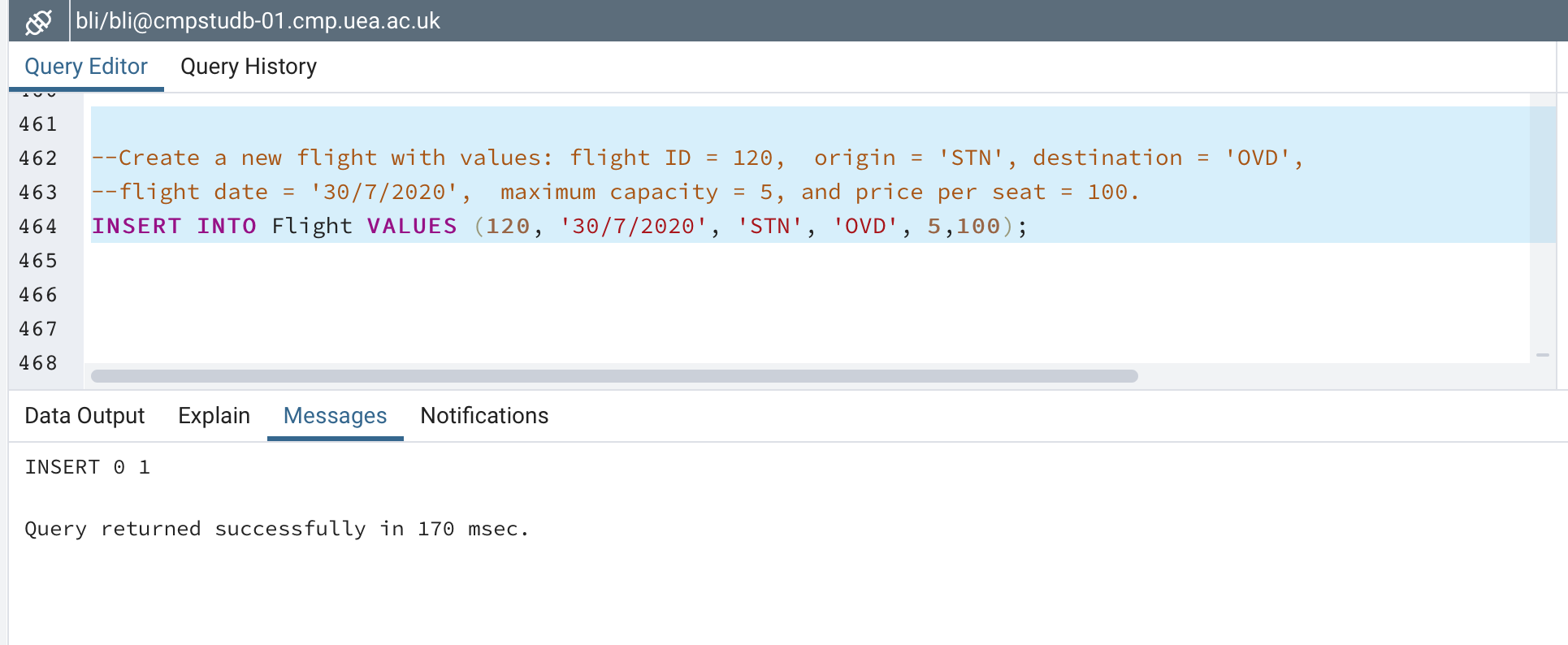
**Testing task 1**

--------------------------------------------------------------------------------------------------------------------------------------

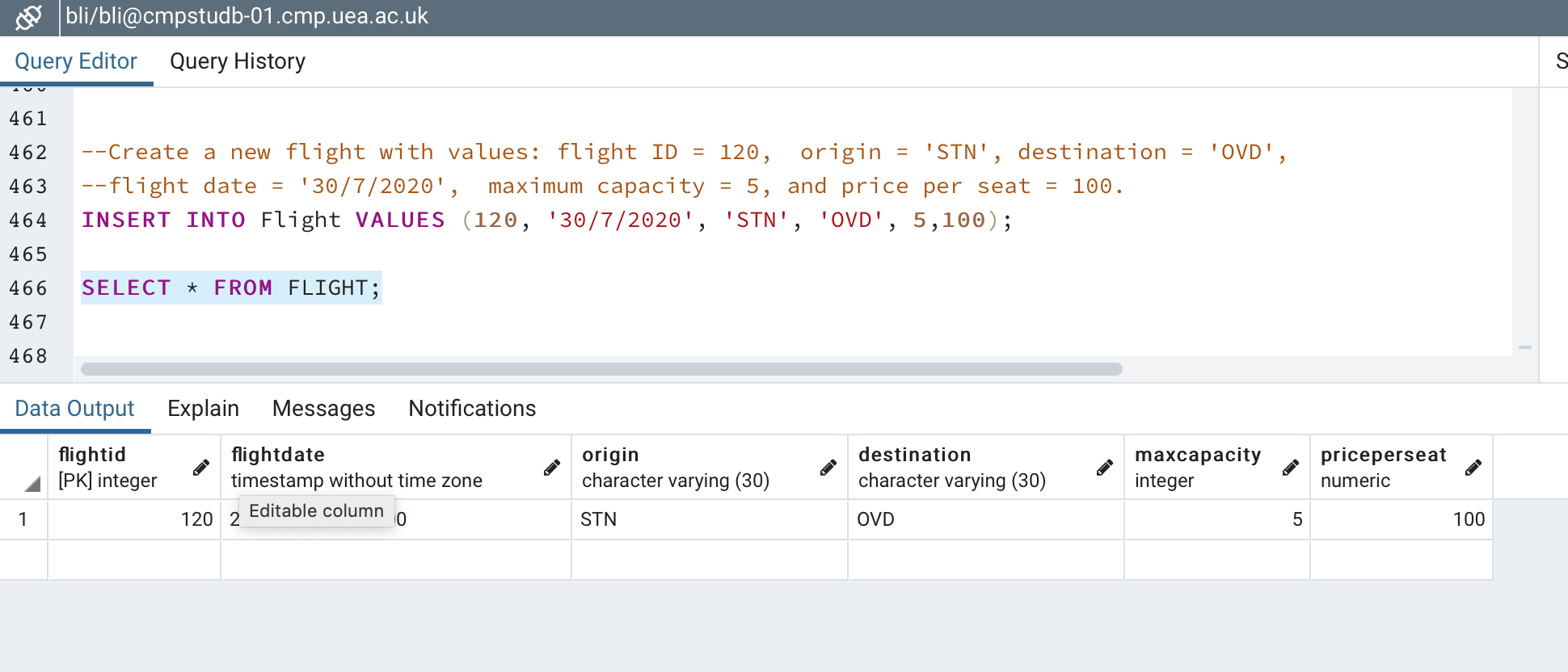
**1. Given flight details create new flight record**

1. Create a new flight with values: flight ID = **120**, origin = **'STN'**, destination = **'OVD'**, flight date = **'30/7/2020'**, maximum capacity = **5**, and price per seat = **100**.

INSERT YOUR SQL QUERY AND OUTPUT HERE

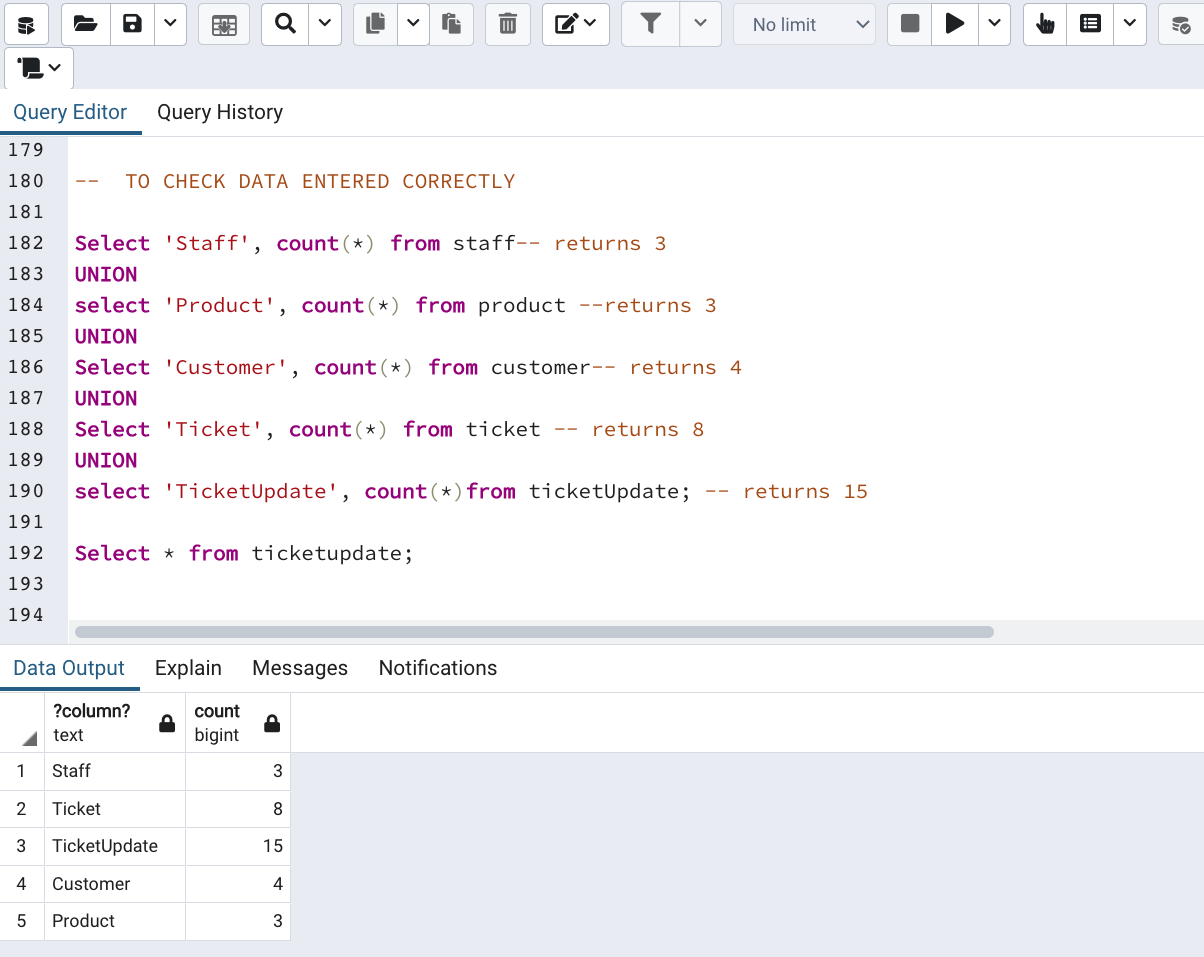


SHOW THE CONTENTS OF FLIGHT TABLE AT THE END OF THE TASK

****

**2. Produce a query that counts the number of tuples in each table.**

INSERT YOUR SQL QUERY AND OUTPUT HERE

****