GEBZE TECHNICAL UNIVERSITY

CSE 414
DATABASE COURSE

THE EVENT MANAGEMENT SYSTEM PROJECT REPORT

BARAN SOLMAZ
1801042601

CONTENTS

1) PROBLEM DEFINITION

The main purpose of this project is to efficiently manage the database of an Event Management System. This aim will be achieved by creating a comprehensive database that includes information about various events, event organizers, participants, venues and other relevant entities.

The Event Management System will enable event organizers to minimize their time and resources by systematically maintaining event details, allowing for quick access to the data when needed. By using this system, event organizers can reduce their workload, save costs, and minimize the use of physical documents. The user-friendly interface of the event management system will greatly assist event organizers in making their process easier and the system more efficient.

2) IMPLEMENTATION DETAILS

All requests were performed with MySQL commands using Python. The interface is built with the Python Tkinter library.

3) USER REQUIREMENTS

The Event Management System has 4 different main users.

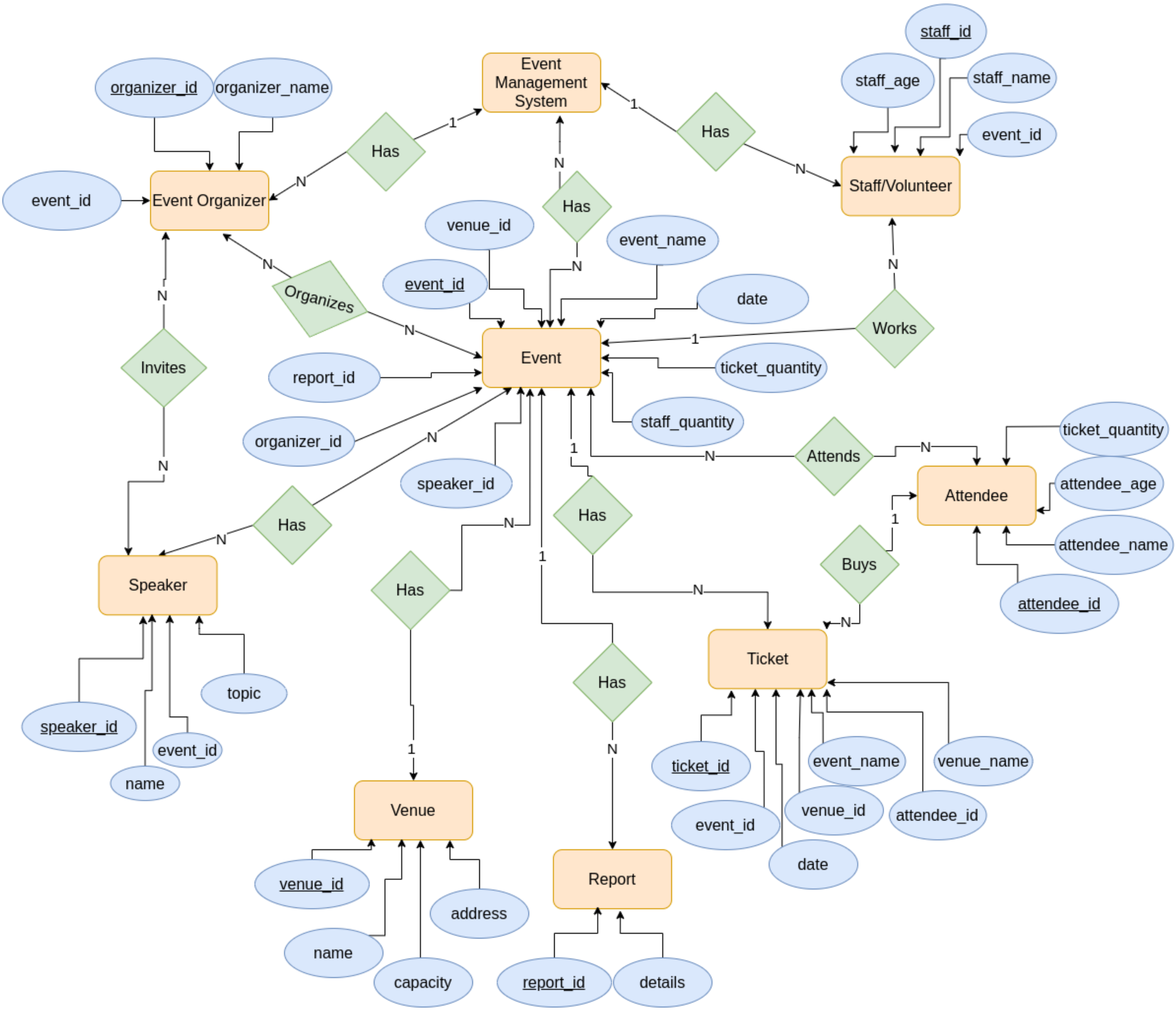
- Event Organizers
    - Can create, update, cancel event,
    - Can see events and details,
    - Can generate reports on event,
    - Can view reports,
    - Can view speakers,
    - Can view venues,

- Attendees
  - Can see events,
  - Can list and add attendees,
  - Can list tickets,
  - Can buy and cancel ticket,

- Staff/Volunteers
  - Can see events and details,
  - Can volunteer,
  - Can list staff,
  - Can add,edit and remove staff.

- Speakers
  - Can attend events,
  - Can see events,
  - Can change their topic,
  - Can list speakers,
  - Can add speaker.

4) ER DIAGRAM



staff_id
staff_age
staff_name
event_id
organizer_id
organizer_name
Event Management System
Has
1
1
N
Has
N
event_id
Event Organizer
N
Has
N
Staff/Volunteer
venue_id
event_name
event_id
N
date
N
Organizes
N
Works
N
1
Event
Invites
report_id
ticket_quantity
staff_quantity
organizer_id
N
ticket_quantity
1
N
Attends
N
speaker_id
Has
attendee_age
Attendee
N
Has
N
1
Buys
attendee_name
N
Has
1
N
Speaker
attendee_id
N
Ticket
Has
topic
speaker_id
event_id
name
ticket_id
event_name
venue_name
1
N
event_id
venue_id
attendee_id
Venue
date
venue_id
address
Report
name
capacity
report_id
details

5) FUNCTIONAL DEPENDENCIES

- Event

event_id⟶event_name,**venue_id**,venue_name,**organizer_id**,
organizer_name,**speaker_id**,speaker_name,ticket_quantity,
staff_quantity,**report_id,**date

- Organizer

organizer_id → organizer_name,**event_id**

- Venue

venue_id → venue_name,capacity,address

- Attendee

attendee_id → attendee_name,attendee_age,**ticket_id**

- Ticket

ticket_id →**event_id**,event_name,date,**venue_id**,venue_name

- Staff/Volunteer

staff_id → staff_name,staff_age,**event_id**

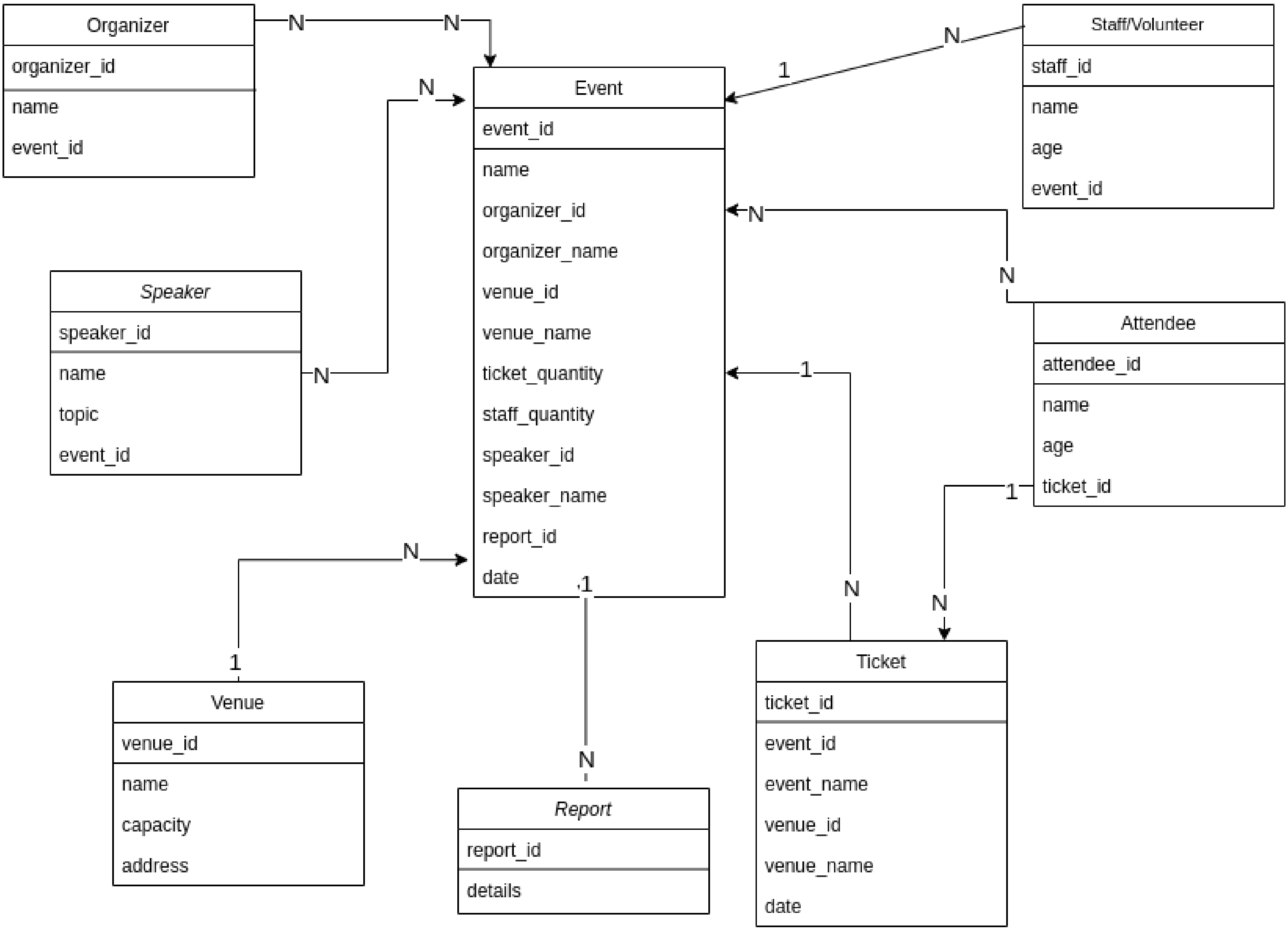
- Speaker

speaker_id →speaker_name, topic,**event_id**

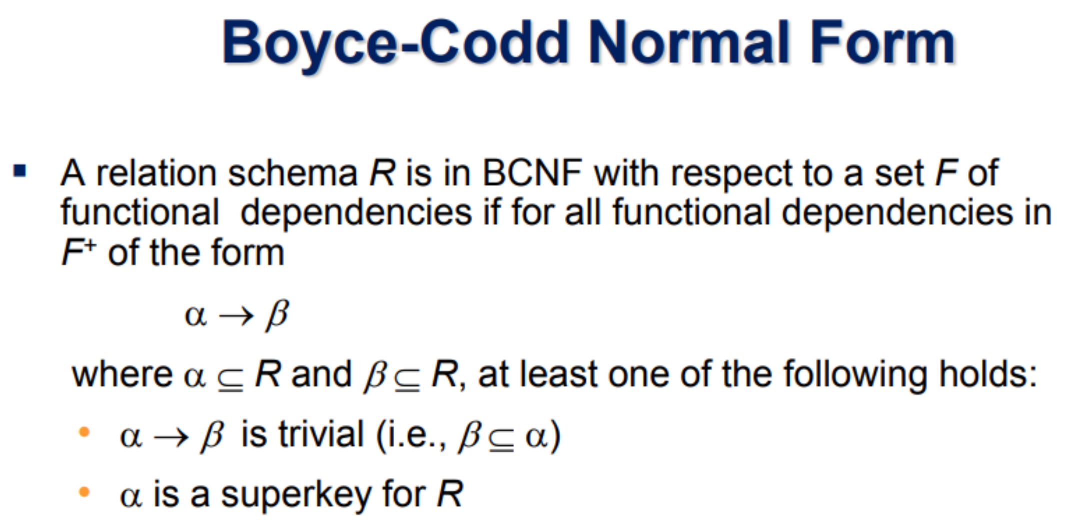
- Report

report_id →details

6) TABLES


Organizer
organizer_id
name
event_id
N
N
Event
event_id
name
organizer_id
organizer_name
venue_id
venue_name
ticket_quantity
staff_quantity
speaker_id
speaker_name
report_id
date
N
1
Staff/Volunteer
staff_id
name
age
event_id
N
N
N
Speaker
speaker_id
name
topic
event_id
N
Attendee
attendee_id
name
age
ticket_id
1
1
N
N
N
1
N
1
Venue
venue_id
name
capacity
address
Report
report_id
details
Ticket
ticket_id
event_id
event_name
venue_id
venue_name
date

7) NORMALIZATIONS

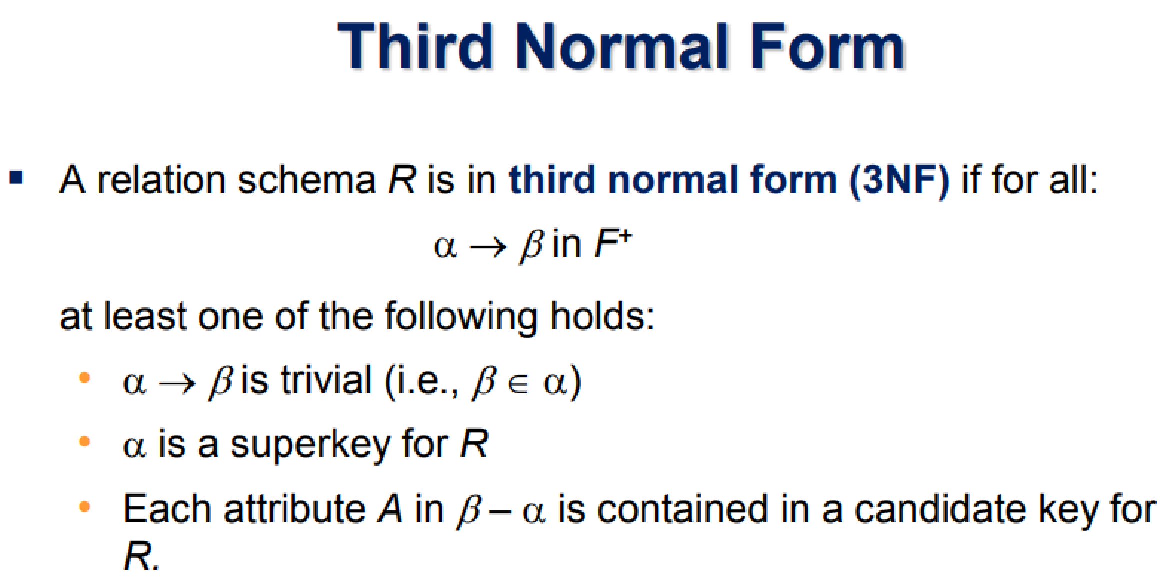


Boyce-Codd Normal Form

- A relation schema *R* is in BCNF with respect to a set *F* of functional dependencies if for all functional dependencies in F^+ of the form

  $\alpha \rightarrow \beta$

  where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for *R*



Third Normal Form

- A relation schema *R* is in **third normal form (3NF)** if for all:

  $\alpha \rightarrow \beta$ in F^+

  at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
  - α is a superkey for *R*
  - Each attribute *A* in $\beta - \alpha$ is contained in a candidate key for *R*.

- Venue Table

venue_id → venue_name,capacity,address

The candidate key is **venue_id**,
The set of key attributes are { **venue_id** }.
venue_id is a super-key for Venue Table.
Dependency is trivial.
The Venue Table is in BCNF and 3NF. No need for normalization.

- Report Table

report_id →details

The candidate key is **report_id**,
The set of key attributes are { **report_id** }.
report_id is a super-key for Report Table.
Dependency is trivial.
The Report Table is in BCNF and 3NF. No need for normalization.

- Attendee Table

attendee_id → attendee_name,attendee_age,**ticket_id**

The candidate key is **attendee_id**,
The set of key attributes are { **attendee_id** }.
attendee_id is a super-key for Attendee Table.
Dependency is trivial.
The Attendee Table is in BCNF and 3NF. No need for normalization.

- Organizer Table

organizer_id → organizer_name,**event_id**

The candidate key is **organizer_id**,
The set of key attributes are { **organizer_id** }.
organizer_id is a super-key for Organizer Table.
Dependency is trivial.
The Organizer Table is in BCNF and 3NF. No need for normalization.

- Staff/Volunteer Table

staff_id → staff_name,staff_age,**event_id**

The candidate key is **staff_id**,
The set of key attributes are { **staff_id** }.
staff_id is a super-key for Staff/Volunteer Table.
Dependency is trivial.
The Staff/Volunteer Table is in BCNF and 3NF. No need for normalization.

- Speaker Table

speaker_id →speaker_name, topic,**event_id**

The candidate key is **speaker_id**,

The set of key attributes are { **speaker_id** }.

speaker_id is a super-key for Speaker Table.

Dependency is trivial.

The Speaker Table is in BCNF and 3NF. No need for normalization.

- Ticket Table

ticket_id →**event_id**,event_name,date,**venue_id**,venue_name
event_id →event_name,date
venue_id →venue_name

The candidate key is **ticket_id**,

The set of key attributes are { **ticket_id** }.

ticket_id is a super-key for Speaker Table.

Since we can get event_name,date,venue_name from **event_id** and **venue_id** , dependency is non-trivial.This violates BCNF and 3NF.

The Ticket Table is not in BCNF and 3NF. Needs normalization.

– If we remove event_name,date and venue_name from the table, Dependency will become trivial and the Ticket Table will be in BCNF and 3NF.

- Event Table

event_id⟶event_name,**venue_id**,venue_name,**organizer_id**,
organizer_name,**speaker_id**,speaker_name,ticket_quantity,
staff_quantity,**report_id**,date

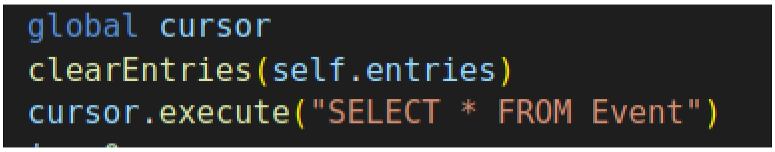
venue_id → venue_name
organizer_id → organizer_name
speaker_id →speaker_name

Since we can get venue_name, organizer_name, and speaker_name from **organizer_id, venue_id and speaker_id** , dependency is non-trivial.This violates BCNF and 3NF.
The Event Table is not in BCNF and 3NF. Needs normalization.

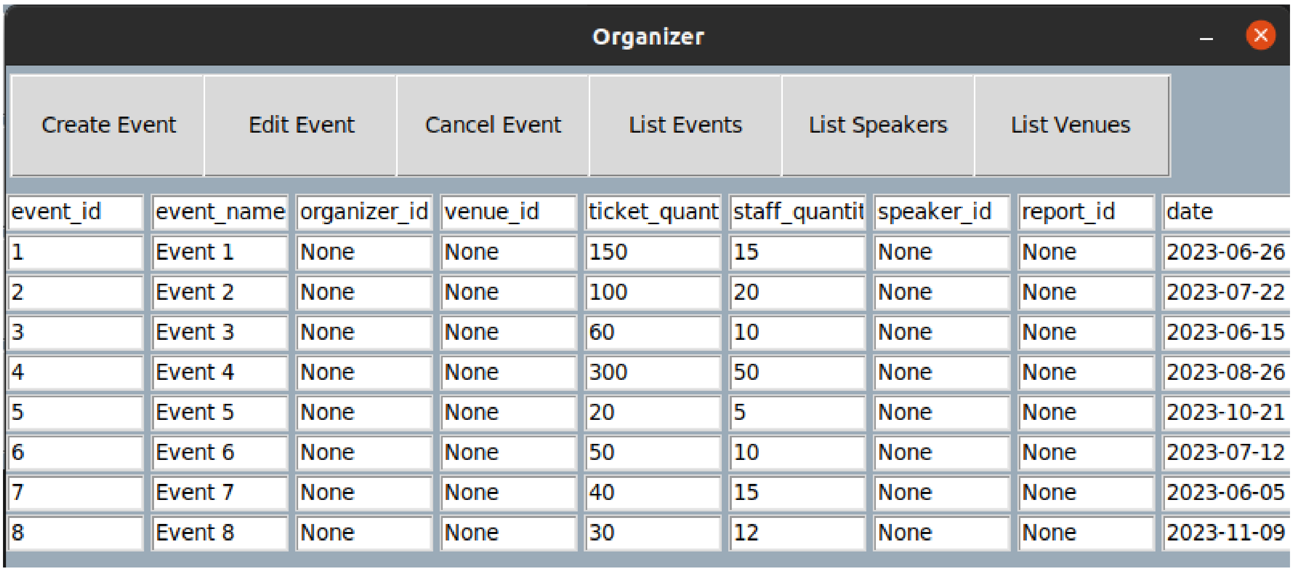
> – If we remove venue_name, organizer_name and speaker_name from the table,Dependency will become trivial and the Event Table will be in BCNF and 3NF.

8) QUERIES

- SELECT
    - SELECT * FROM Event

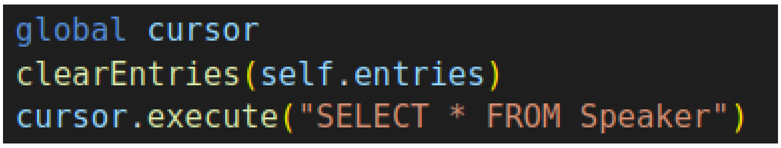


```
global cursor
clearEntries(self.entries)
cursor.execute("SELECT * FROM Event")
```

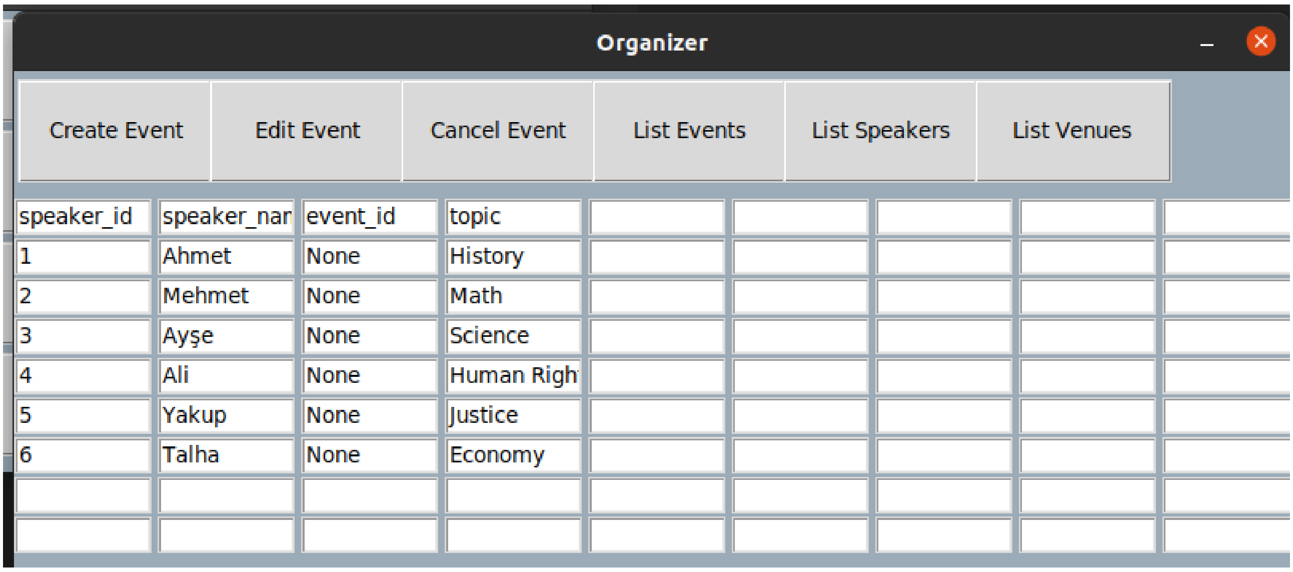


event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date
1	Event 1	None	None	150	15	None	None	2023-06-26
2	Event 2	None	None	100	20	None	None	2023-07-22
3	Event 3	None	None	60	10	None	None	2023-06-15
4	Event 4	None	None	300	50	None	None	2023-08-26
5	Event 5	None	None	20	5	None	None	2023-10-21
6	Event 6	None	None	50	10	None	None	2023-07-12
7	Event 7	None	None	40	15	None	None	2023-06-05
8	Event 8	None	None	30	12	None	None	2023-11-09

-
    - SELECT * FROM Speaker

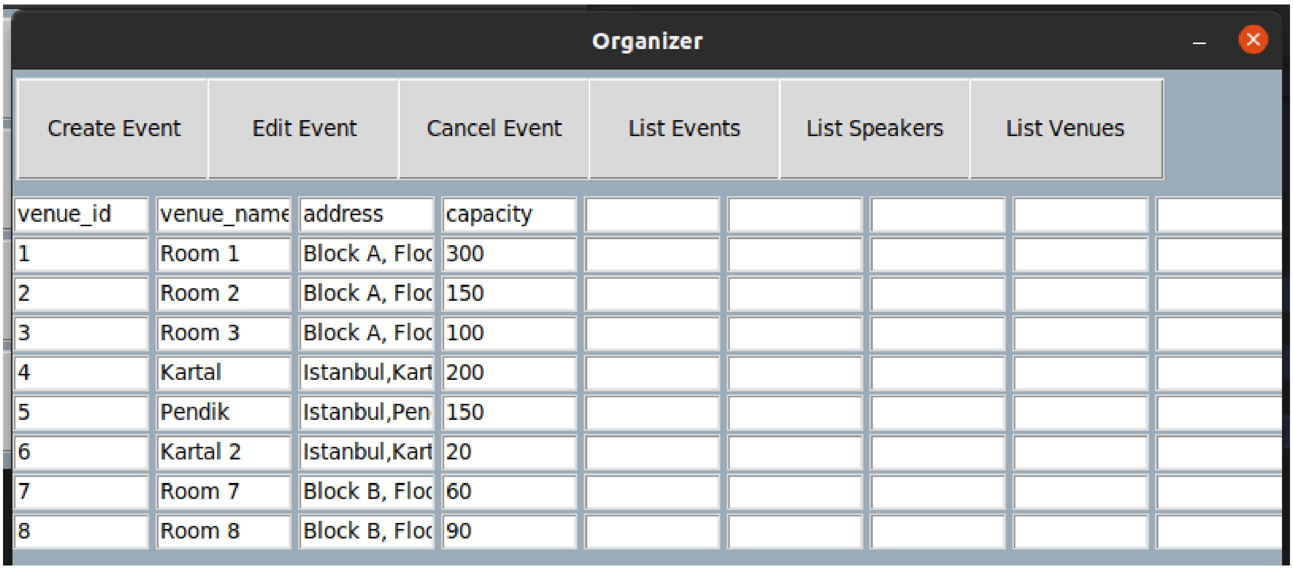


```
global cursor
clearEntries(self.entries)
cursor.execute("SELECT * FROM Speaker")
```



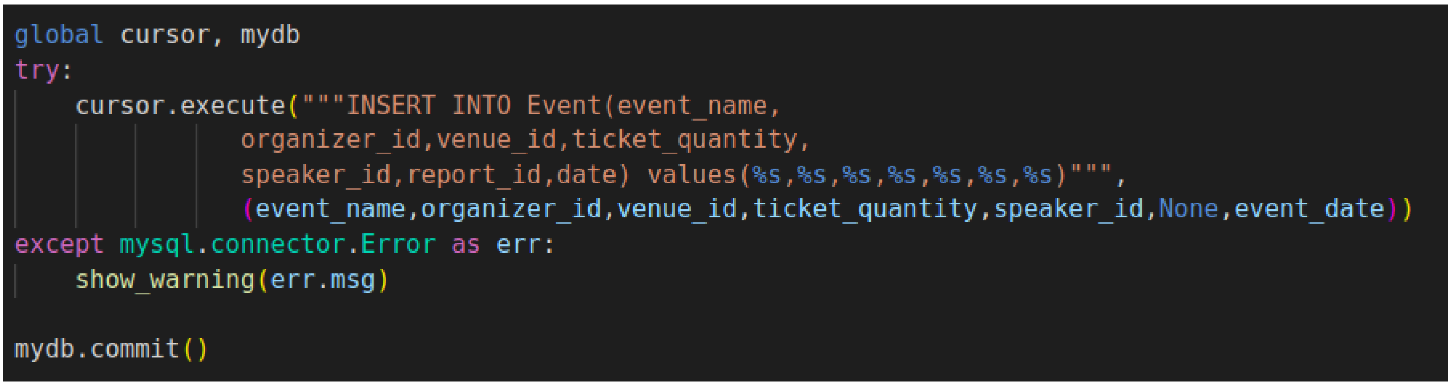
speaker_id	speaker_nar	event_id	topic
1	Ahmet	None	History
2	Mehmet	None	Math
3	Ayşe	None	Science
4	Ali	None	Human Righ
5	Yakup	None	Justice
6	Talha	None	Economy

- ○ SELECT * FROM Venue



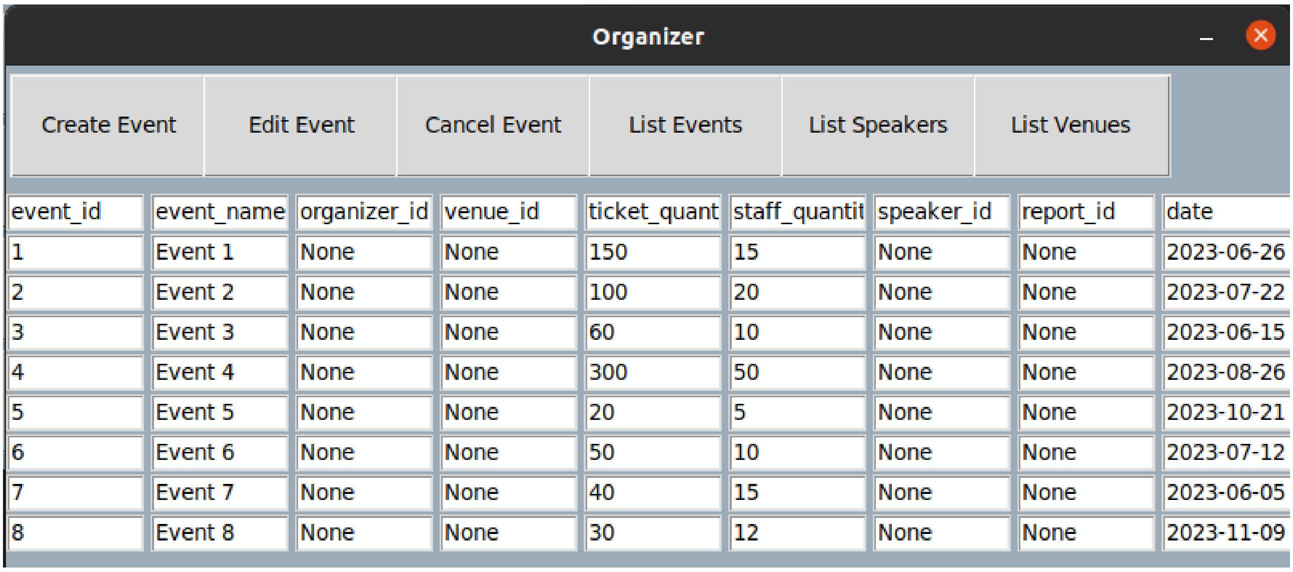
venue_id	venue_name	address	capacity
1	Room 1	Block A, Floo	300
2	Room 2	Block A, Floo	150
3	Room 3	Block A, Floo	100
4	Kartal	Istanbul,Kart	200
5	Pendik	Istanbul,Pen	150
6	Kartal 2	Istanbul,Kart	20
7	Room 7	Block B, Floo	60
8	Room 8	Block B, Floo	90

- INSERT
  - INSERT INTO Event



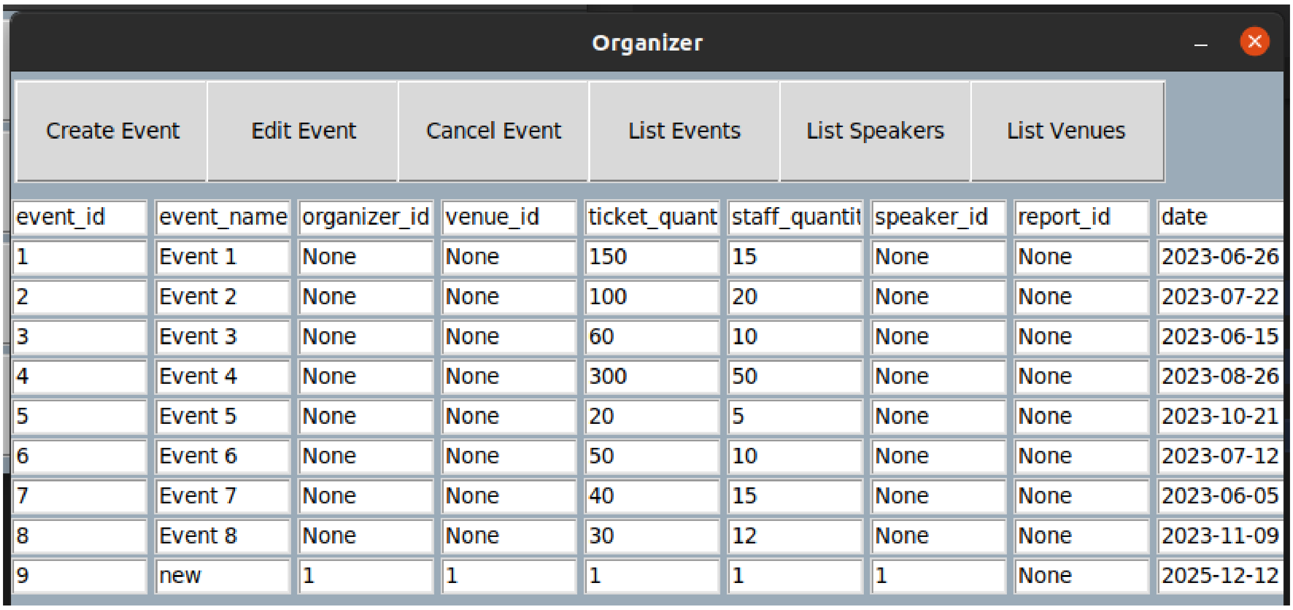
```
global cursor, mydb
try:
    cursor.execute("""INSERT INTO Event(event_name,
                organizer_id,venue_id,ticket_quantity,
                speaker_id,report_id,date) values(%s,%s,%s,%s,%s,%s,%s)""",
                (event_name,organizer_id,venue_id,ticket_quantity,speaker_id,None,event_date))
except mysql.connector.Error as err:
    show_warning(err.msg)

mydb.commit()
```

Before:



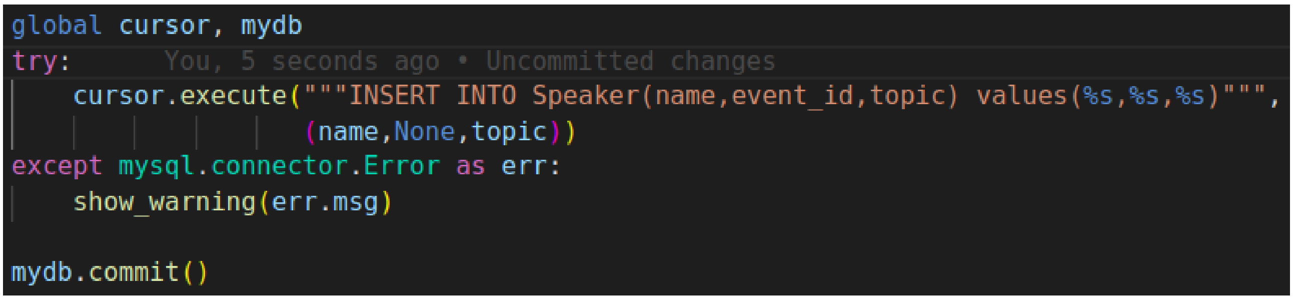
event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date
1	Event 1	None	None	150	15	None	None	2023-06-26
2	Event 2	None	None	100	20	None	None	2023-07-22
3	Event 3	None	None	60	10	None	None	2023-06-15
4	Event 4	None	None	300	50	None	None	2023-08-26
5	Event 5	None	None	20	5	None	None	2023-10-21
6	Event 6	None	None	50	10	None	None	2023-07-12
7	Event 7	None	None	40	15	None	None	2023-06-05
8	Event 8	None	None	30	12	None	None	2023-11-09

After:



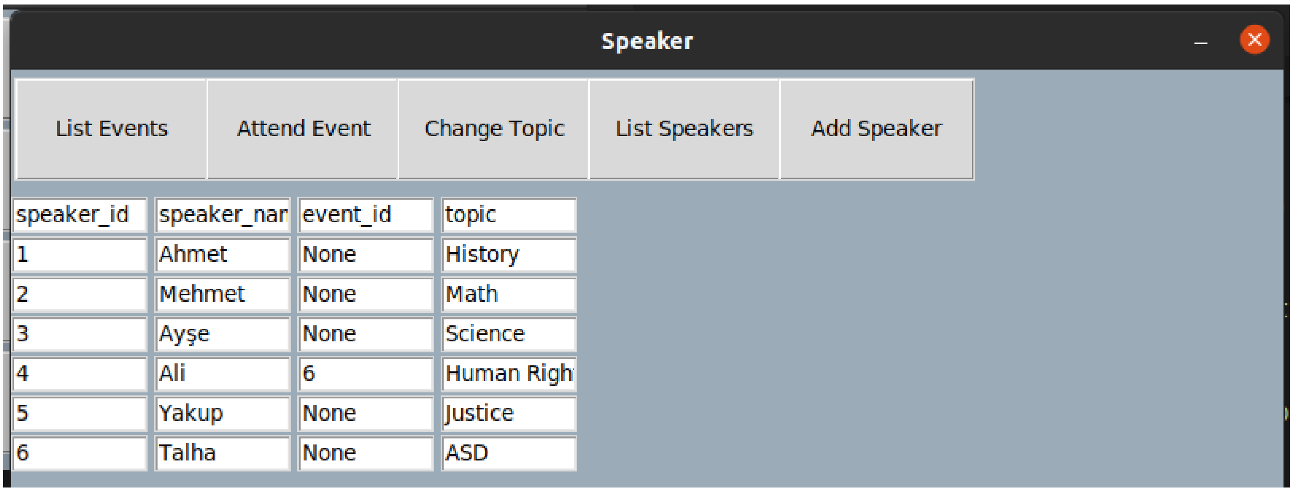
event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date
1	Event 1	None	None	150	15	None	None	2023-06-26
2	Event 2	None	None	100	20	None	None	2023-07-22
3	Event 3	None	None	60	10	None	None	2023-06-15
4	Event 4	None	None	300	50	None	None	2023-08-26
5	Event 5	None	None	20	5	None	None	2023-10-21
6	Event 6	None	None	50	10	None	None	2023-07-12
7	Event 7	None	None	40	15	None	None	2023-06-05
8	Event 8	None	None	30	12	None	None	2023-11-09
9	new	1	1	1	1	1	None	2025-12-12

- INSERT INTO Speaker



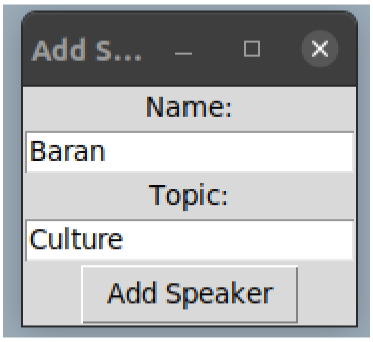
```
global cursor, mydb
try:        You, 5 seconds ago • Uncommitted changes
    cursor.execute("""INSERT INTO Speaker(name,event_id,topic) values(%s,%s,%s)""",
                   (name,None,topic))
except mysql.connector.Error as err:
    show_warning(err.msg)

mydb.commit()
```

Before:

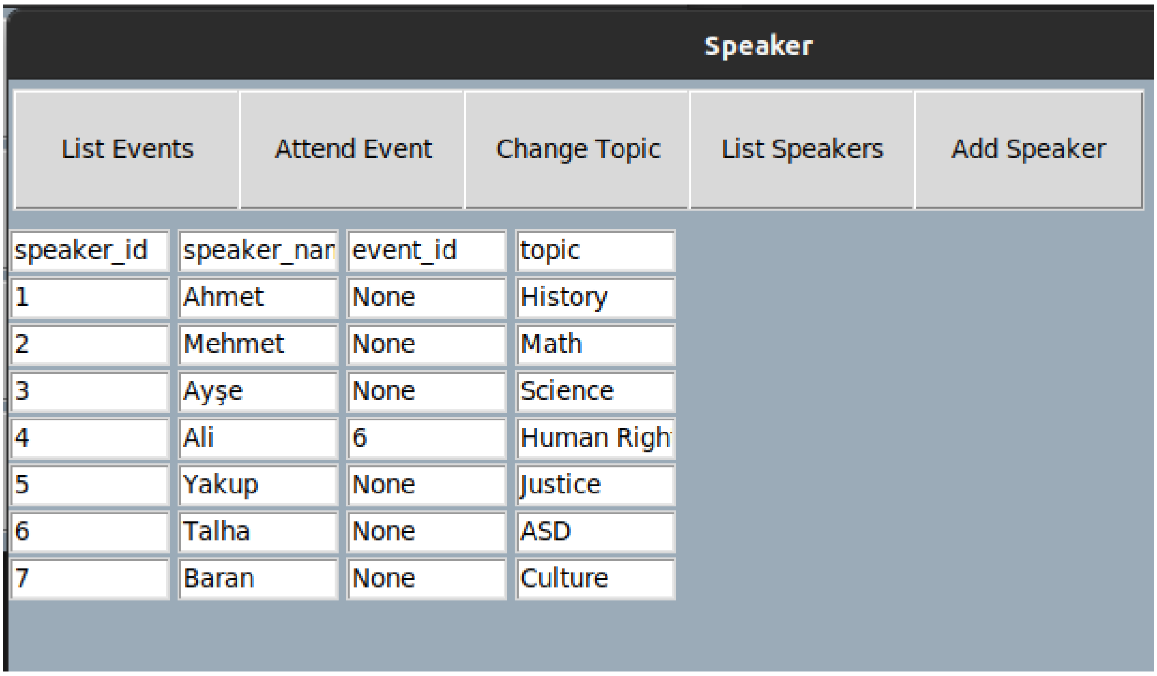


speaker_id	speaker_nar	event_id	topic
1	Ahmet	None	History
2	Mehmet	None	Math
3	Ayşe	None	Science
4	Ali	6	Human Righ
5	Yakup	None	Justice
6	Talha	None	ASD

After:

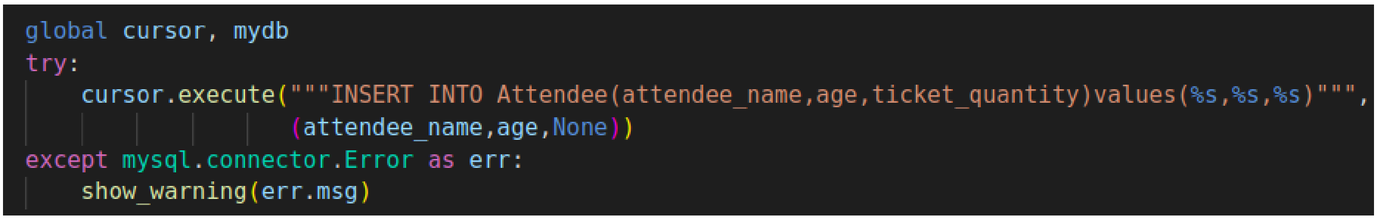


Name:
Baran
Topic:
Culture



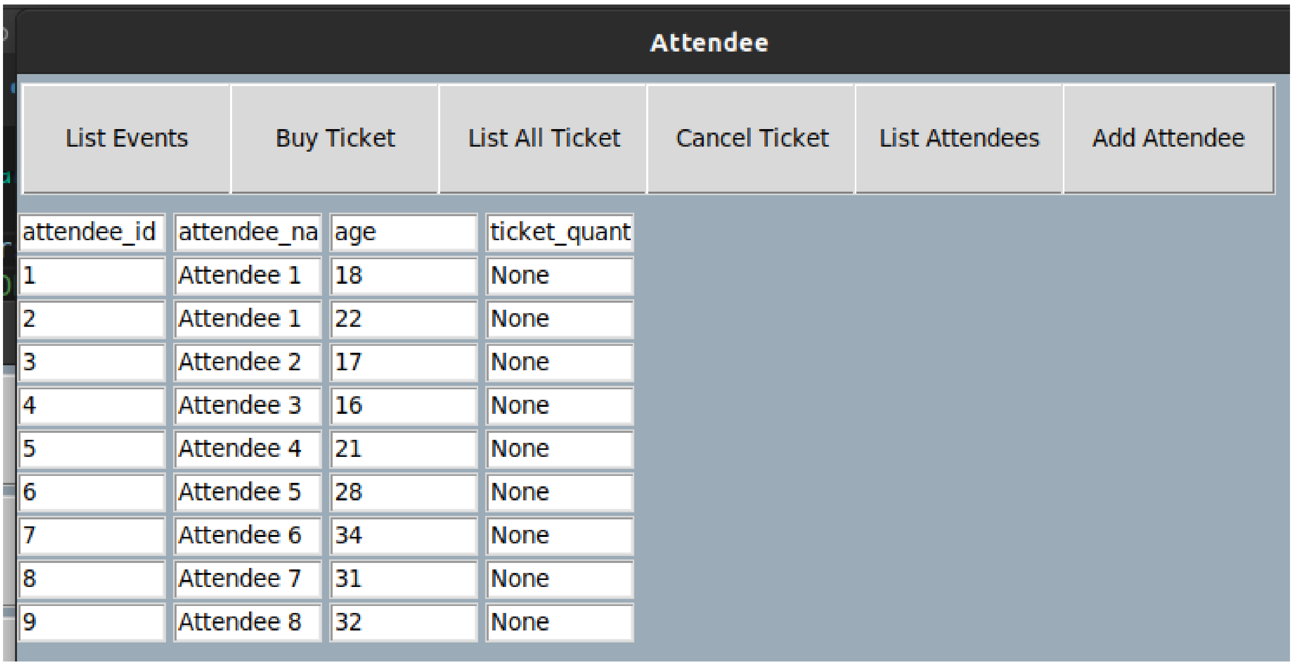
speaker_id	speaker_nar	event_id	topic
1	Ahmet	None	History
2	Mehmet	None	Math
3	Ayşe	None	Science
4	Ali	6	Human Righ
5	Yakup	None	Justice
6	Talha	None	ASD
7	Baran	None	Culture

- INSERT INTO Attendee



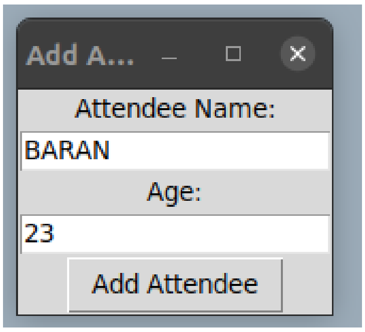
```
global cursor, mydb
try:
    cursor.execute("""INSERT INTO Attendee(attendee_name,age,ticket_quantity)values(%s,%s,%s)""",
                    (attendee_name,age,None))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:



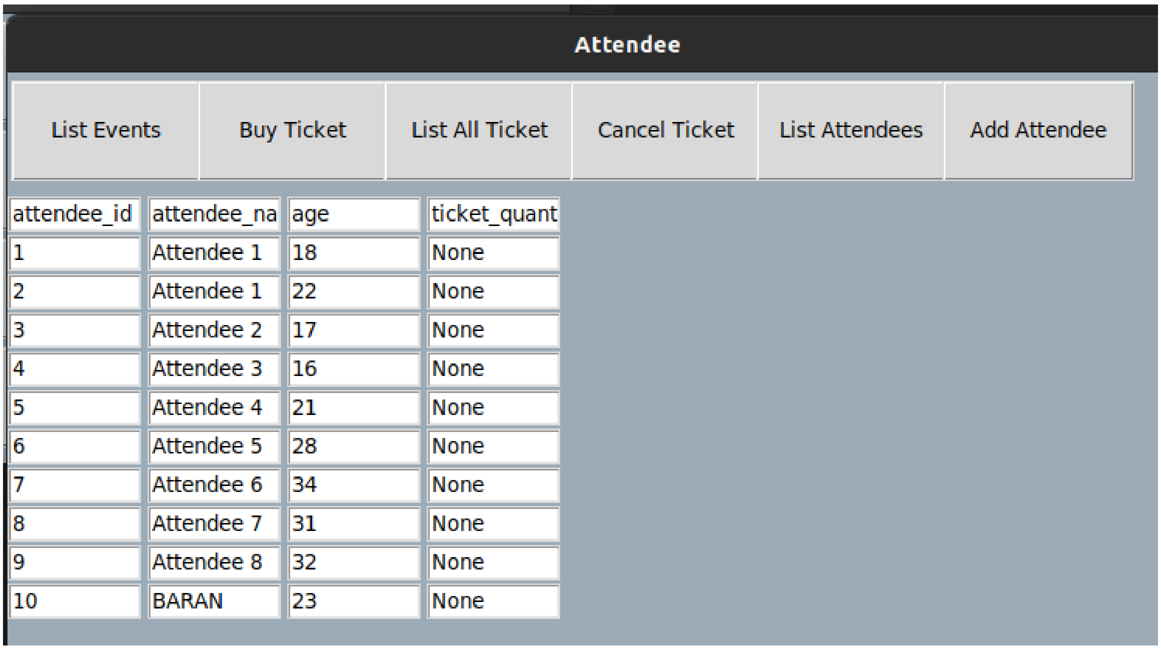
attendee_id	attendee_na	age	ticket_quant
1	Attendee 1	18	None
2	Attendee 1	22	None
3	Attendee 2	17	None
4	Attendee 3	16	None
5	Attendee 4	21	None
6	Attendee 5	28	None
7	Attendee 6	34	None
8	Attendee 7	31	None
9	Attendee 8	32	None

After:



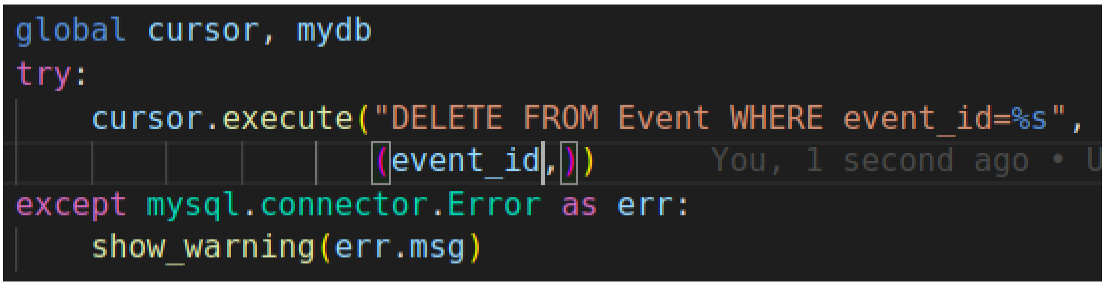
Attendee Name:
BARAN

Age:
23



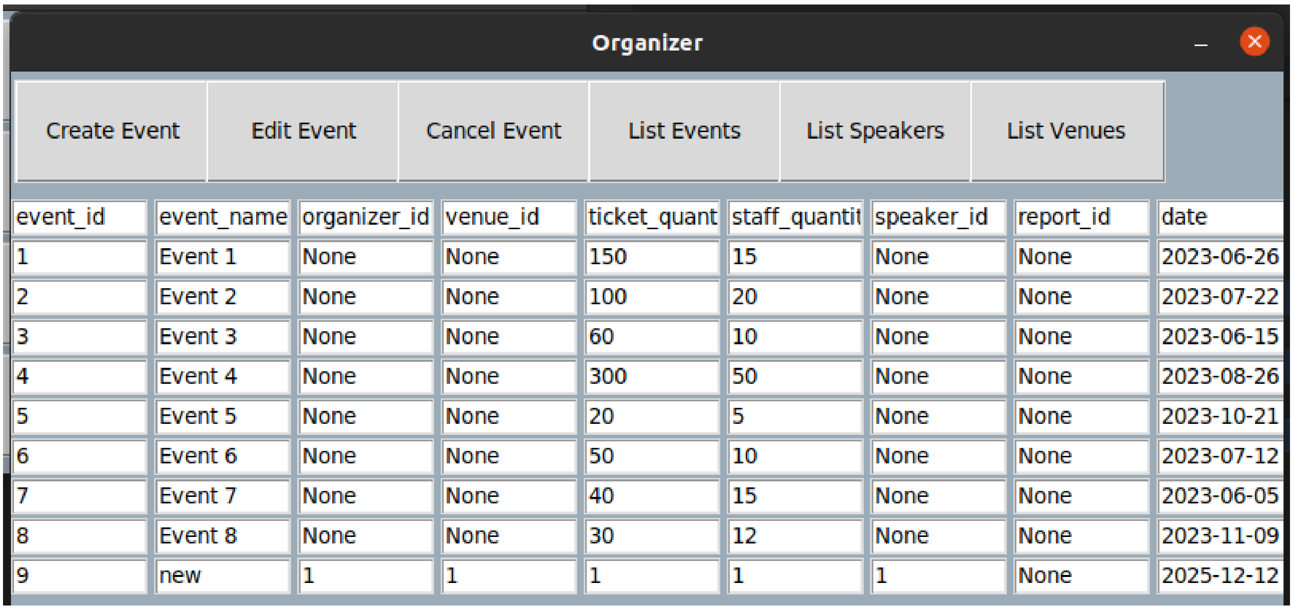
attendee_id	attendee_na	age	ticket_quant
1	Attendee 1	18	None
2	Attendee 1	22	None
3	Attendee 2	17	None
4	Attendee 3	16	None
5	Attendee 4	21	None
6	Attendee 5	28	None
7	Attendee 6	34	None
8	Attendee 7	31	None
9	Attendee 8	32	None
10	BARAN	23	None

- DELETE
    - DELETE id FROM Event



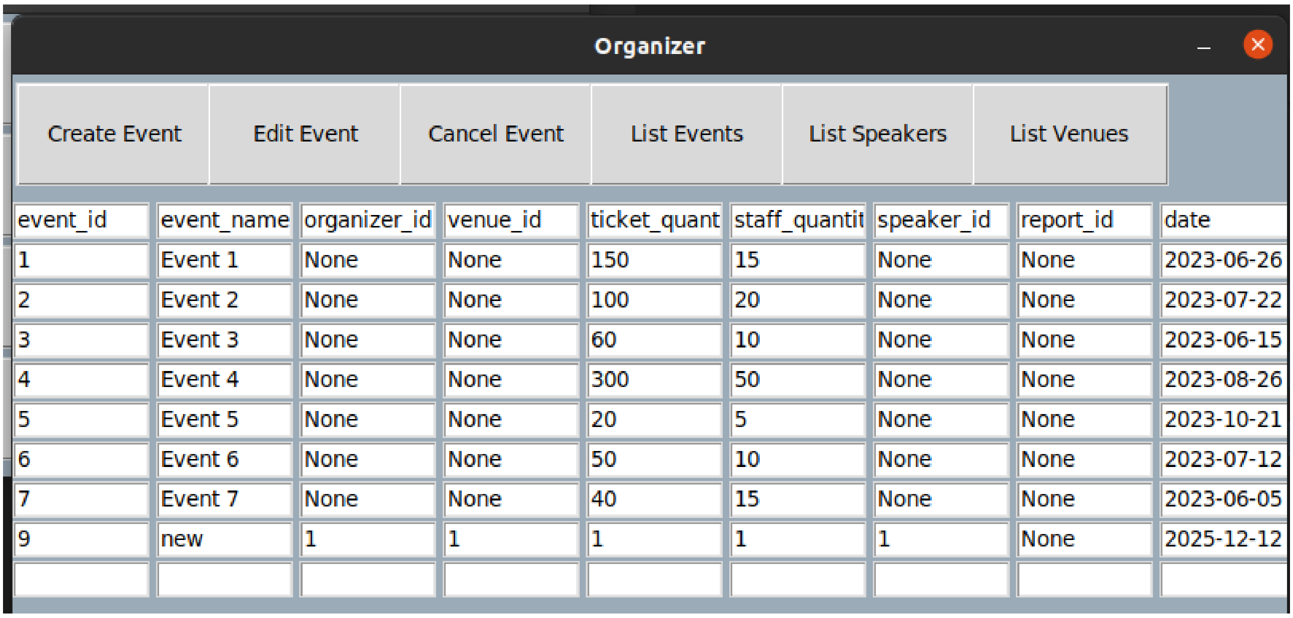
```
global cursor, mydb
try:
    cursor.execute("DELETE FROM Event WHERE event_id=%s",
                   (event_id,))        You, 1 second ago • U
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:



event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date
1	Event 1	None	None	150	15	None	None	2023-06-26
2	Event 2	None	None	100	20	None	None	2023-07-22
3	Event 3	None	None	60	10	None	None	2023-06-15
4	Event 4	None	None	300	50	None	None	2023-08-26
5	Event 5	None	None	20	5	None	None	2023-10-21
6	Event 6	None	None	50	10	None	None	2023-07-12
7	Event 7	None	None	40	15	None	None	2023-06-05
8	Event 8	None	None	30	12	None	None	2023-11-09
9	new	1	1	1	1	1	None	2025-12-12

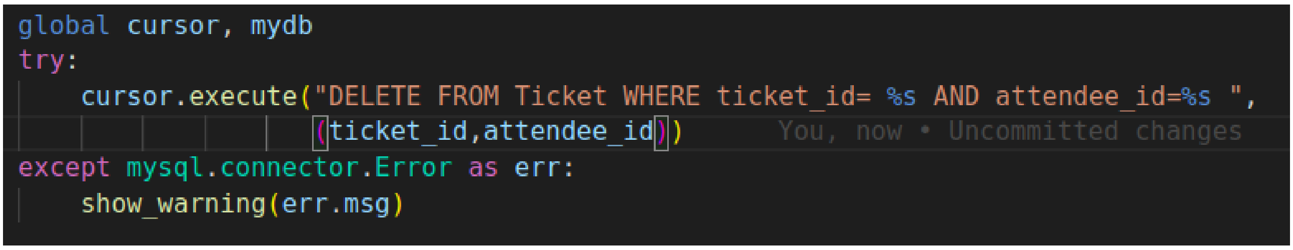
After:



event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date
1	Event 1	None	None	150	15	None	None	2023-06-26
2	Event 2	None	None	100	20	None	None	2023-07-22
3	Event 3	None	None	60	10	None	None	2023-06-15
4	Event 4	None	None	300	50	None	None	2023-08-26
5	Event 5	None	None	20	5	None	None	2023-10-21
6	Event 6	None	None	50	10	None	None	2023-07-12
7	Event 7	None	None	40	15	None	None	2023-06-05
9	new	1	1	1	1	1	None	2025-12-12

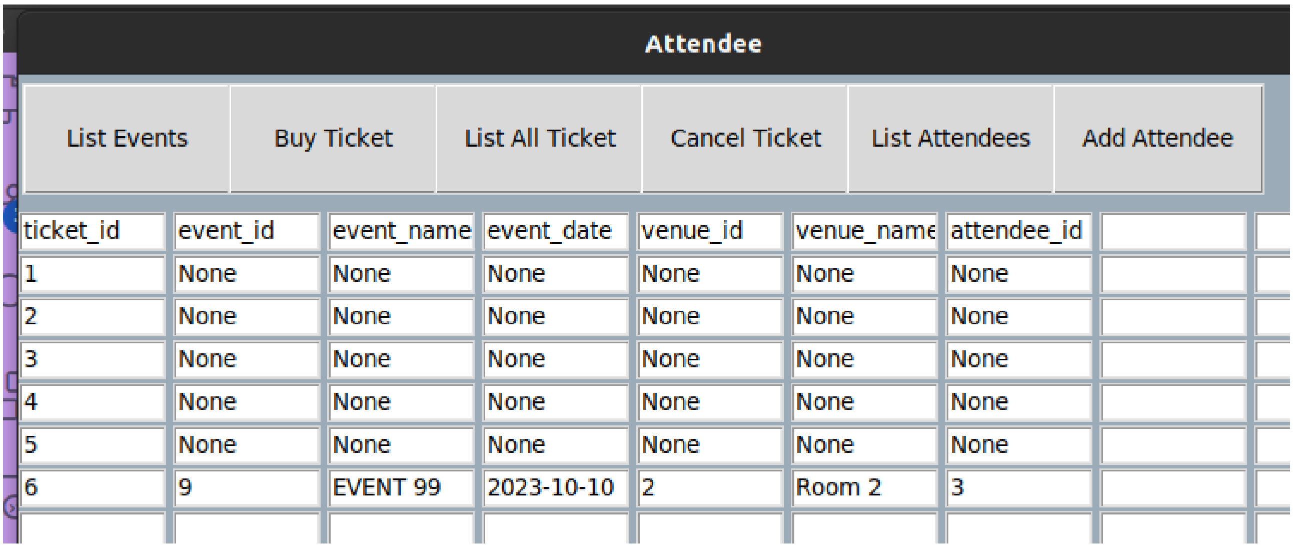
-id(8) removed.

- DELETE ticket_id FROM Ticket



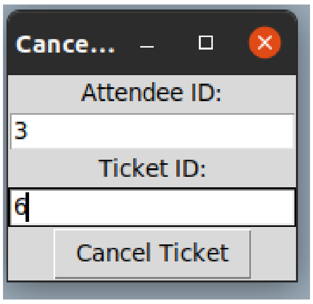
```
global cursor, mydb
try:
    cursor.execute("DELETE FROM Ticket WHERE ticket_id= %s AND attendee_id=%s ",
                   (ticket_id,attendee_id))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:



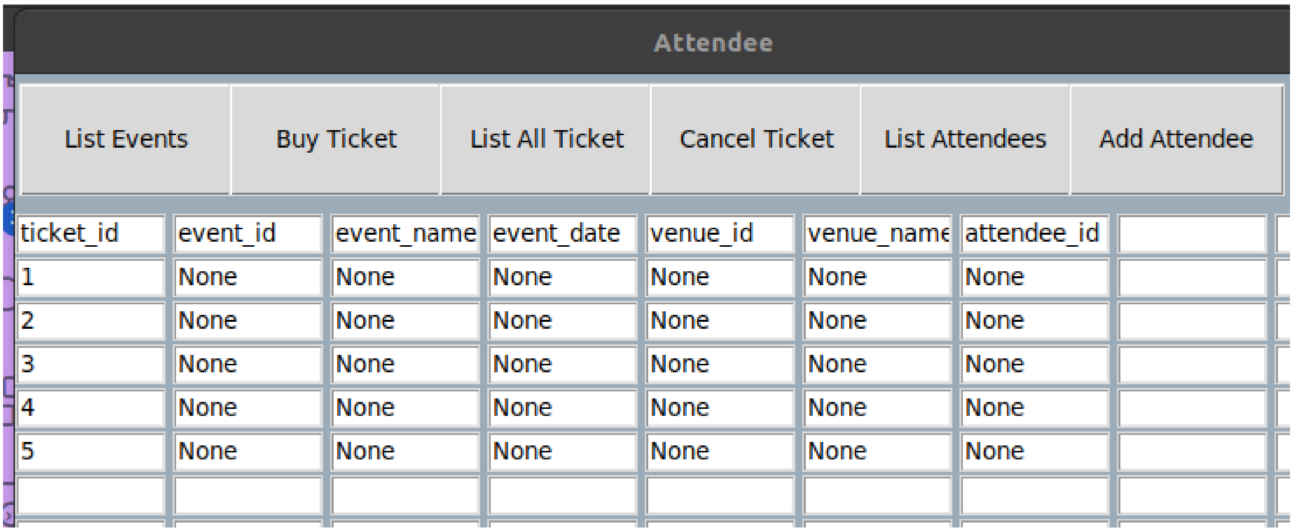
ticket_id	event_id	event_name	event_date	venue_id	venue_name	attendee_id
1	None	None	None	None	None	None
2	None	None	None	None	None	None
3	None	None	None	None	None	None
4	None	None	None	None	None	None
5	None	None	None	None	None	None
6	9	EVENT 99	2023-10-10	2	Room 2	3

After:



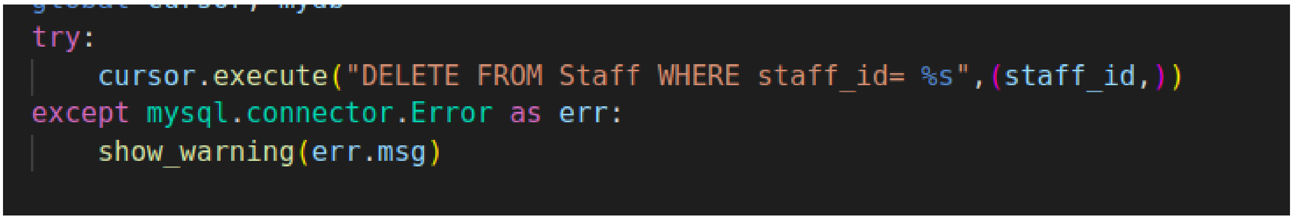
Attendee ID:
3

Ticket ID:
6



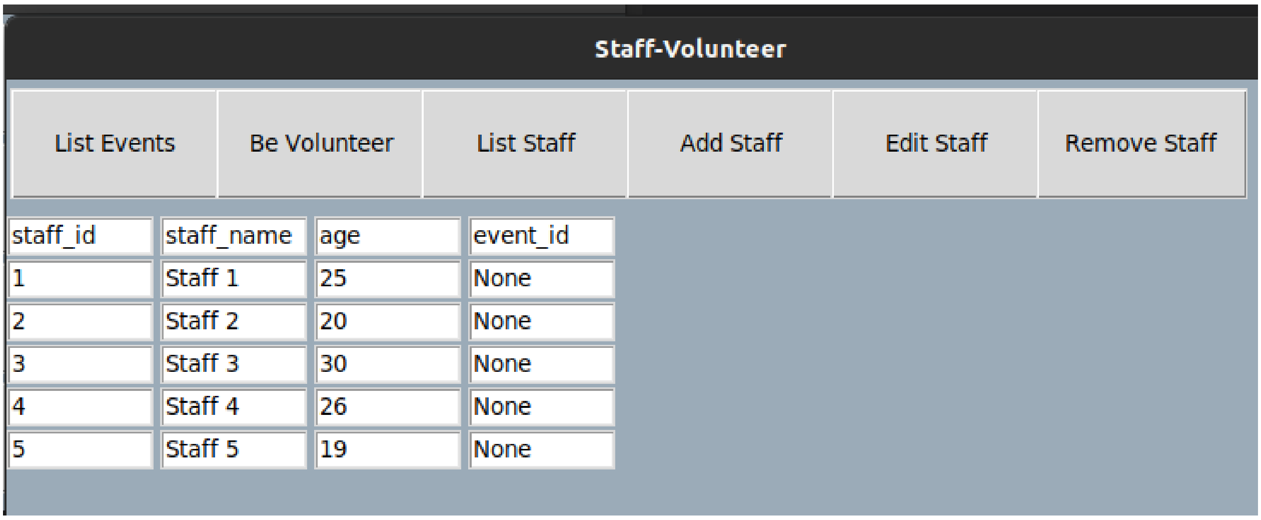
ticket_id	event_id	event_name	event_date	venue_id	venue_name	attendee_id
1	None	None	None	None	None	None
2	None	None	None	None	None	None
3	None	None	None	None	None	None
4	None	None	None	None	None	None
5	None	None	None	None	None	None

- DELETE id FROM Staff



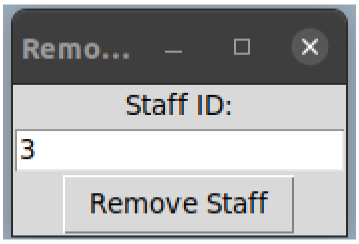
```
global cursor, mydb
try:
    cursor.execute("DELETE FROM Staff WHERE staff_id= %s",(staff_id,))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:

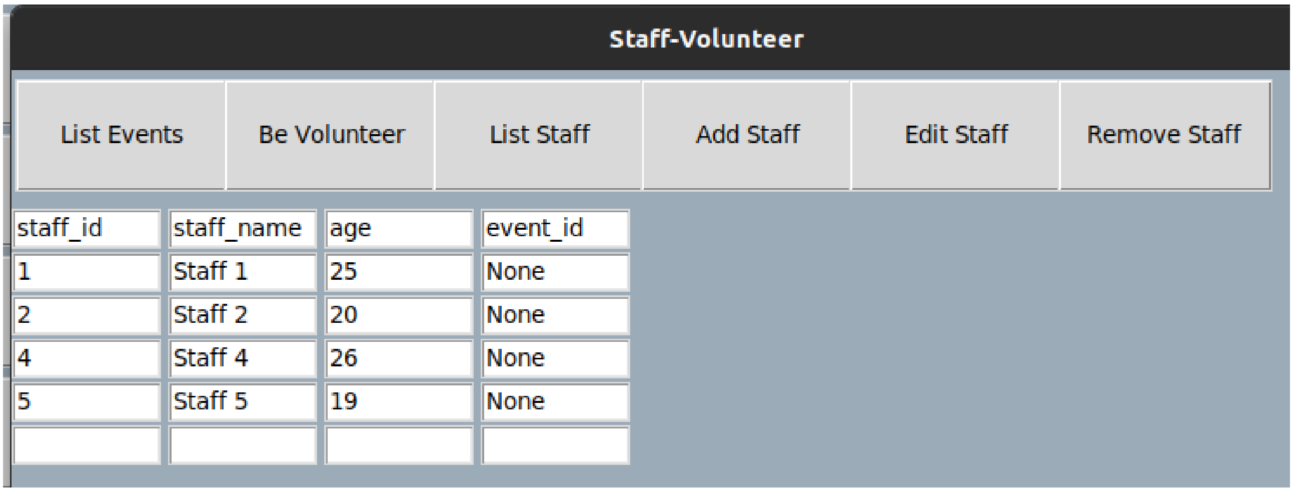


staff_id	staff_name	age	event_id
1	Staff 1	25	None
2	Staff 2	20	None
3	Staff 3	30	None
4	Staff 4	26	None
5	Staff 5	19	None

After:

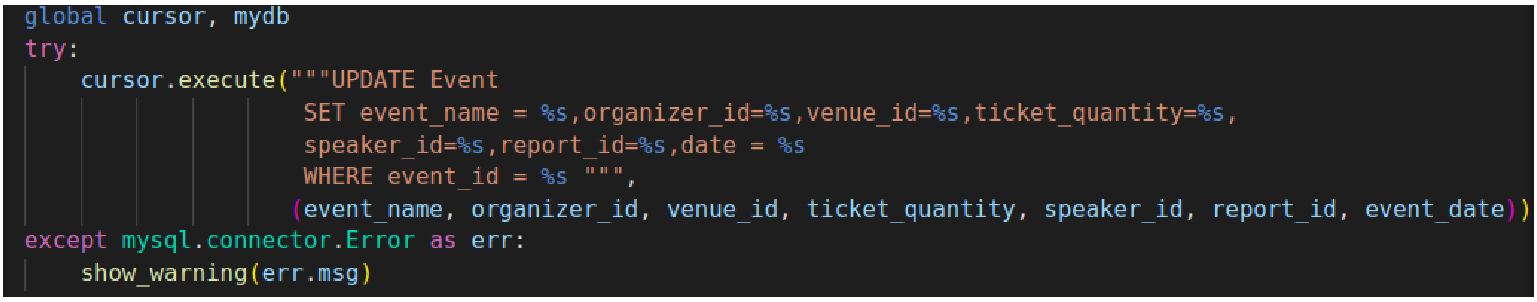


Staff ID: 3



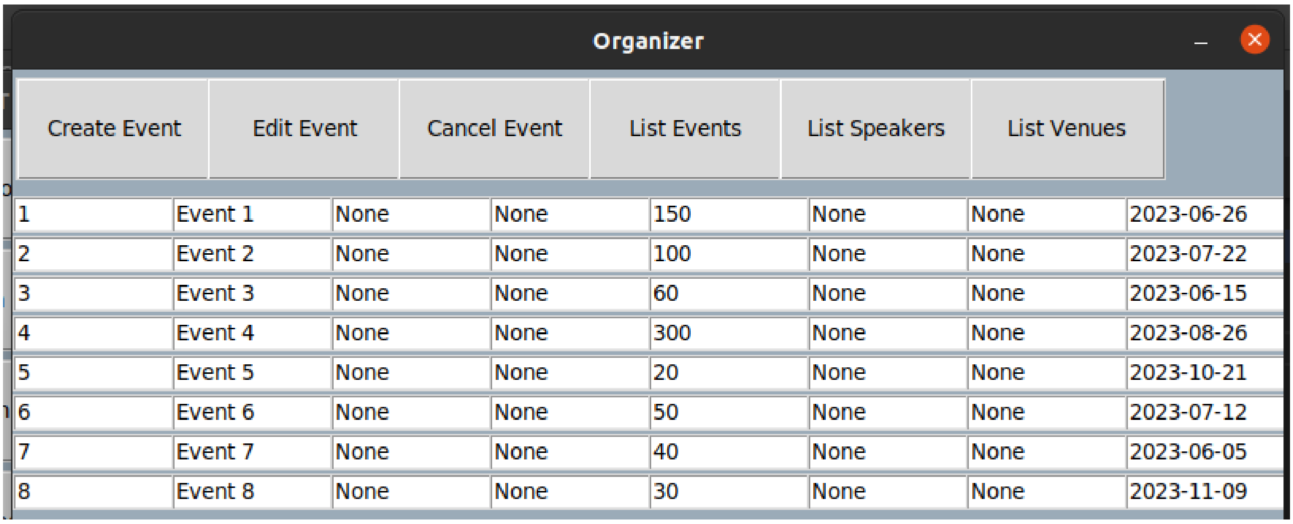
staff_id	staff_name	age	event_id
1	Staff 1	25	None
2	Staff 2	20	None
4	Staff 4	26	None
5	Staff 5	19	None

- UPDATE
    - UPDATE Event



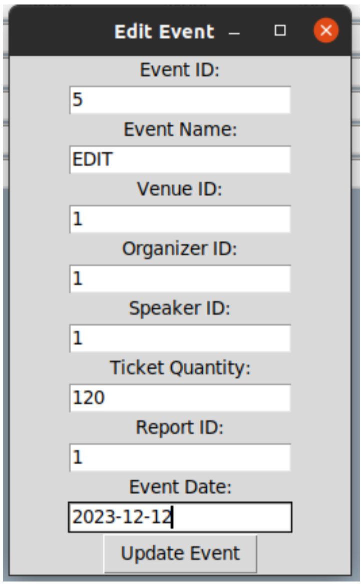
```
global cursor, mydb
try:
    cursor.execute("""UPDATE Event
                    SET event_name = %s,organizer_id=%s,venue_id=%s,ticket_quantity=%s,
                    speaker_id=%s,report_id=%s,date = %s
                    WHERE event_id = %s """,
                   (event_name, organizer_id, venue_id, ticket_quantity, speaker_id, report_id, event_date))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:

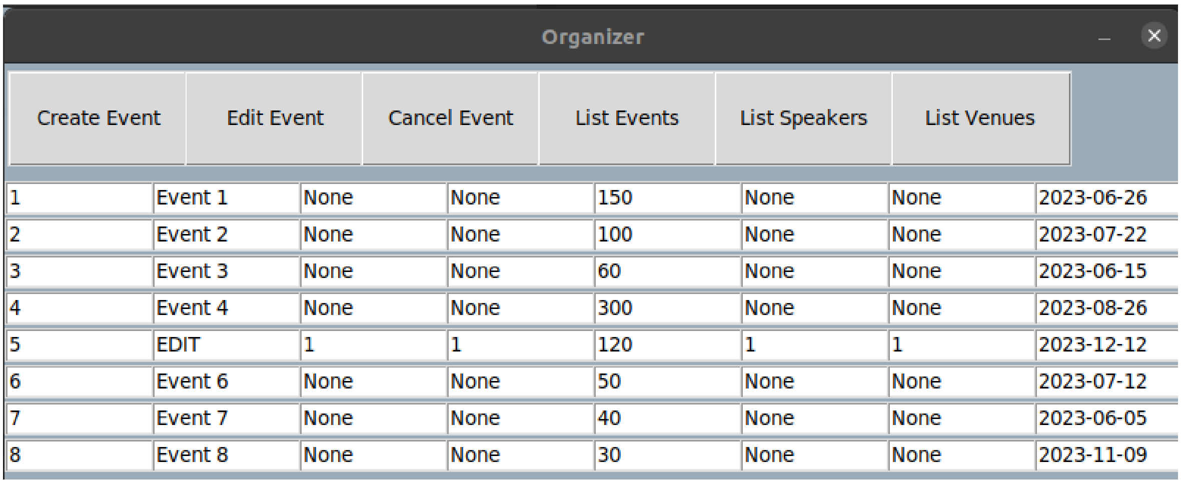


1	Event 1	None	None	150	None	None	2023-06-26
2	Event 2	None	None	100	None	None	2023-07-22
3	Event 3	None	None	60	None	None	2023-06-15
4	Event 4	None	None	300	None	None	2023-08-26
5	Event 5	None	None	20	None	None	2023-10-21
6	Event 6	None	None	50	None	None	2023-07-12
7	Event 7	None	None	40	None	None	2023-06-05
8	Event 8	None	None	30	None	None	2023-11-09

After:



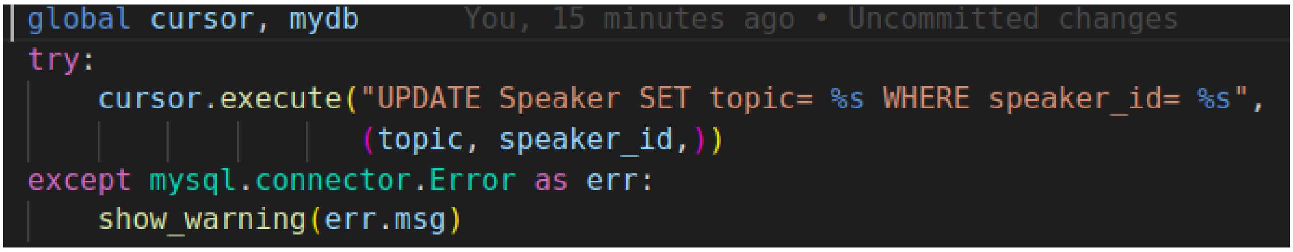
Event ID:
5
Event Name:
EDIT
Venue ID:
1
Organizer ID:
1
Speaker ID:
1
Ticket Quantity:
120
Report ID:
1
Event Date:
2023-12-12



1	Event 1	None	None	150	None	None	2023-06-26
2	Event 2	None	None	100	None	None	2023-07-22
3	Event 3	None	None	60	None	None	2023-06-15
4	Event 4	None	None	300	None	None	2023-08-26
5	EDIT	1	1	120	1	1	2023-12-12
6	Event 6	None	None	50	None	None	2023-07-12
7	Event 7	None	None	40	None	None	2023-06-05
8	Event 8	None	None	30	None	None	2023-11-09

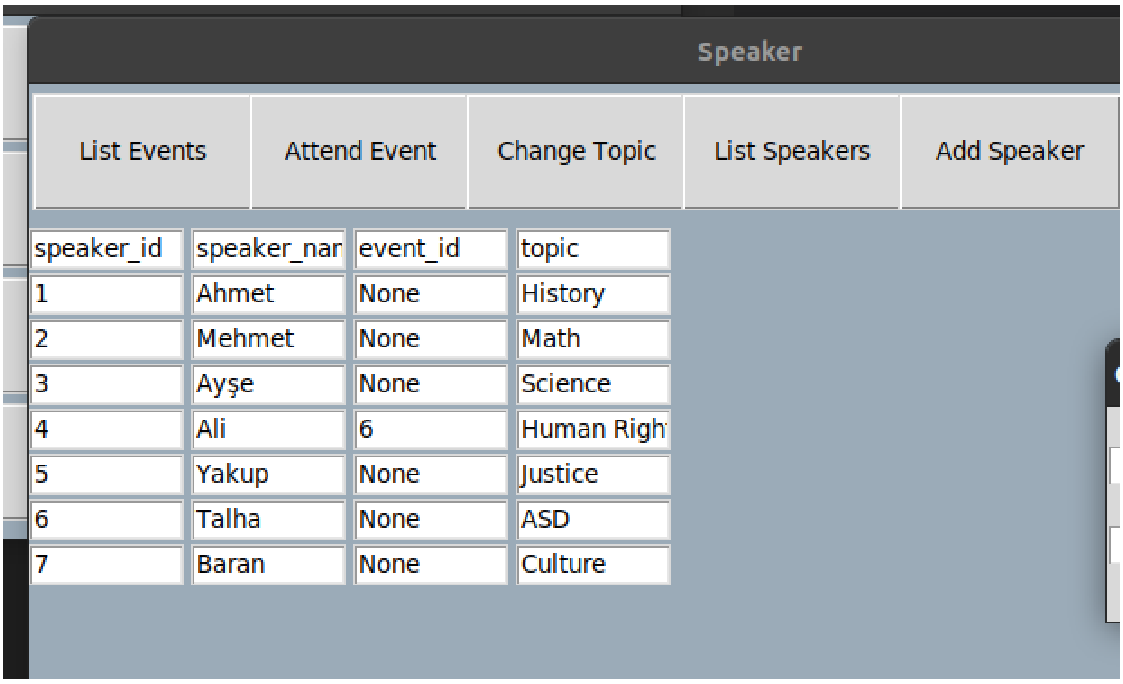
-id(5) updated.

- UPDATE Speaker



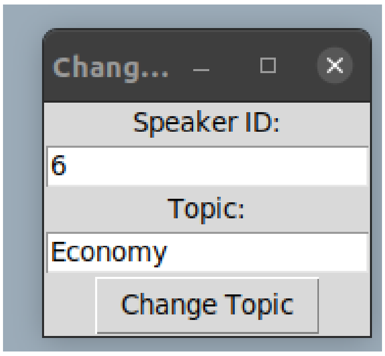
```
global cursor, mydb        You, 15 minutes ago • Uncommitted changes
try:
    cursor.execute("UPDATE Speaker SET topic= %s WHERE speaker_id= %s",
                   (topic, speaker_id,))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:

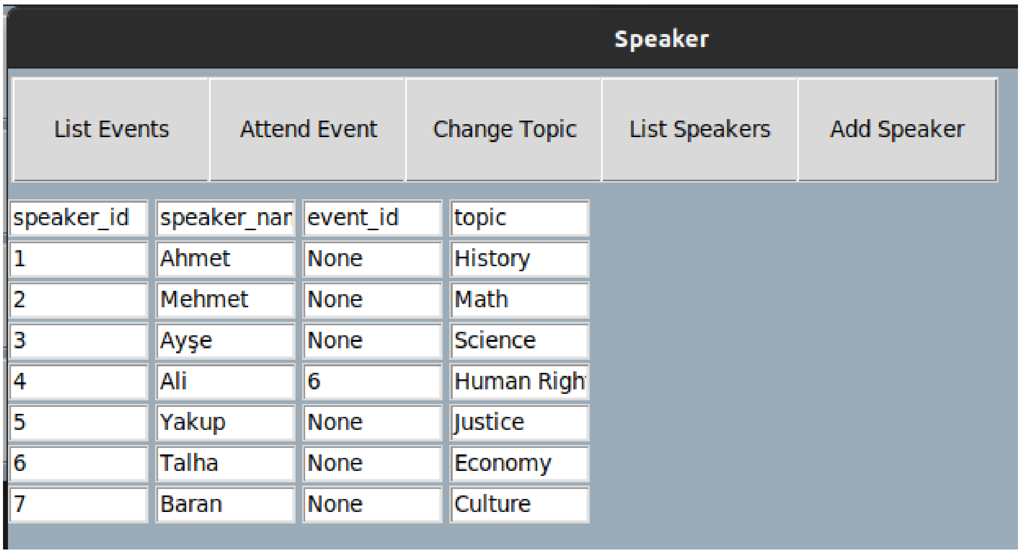


speaker_id	speaker_nar	event_id	topic
1	Ahmet	None	History
2	Mehmet	None	Math
3	Ayşe	None	Science
4	Ali	6	Human Righ
5	Yakup	None	Justice
6	Talha	None	ASD
7	Baran	None	Culture

After:

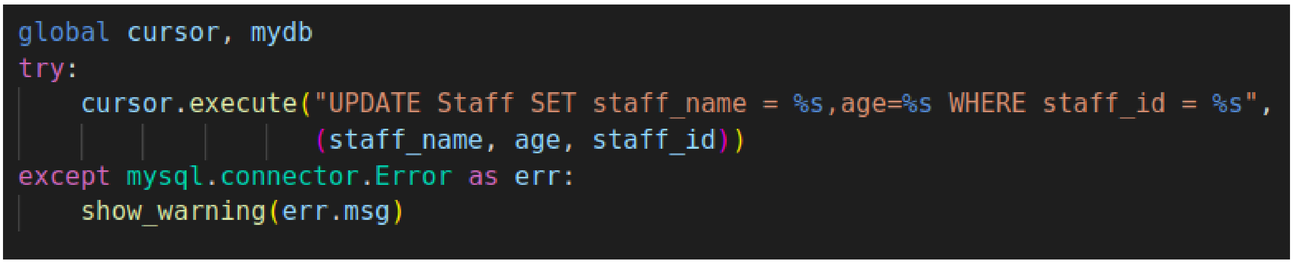


Speaker ID:
6
Topic:
Economy



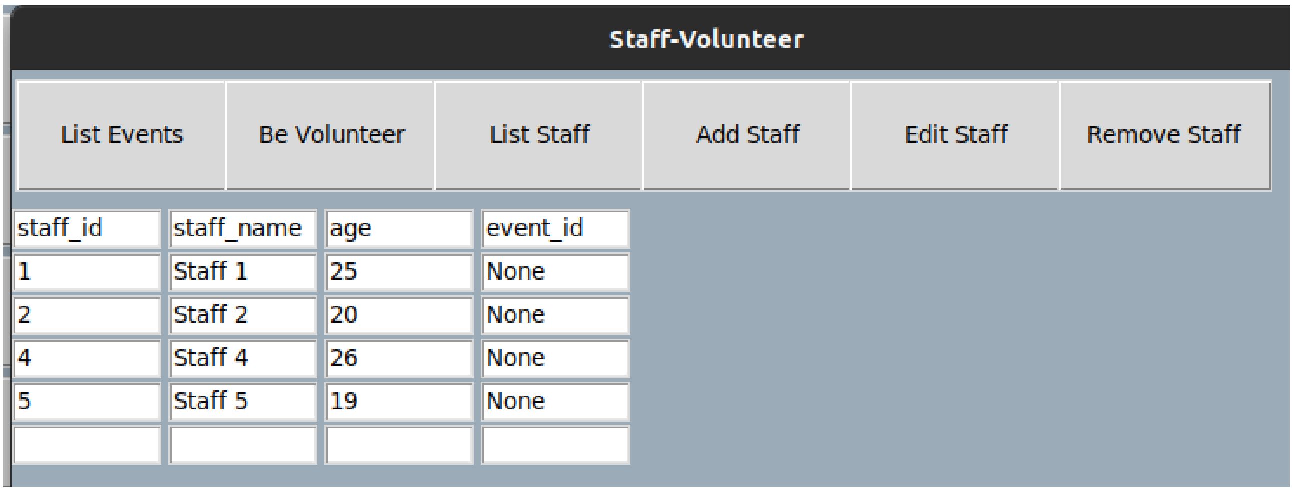
speaker_id	speaker_nar	event_id	topic
1	Ahmet	None	History
2	Mehmet	None	Math
3	Ayşe	None	Science
4	Ali	6	Human Righ
5	Yakup	None	Justice
6	Talha	None	Economy
7	Baran	None	Culture

- ○ UPDATE Staff



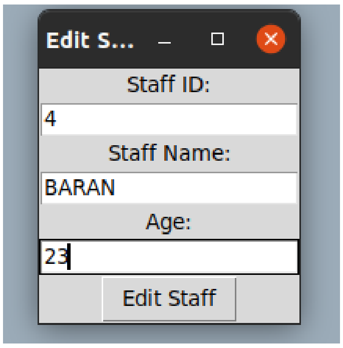
```
global cursor, mydb
try:
    cursor.execute("UPDATE Staff SET staff_name = %s,age=%s WHERE staff_id = %s",
                   (staff_name, age, staff_id))
except mysql.connector.Error as err:
    show_warning(err.msg)
```

Before:



staff_id	staff_name	age	event_id
1	Staff 1	25	None
2	Staff 2	20	None
4	Staff 4	26	None
5	Staff 5	19	None

After:



Staff ID:

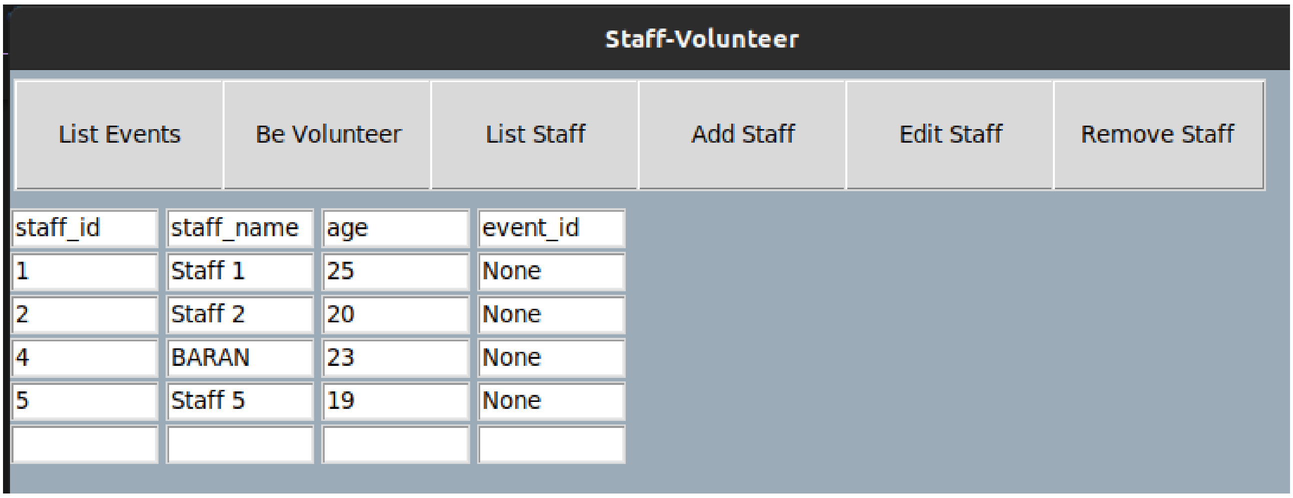
4

Staff Name:

BARAN

Age:

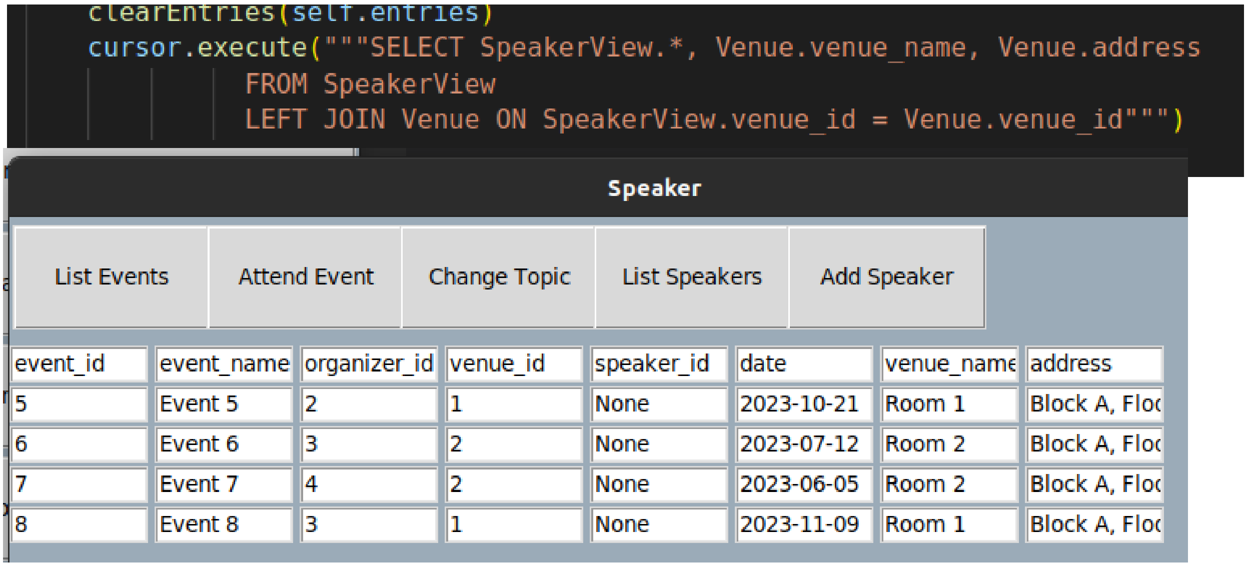
23



staff_id	staff_name	age	event_id
1	Staff 1	25	None
2	Staff 2	20	None
4	BARAN	23	None
5	Staff 5	19	None

9) JOIN

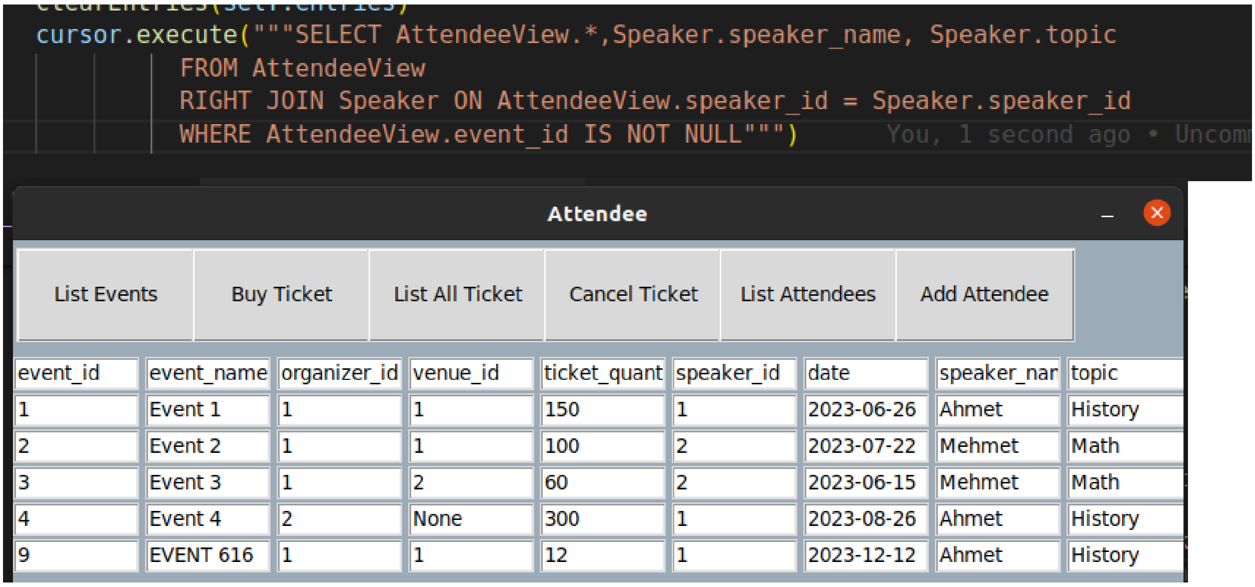
- LEFT JOIN



```
    clearEntries(self.entries)
    cursor.execute("""SELECT SpeakerView.*, Venue.venue_name, Venue.address
                FROM SpeakerView
                LEFT JOIN Venue ON SpeakerView.venue_id = Venue.venue_id""")
```

event_id	event_name	organizer_id	venue_id	speaker_id	date	venue_name	address
5	Event 5	2	1	None	2023-10-21	Room 1	Block A, Floo
6	Event 6	3	2	None	2023-07-12	Room 2	Block A, Floo
7	Event 7	4	2	None	2023-06-05	Room 2	Block A, Floo
8	Event 8	3	1	None	2023-11-09	Room 1	Block A, Floo

- RIGHT JOIN

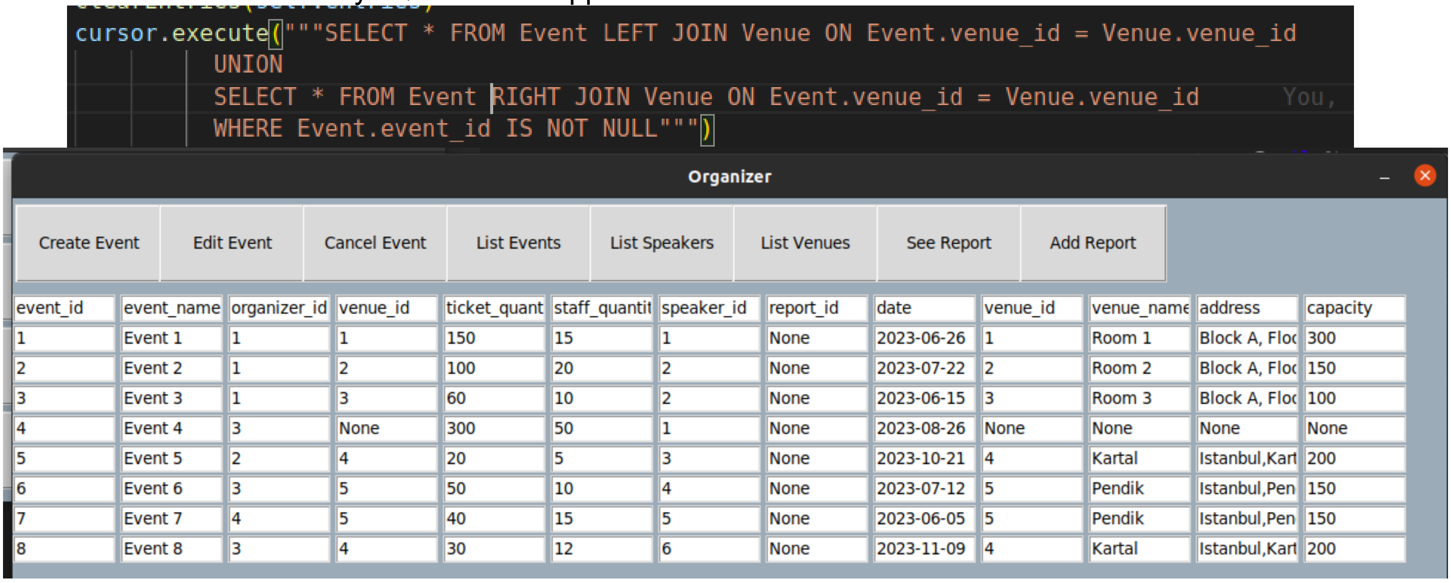


```
clearEntries(self.entries)
cursor.execute("""SELECT AttendeeView.*,Speaker.speaker_name, Speaker.topic
                FROM AttendeeView
                RIGHT JOIN Speaker ON AttendeeView.speaker_id = Speaker.speaker_id
                WHERE AttendeeView.event_id IS NOT NULL""")      You, 1 second ago • Uncom
```

event_id	event_name	organizer_id	venue_id	ticket_quant	speaker_id	date	speaker_nar	topic
1	Event 1	1	1	150	1	2023-06-26	Ahmet	History
2	Event 2	1	1	100	2	2023-07-22	Mehmet	Math
3	Event 3	1	2	60	2	2023-06-15	Mehmet	Math
4	Event 4	2	None	300	1	2023-08-26	Ahmet	History
9	EVENT 616	1	1	12	1	2023-12-12	Ahmet	History

- FULL JOIN

MySQL doesn't support FULL JOIN.



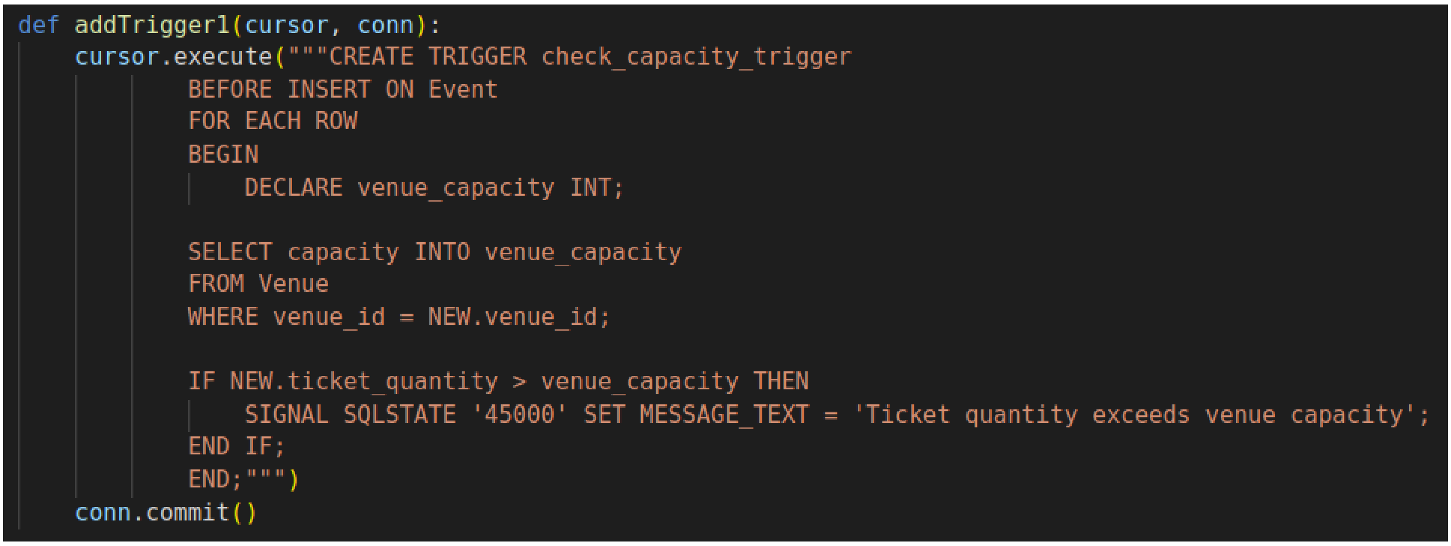
```
cursor.execute("""SELECT * FROM Event LEFT JOIN Venue ON Event.venue_id = Venue.venue_id
          UNION
          SELECT * FROM Event RIGHT JOIN Venue ON Event.venue_id = Venue.venue_id      You,
          WHERE Event.event_id IS NOT NULL""")
```

event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	report_id	date	venue_id	venue_name	address	capacity
1	Event 1	1	1	150	15	1	None	2023-06-26	1	Room 1	Block A, Floo	300
2	Event 2	1	2	100	20	2	None	2023-07-22	2	Room 2	Block A, Floo	150
3	Event 3	1	3	60	10	2	None	2023-06-15	3	Room 3	Block A, Floo	100
4	Event 4	3	None	300	50	1	None	2023-08-26	None	None	None	None
5	Event 5	2	4	20	5	3	None	2023-10-21	4	Kartal	Istanbul,Kart	200
6	Event 6	3	5	50	10	4	None	2023-07-12	5	Pendik	Istanbul,Pen	150
7	Event 7	4	5	40	15	5	None	2023-06-05	5	Pendik	Istanbul,Pen	150
8	Event 8	3	4	30	12	6	None	2023-11-09	4	Kartal	Istanbul,Kart	200

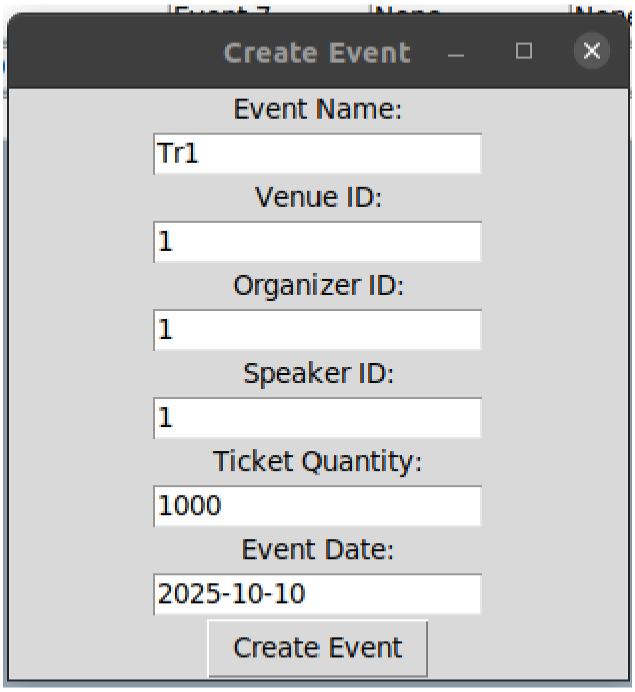
10) TRIGGERS

- Trigger 1: Venue Capacity - Ticket Quantity

  Controlling venue capacity is enough for the event's ticket quantity.



```
def addTrigger1(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_capacity_trigger
            BEFORE INSERT ON Event
            FOR EACH ROW
            BEGIN
                DECLARE venue_capacity INT;

            SELECT capacity INTO venue_capacity
            FROM Venue
            WHERE venue_id = NEW.venue_id;

            IF NEW.ticket_quantity > venue_capacity THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ticket quantity exceeds venue capacity';
            END IF;
            END;""")
    conn.commit()
```



Create Event

Event Name: Tr1

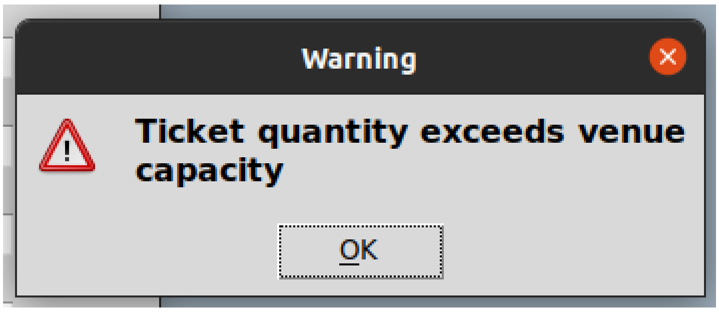
Venue ID: 1

Organizer ID: 1

Speaker ID: 1

Ticket Quantity: 1000

Event Date: 2025-10-10

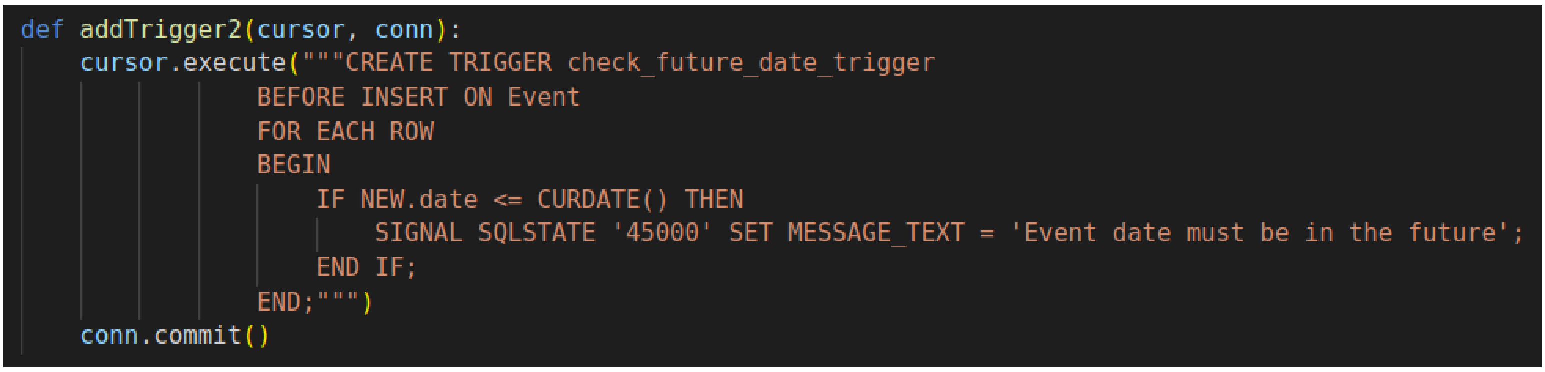


Warning

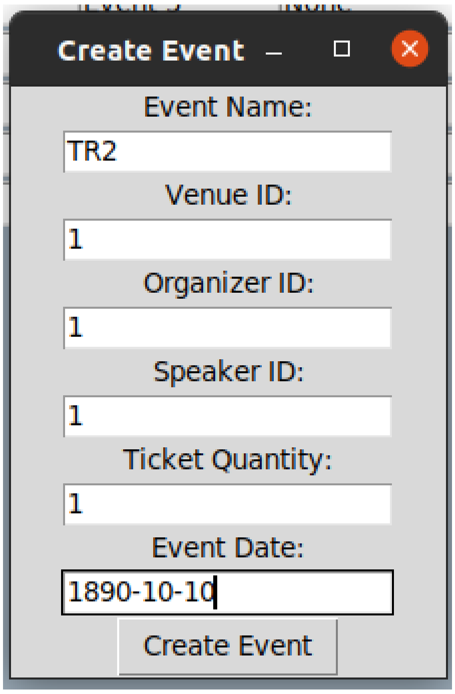
Ticket quantity exceeds venue capacity

- Trigger 2: New Event Date

  New event date must be in the future.



```
def addTrigger2(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_future_date_trigger
            BEFORE INSERT ON Event
            FOR EACH ROW
            BEGIN
                IF NEW.date <= CURDATE() THEN
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event date must be in the future';
                END IF;
            END;""")
    conn.commit()
```



Event Name: TR2

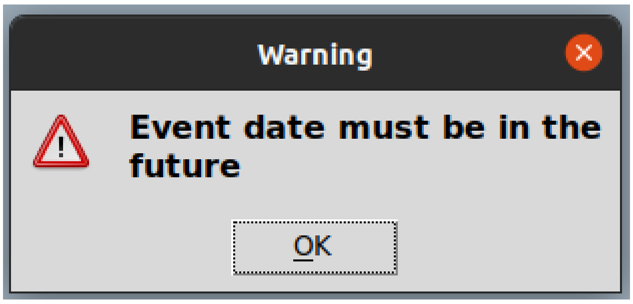
Venue ID: 1

Organizer ID: 1

Speaker ID: 1

Ticket Quantity: 1

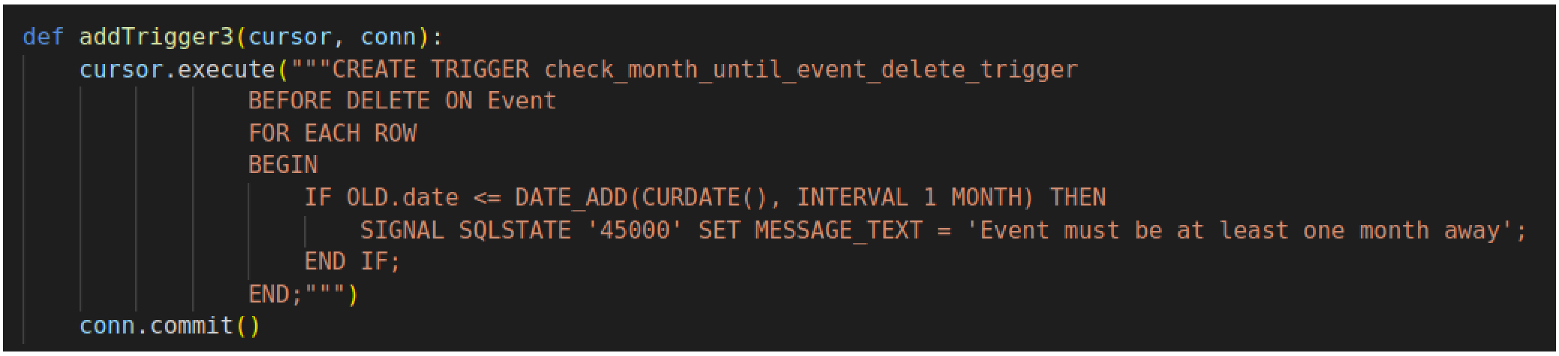
Event Date: 1890-10-10



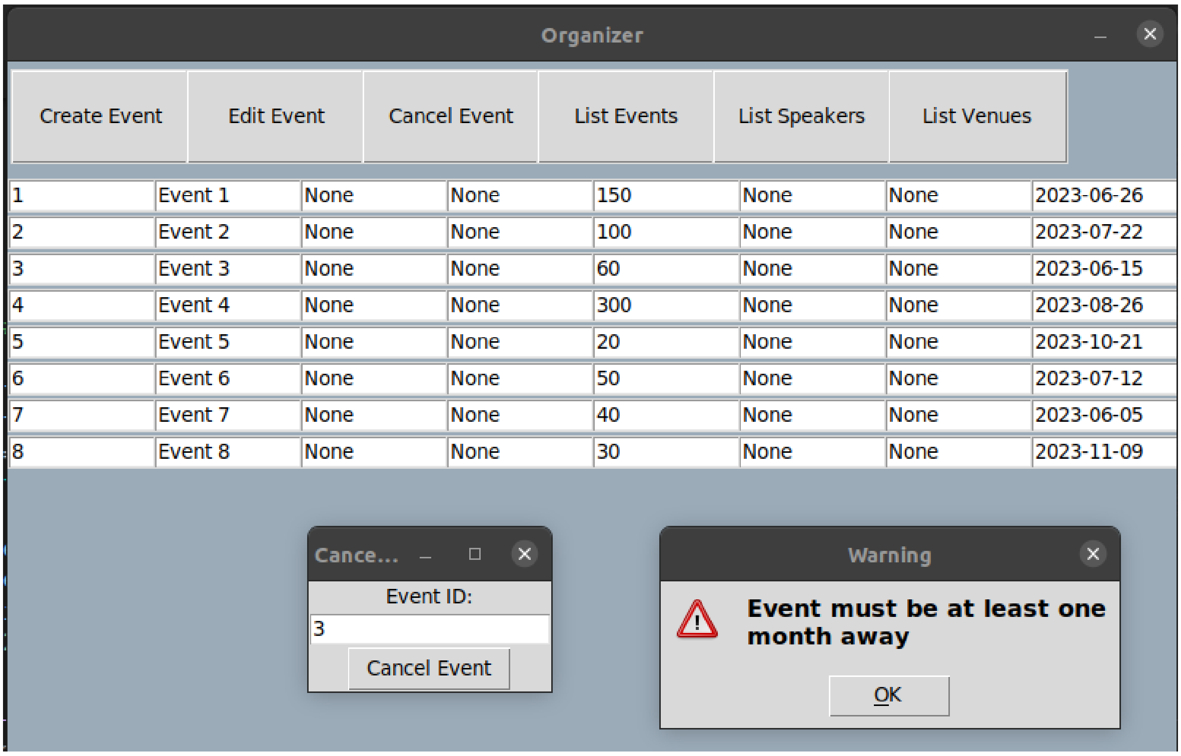
Event date must be in the future

- Trigger 3: Near Future Event

  Events closer than 1 month cannot be canceled.



```
def addTrigger3(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_month_until_event_delete_trigger
            BEFORE DELETE ON Event
            FOR EACH ROW
            BEGIN
                IF OLD.date <= DATE_ADD(CURDATE(), INTERVAL 1 MONTH) THEN
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Event must be at least one month away';
                END IF;
            END;""")
    conn.commit()
```



1	Event 1	None	None	150	None	None	2023-06-26
2	Event 2	None	None	100	None	None	2023-07-22
3	Event 3	None	None	60	None	None	2023-06-15
4	Event 4	None	None	300	None	None	2023-08-26
5	Event 5	None	None	20	None	None	2023-10-21
6	Event 6	None	None	50	None	None	2023-07-12
7	Event 7	None	None	40	None	None	2023-06-05
8	Event 8	None	None	30	None	None	2023-11-09

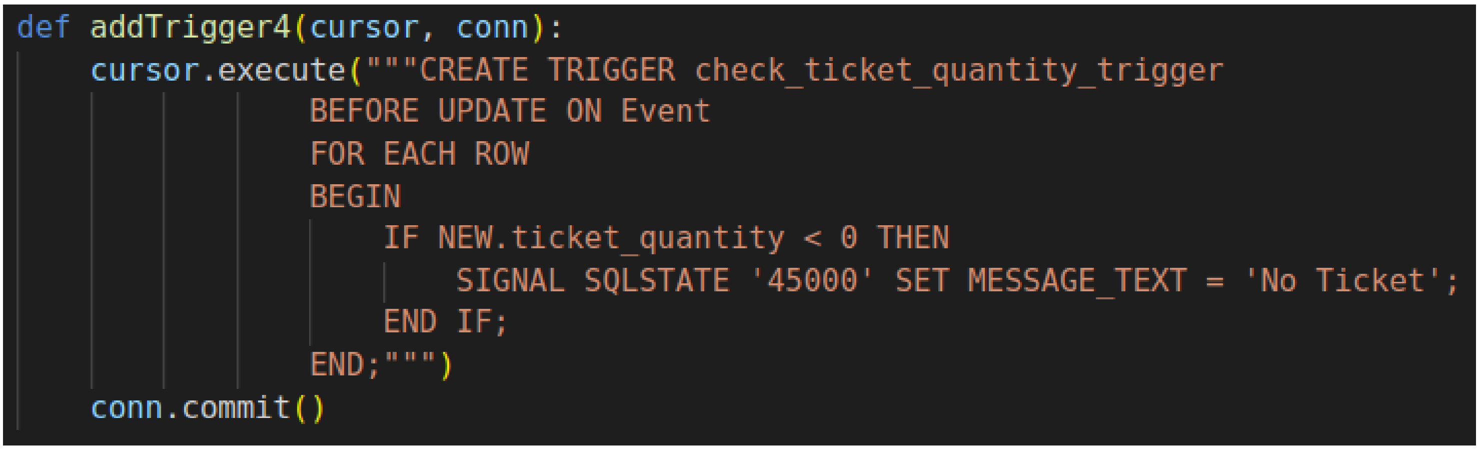
Event ID: 3

Event must be at least one month away

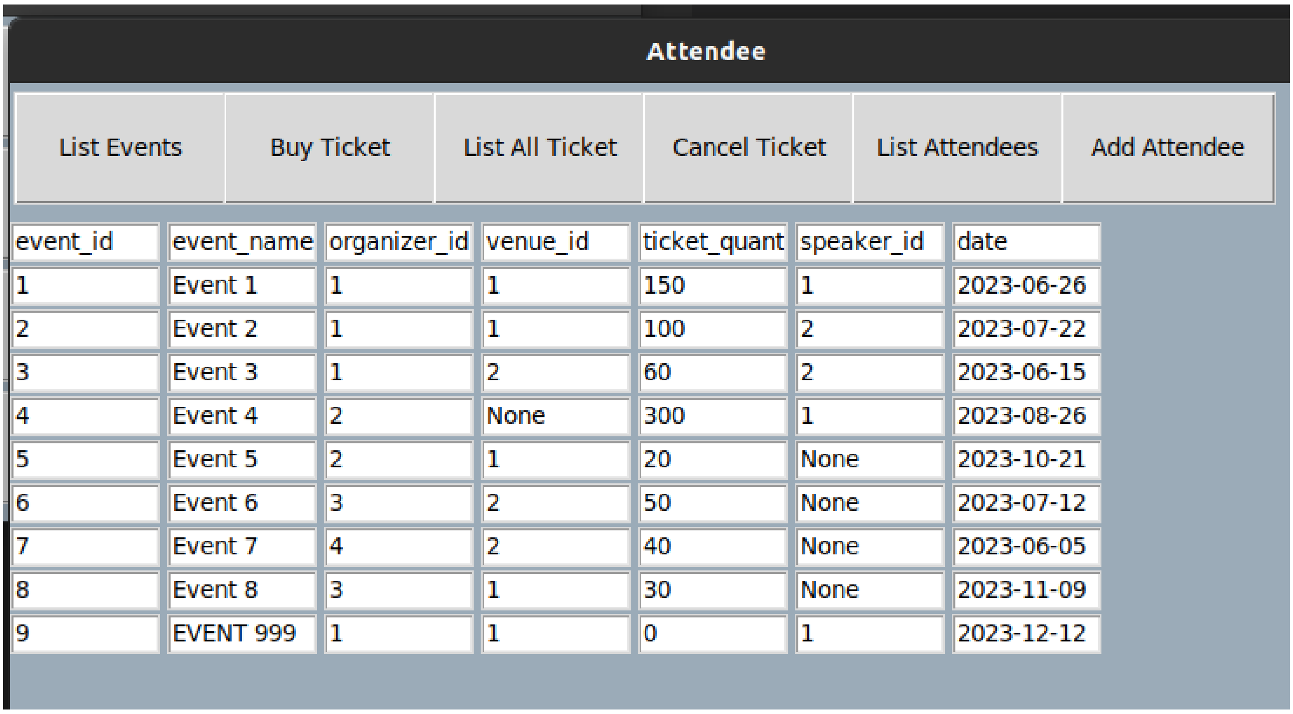
Test Date: 2023-06-10

- Trigger 4: Ticket Quantity

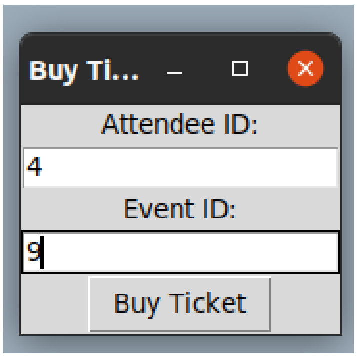
Can not buy a ticket if there is no ticket.



```
def addTrigger4(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_ticket_quantity_trigger
                BEFORE UPDATE ON Event
                FOR EACH ROW
                BEGIN
                    IF NEW.ticket_quantity < 0 THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No Ticket';
                    END IF;
                END;""")
    conn.commit()
```

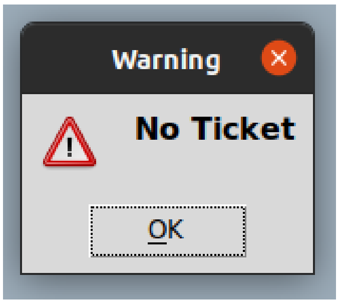


event_id	event_name	organizer_id	venue_id	ticket_quant	speaker_id	date
1	Event 1	1	1	150	1	2023-06-26
2	Event 2	1	1	100	2	2023-07-22
3	Event 3	1	2	60	2	2023-06-15
4	Event 4	2	None	300	1	2023-08-26
5	Event 5	2	1	20	None	2023-10-21
6	Event 6	3	2	50	None	2023-07-12
7	Event 7	4	2	40	None	2023-06-05
8	Event 8	3	1	30	None	2023-11-09
9	EVENT 999	1	1	0	1	2023-12-12



Attendee ID: 4

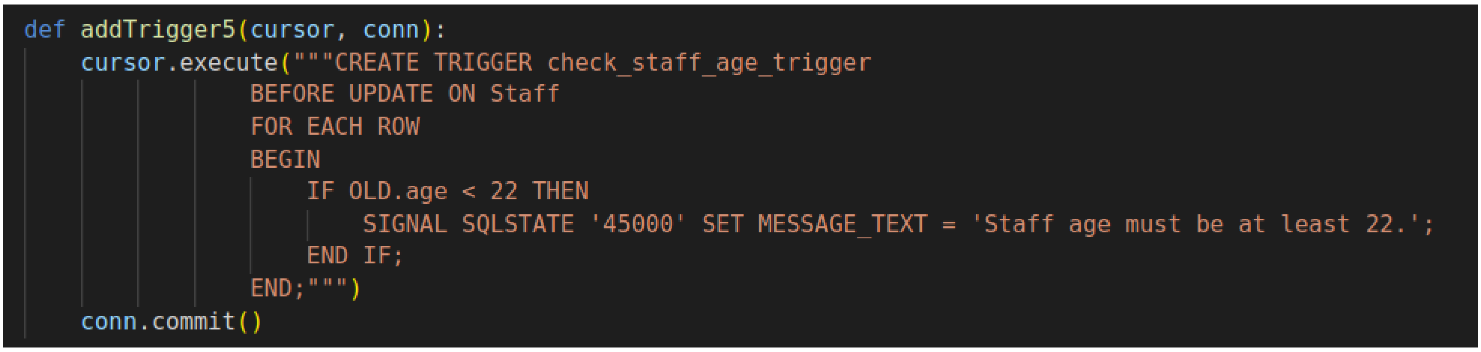
Event ID: 9



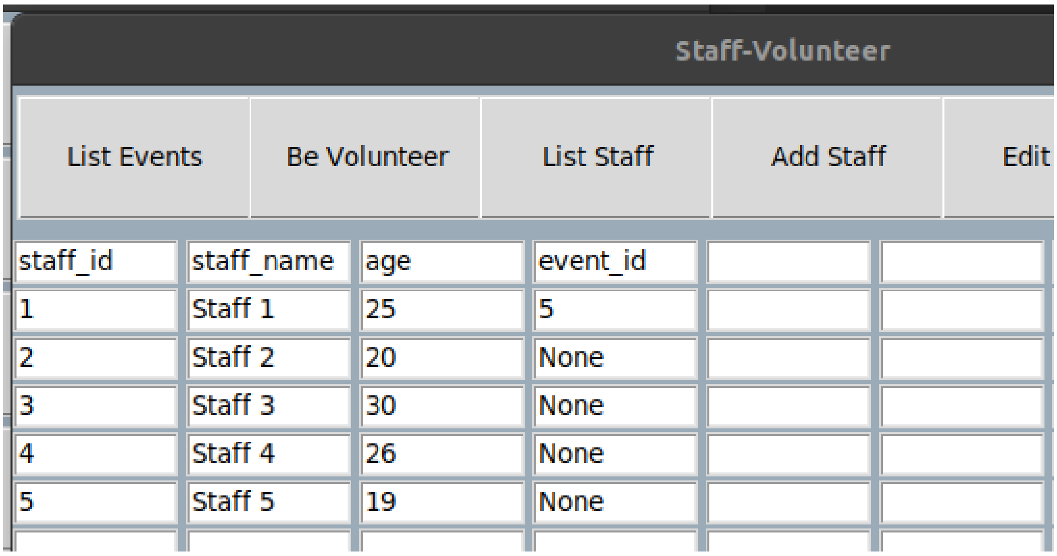
No Ticket

- Trigger 5: Volunteering Age

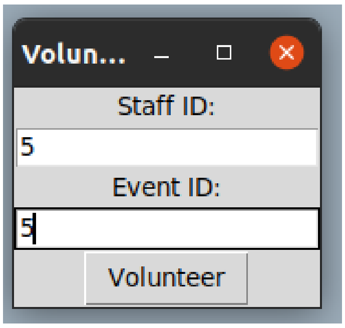
  Staff must be over 22 years of age to volunteer.



```
def addTrigger5(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_staff_age_trigger
            BEFORE UPDATE ON Staff
            FOR EACH ROW
            BEGIN
                IF OLD.age < 22 THEN
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Staff age must be at least 22.';
                END IF;
            END;""")
    conn.commit()
```

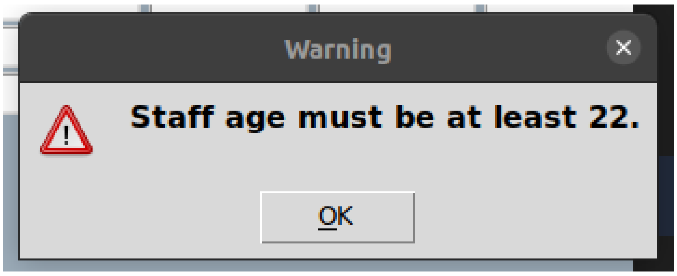


staff_id	staff_name	age	event_id		
1	Staff 1	25	5		
2	Staff 2	20	None		
3	Staff 3	30	None		
4	Staff 4	26	None		
5	Staff 5	19	None		



Staff ID: 5

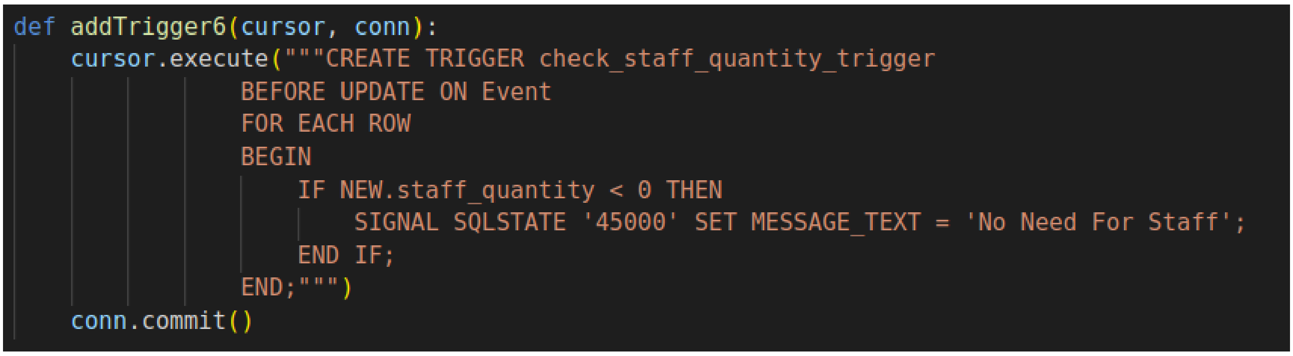
Event ID: 5



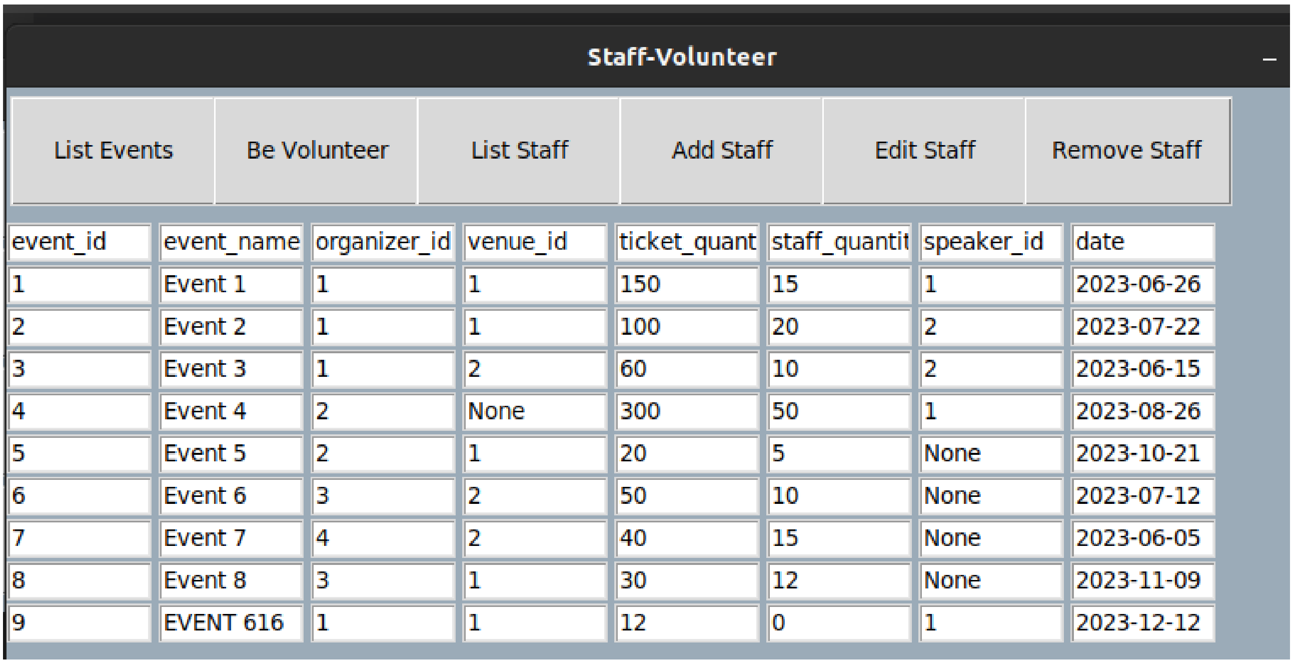
Staff age must be at least 22.

- Trigger 6: Staff Quantity

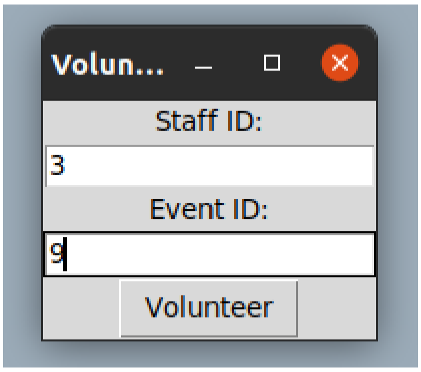
  Staff can’t volunteer if there is no need for staff for that event.



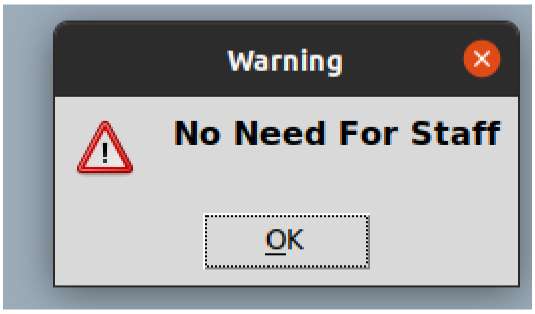
```
def addTrigger6(cursor, conn):
    cursor.execute("""CREATE TRIGGER check_staff_quantity_trigger
                BEFORE UPDATE ON Event
                FOR EACH ROW
                BEGIN
                    IF NEW.staff_quantity < 0 THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No Need For Staff';
                    END IF;
                END;""")
    conn.commit()
```



event_id	event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	date
1	Event 1	1	1	150	15	1	2023-06-26
2	Event 2	1	1	100	20	2	2023-07-22
3	Event 3	1	2	60	10	2	2023-06-15
4	Event 4	2	None	300	50	1	2023-08-26
5	Event 5	2	1	20	5	None	2023-10-21
6	Event 6	3	2	50	10	None	2023-07-12
7	Event 7	4	2	40	15	None	2023-06-05
8	Event 8	3	1	30	12	None	2023-11-09
9	EVENT 616	1	1	12	0	1	2023-12-12



Staff ID:
3
Event ID:
9

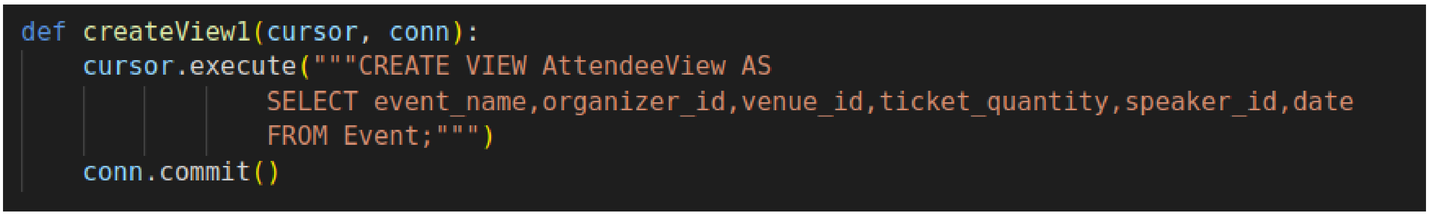


No Need For Staff

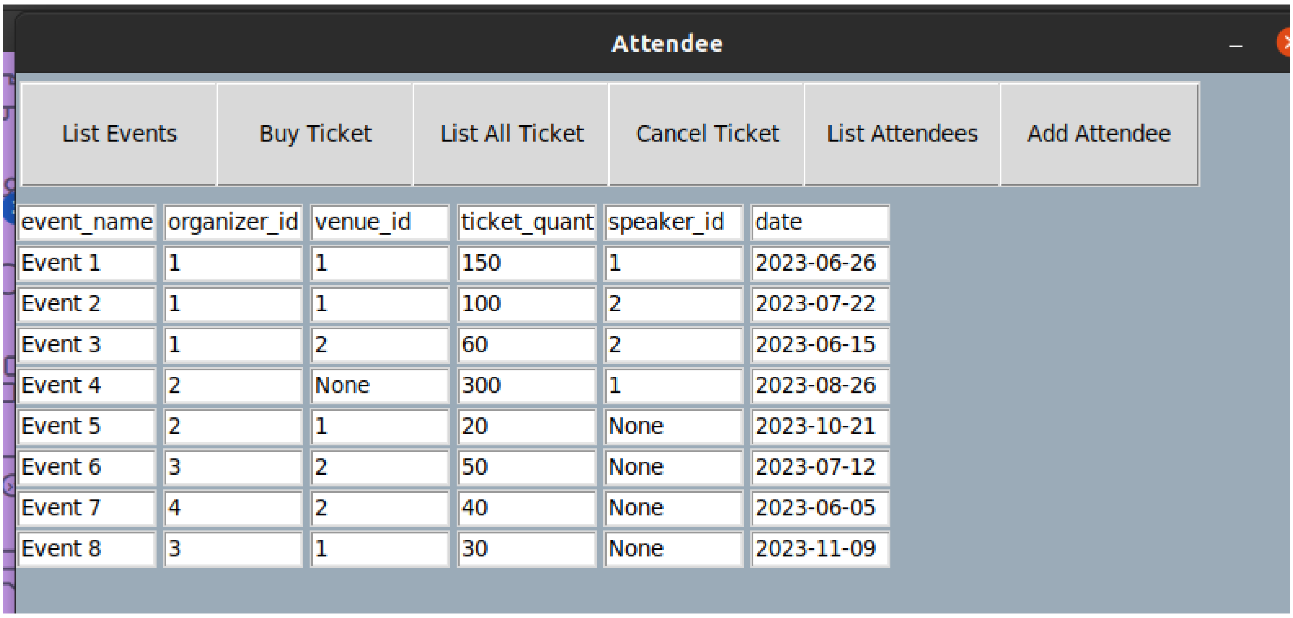
11) VIEWS

- View 1: Attendee View

Attendees can’t see report_id and staff_quantity.



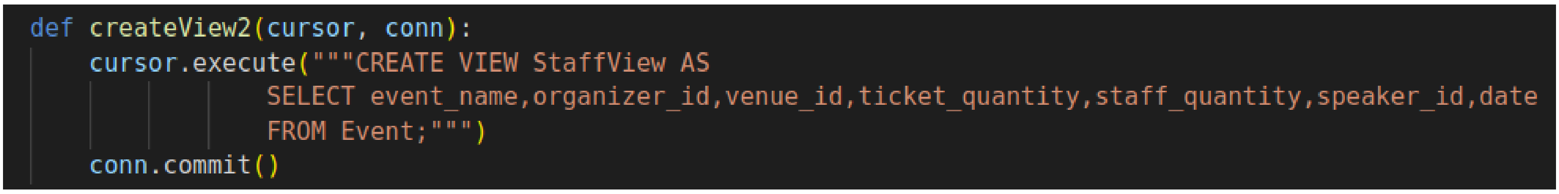
```
def createView1(cursor, conn):
    cursor.execute("""CREATE VIEW AttendeeView AS
                SELECT event_name,organizer_id,venue_id,ticket_quantity,speaker_id,date
                FROM Event;""")
    conn.commit()
```



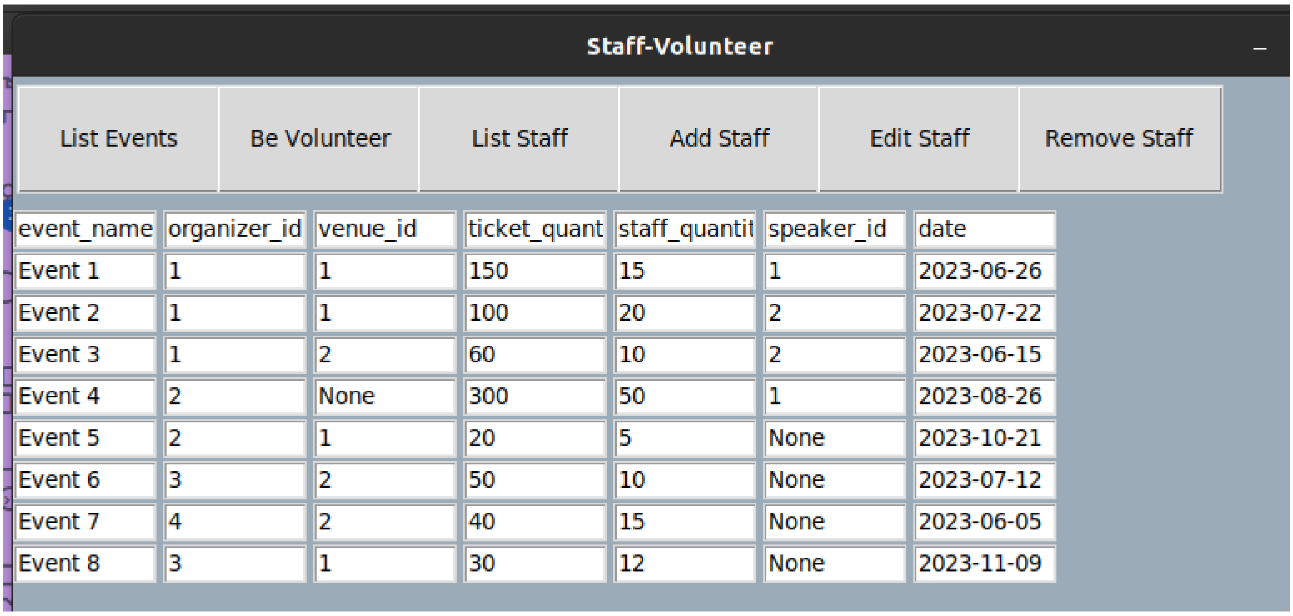
event_name	organizer_id	venue_id	ticket_quant	speaker_id	date
Event 1	1	1	150	1	2023-06-26
Event 2	1	1	100	2	2023-07-22
Event 3	1	2	60	2	2023-06-15
Event 4	2	None	300	1	2023-08-26
Event 5	2	1	20	None	2023-10-21
Event 6	3	2	50	None	2023-07-12
Event 7	4	2	40	None	2023-06-05
Event 8	3	1	30	None	2023-11-09

- View 2: Staff View

Staff can’t see report_id.



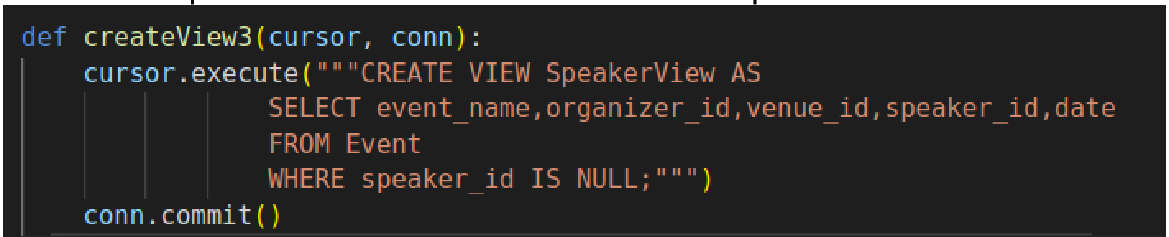
```
def createView2(cursor, conn):
    cursor.execute("""CREATE VIEW StaffView AS
                SELECT event_name,organizer_id,venue_id,ticket_quantity,staff_quantity,speaker_id,date
                FROM Event;""")
    conn.commit()
```



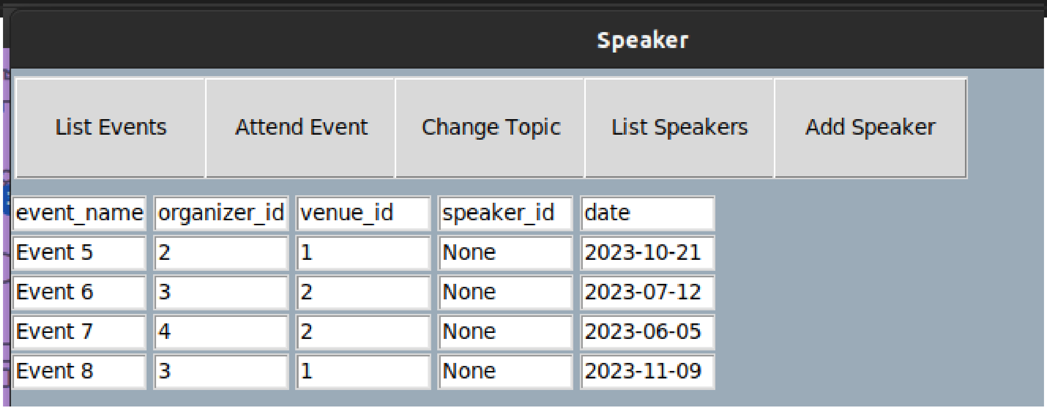
event_name	organizer_id	venue_id	ticket_quant	staff_quantit	speaker_id	date
Event 1	1	1	150	15	1	2023-06-26
Event 2	1	1	100	20	2	2023-07-22
Event 3	1	2	60	10	2	2023-06-15
Event 4	2	None	300	50	1	2023-08-26
Event 5	2	1	20	5	None	2023-10-21
Event 6	3	2	50	10	None	2023-07-12
Event 7	4	2	40	15	None	2023-06-05
Event 8	3	1	30	12	None	2023-11-09

- View 3: Speaker View

Speakers can’t see events that have speakers.

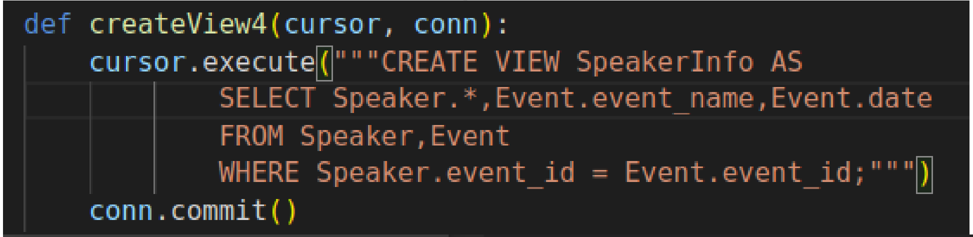


```
def createView3(cursor, conn):
    cursor.execute("""CREATE VIEW SpeakerView AS
                SELECT event_name,organizer_id,venue_id,speaker_id,date
                FROM Event
                WHERE speaker_id IS NULL;""")
    conn.commit()
```



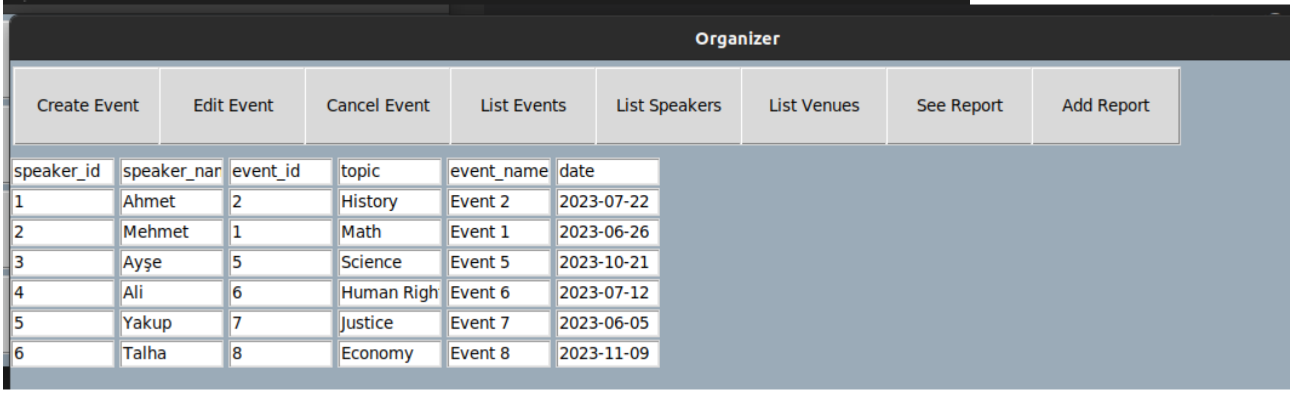
event_name	organizer_id	venue_id	speaker_id	date
Event 5	2	1	None	2023-10-21
Event 6	3	2	None	2023-07-12
Event 7	4	2	None	2023-06-05
Event 8	3	1	None	2023-11-09

- View 4: Speaker Info View



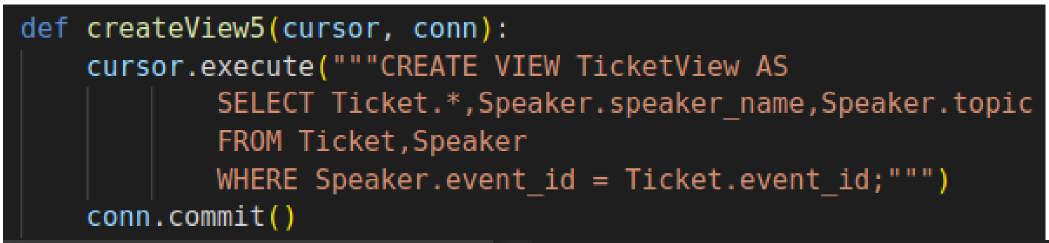
```
def createView4(cursor, conn):
    cursor.execute("""CREATE VIEW SpeakerInfo AS
            SELECT Speaker.*,Event.event_name,Event.date
            FROM Speaker,Event
            WHERE Speaker.event_id = Event.event_id;""")
    conn.commit()
```

Speakers can see other speakers events



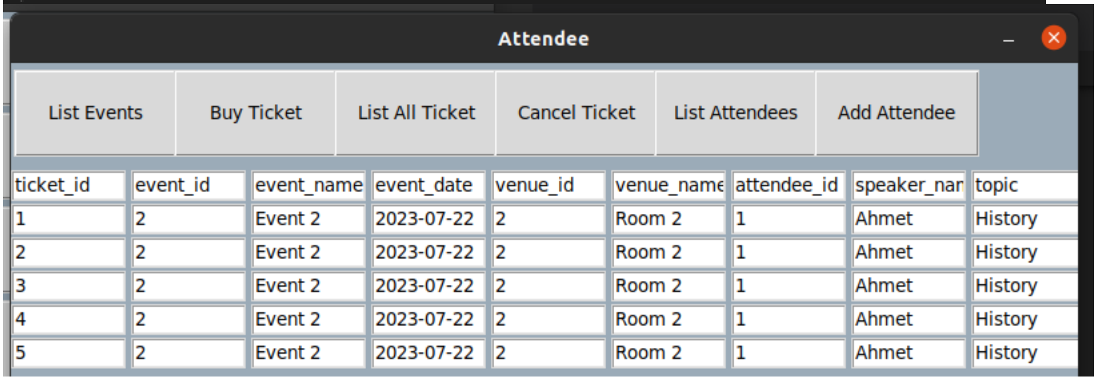
speaker_id	speaker_nar	event_id	topic	event_name	date
1	Ahmet	2	History	Event 2	2023-07-22
2	Mehmet	1	Math	Event 1	2023-06-26
3	Ayşe	5	Science	Event 5	2023-10-21
4	Ali	6	Human Righ	Event 6	2023-07-12
5	Yakup	7	Justice	Event 7	2023-06-05
6	Talha	8	Economy	Event 8	2023-11-09

- View 5: Ticket View



```
def createView5(cursor, conn):
    cursor.execute("""CREATE VIEW TicketView AS
            SELECT Ticket.*,Speaker.speaker_name,Speaker.topic
            FROM Ticket,Speaker
            WHERE Speaker.event_id = Ticket.event_id;""")
    conn.commit()
```

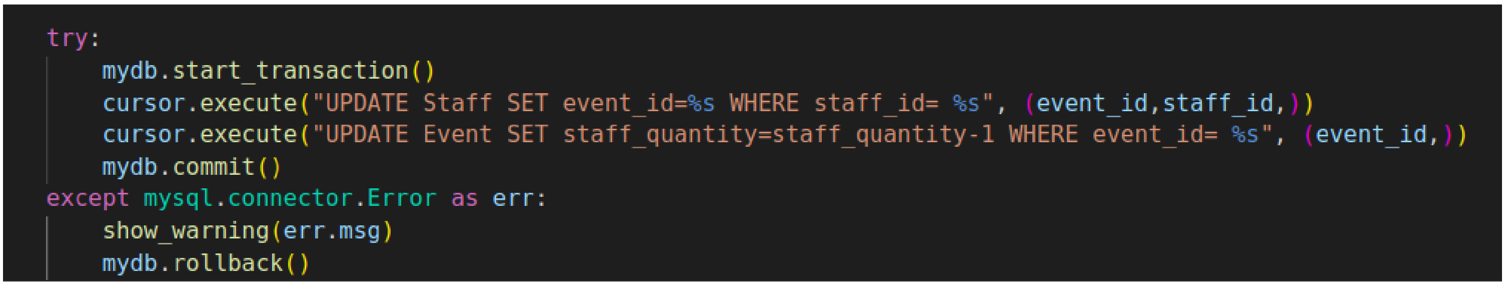
Attendees can see speaker’s name and topic



ticket_id	event_id	event_name	event_date	venue_id	venue_name	attendee_id	speaker_nar	topic
1	2	Event 2	2023-07-22	2	Room 2	1	Ahmet	History
2	2	Event 2	2023-07-22	2	Room 2	1	Ahmet	History
3	2	Event 2	2023-07-22	2	Room 2	1	Ahmet	History
4	2	Event 2	2023-07-22	2	Room 2	1	Ahmet	History
5	2	Event 2	2023-07-22	2	Room 2	1	Ahmet	History

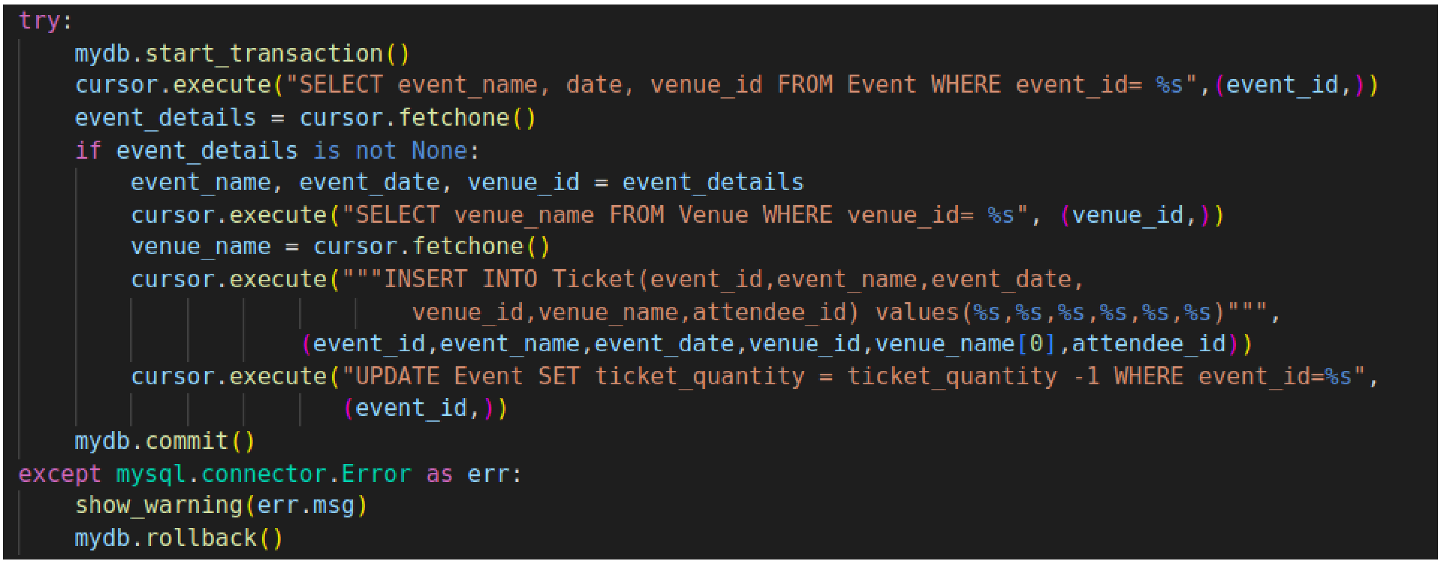
12) TRANSACTIONS

- Transaction 1: Volunteering



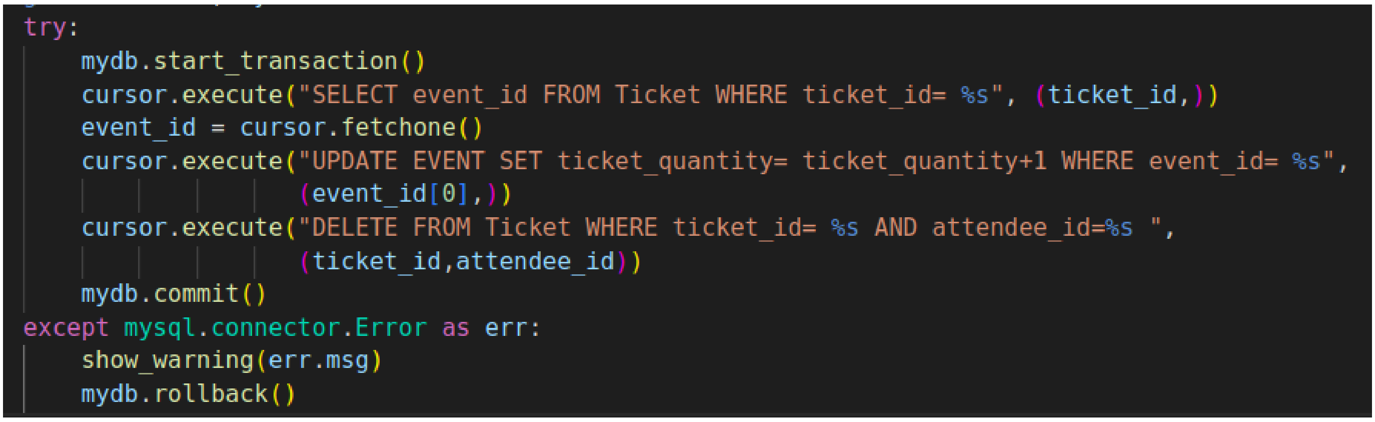
```
try:
    mydb.start_transaction()
    cursor.execute("UPDATE Staff SET event_id=%s WHERE staff_id= %s", (event_id,staff_id,))
    cursor.execute("UPDATE Event SET staff_quantity=staff_quantity-1 WHERE event_id= %s", (event_id,))
    mydb.commit()
except mysql.connector.Error as err:
    show_warning(err.msg)
    mydb.rollback()
```

- Transaction 2: Buying Ticket



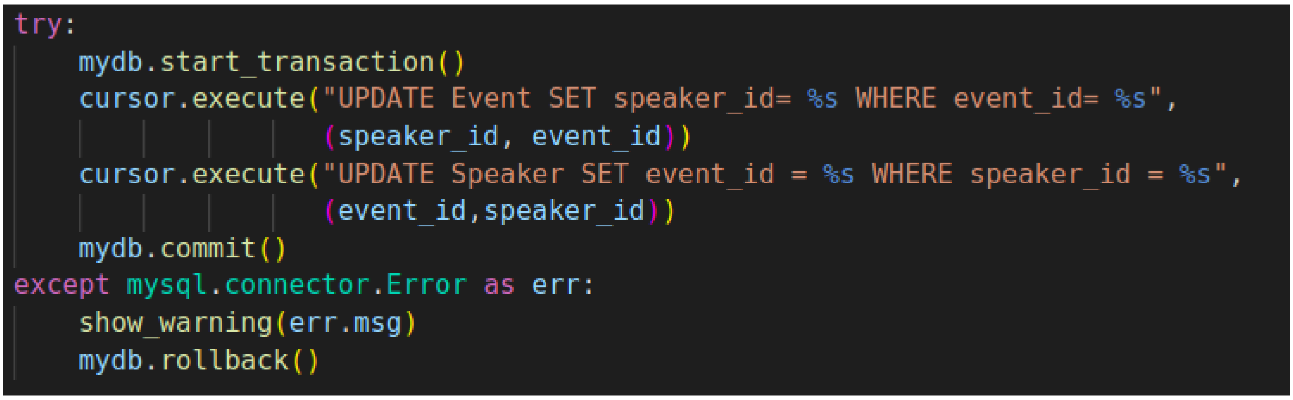
```
try:
    mydb.start_transaction()
    cursor.execute("SELECT event_name, date, venue_id FROM Event WHERE event_id= %s",(event_id,))
    event_details = cursor.fetchone()
    if event_details is not None:
        event_name, event_date, venue_id = event_details
        cursor.execute("SELECT venue_name FROM Venue WHERE venue_id= %s", (venue_id,))
        venue_name = cursor.fetchone()
        cursor.execute("""INSERT INTO Ticket(event_id,event_name,event_date,
                    venue_id,venue_name,attendee_id) values(%s,%s,%s,%s,%s,%s)""",
                (event_id,event_name,event_date,venue_id,venue_name[0],attendee_id))
        cursor.execute("UPDATE Event SET ticket_quantity = ticket_quantity -1 WHERE event_id=%s",
                (event_id,))
    mydb.commit()
except mysql.connector.Error as err:
    show_warning(err.msg)
    mydb.rollback()
```

- Transaction 3: Canceling Ticket



```
try:
    mydb.start_transaction()
    cursor.execute("SELECT event_id FROM Ticket WHERE ticket_id= %s", (ticket_id,))
    event_id = cursor.fetchone()
    cursor.execute("UPDATE EVENT SET ticket_quantity= ticket_quantity+1 WHERE event_id= %s",
                (event_id[0],))
    cursor.execute("DELETE FROM Ticket WHERE ticket_id= %s AND attendee_id=%s ",
                (ticket_id,attendee_id))
    mydb.commit()
except mysql.connector.Error as err:
    show_warning(err.msg)
    mydb.rollback()
```

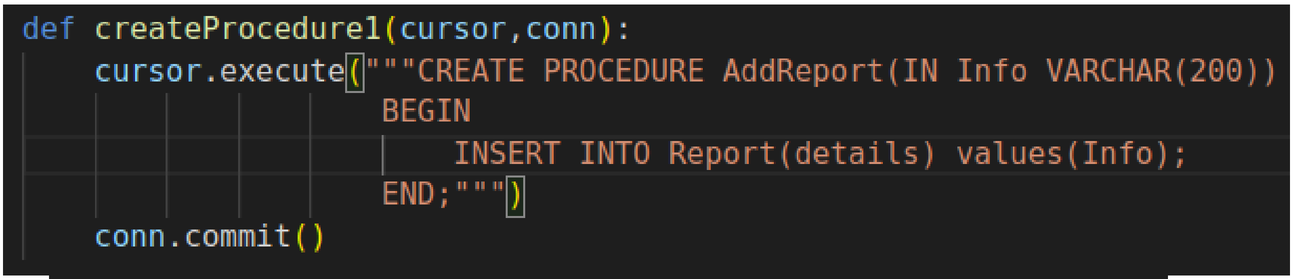
- Transaction 4: Attending Event - Speaker



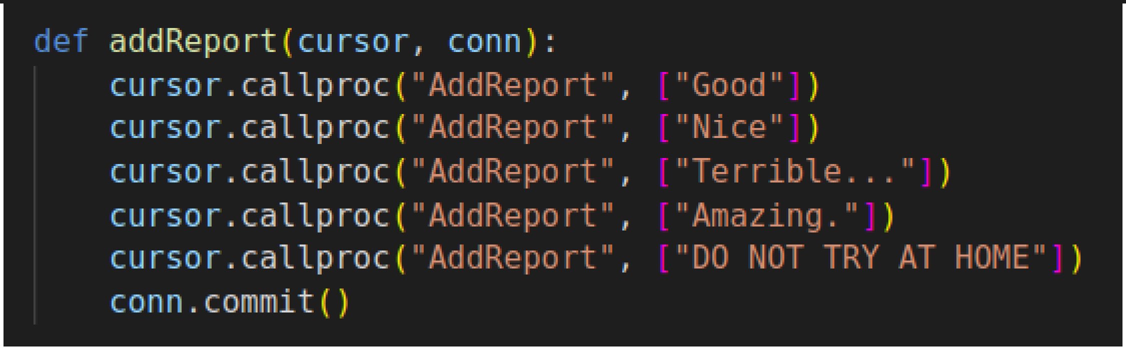
```
try:
    mydb.start_transaction()
    cursor.execute("UPDATE Event SET speaker_id= %s WHERE event_id= %s",
                (speaker_id, event_id))
    cursor.execute("UPDATE Speaker SET event_id = %s WHERE speaker_id = %s",
                (event_id,speaker_id))
    mydb.commit()
except mysql.connector.Error as err:
    show_warning(err.msg)
    mydb.rollback()
```

13) Procedures

- Procedure 1: AddReport

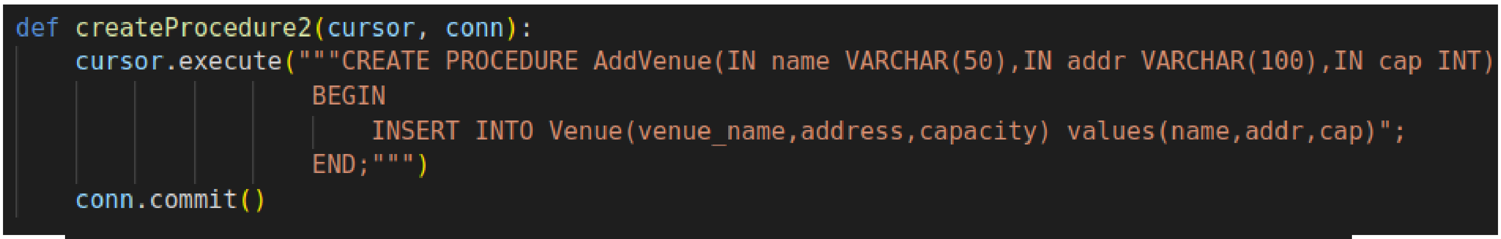


```
def createProcedure1(cursor,conn):
    cursor.execute("""CREATE PROCEDURE AddReport(IN Info VARCHAR(200))
                    BEGIN
                        INSERT INTO Report(details) values(Info);
                    END;""")
    conn.commit()
```

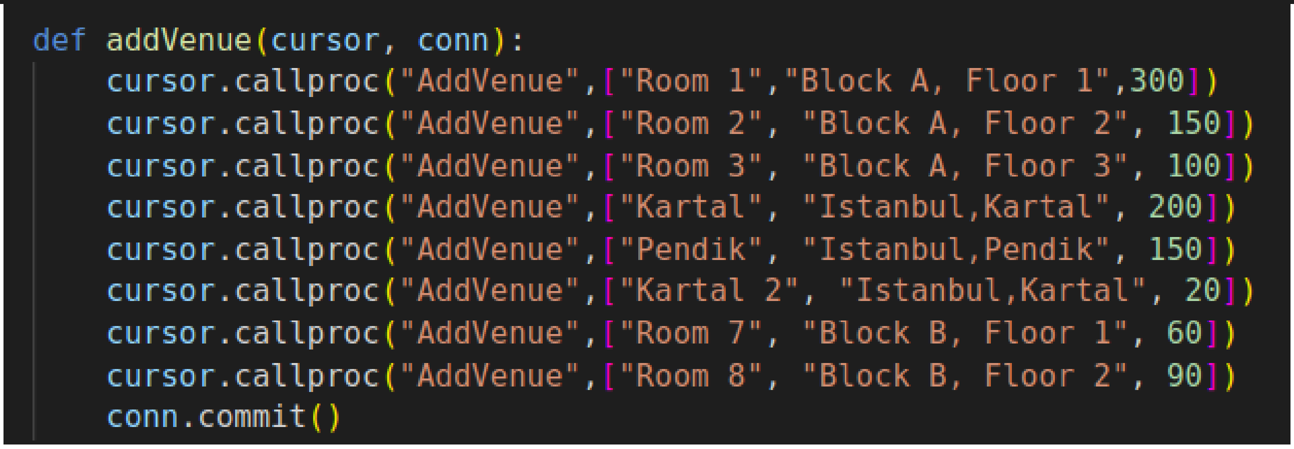


```
def addReport(cursor, conn):
    cursor.callproc("AddReport", ["Good"])
    cursor.callproc("AddReport", ["Nice"])
    cursor.callproc("AddReport", ["Terrible..."])
    cursor.callproc("AddReport", ["Amazing."])
    cursor.callproc("AddReport", ["DO NOT TRY AT HOME"])
    conn.commit()
```

- Procedure 2: AddVenue

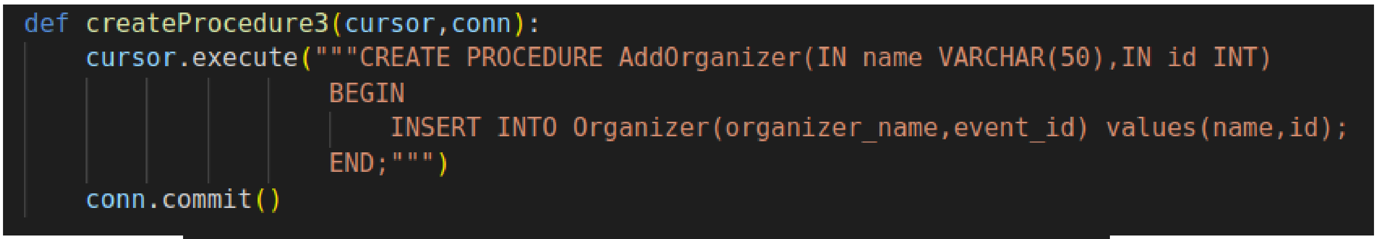


```
def createProcedure2(cursor, conn):
    cursor.execute("""CREATE PROCEDURE AddVenue(IN name VARCHAR(50),IN addr VARCHAR(100),IN cap INT)
                    BEGIN
                        INSERT INTO Venue(venue_name,address,capacity) values(name,addr,cap)";
                    END;""")
    conn.commit()
```

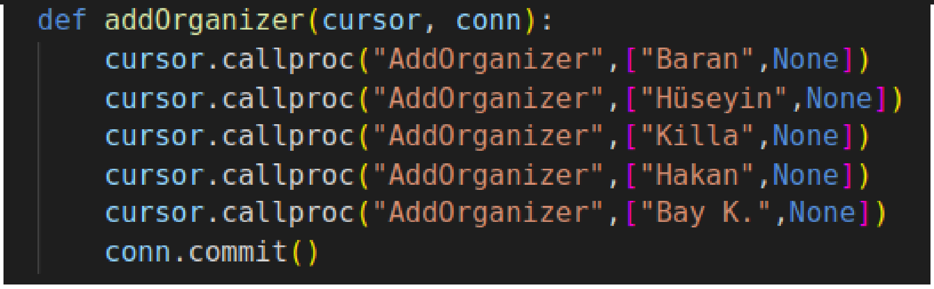


```
def addVenue(cursor, conn):
    cursor.callproc("AddVenue",["Room 1","Block A, Floor 1",300])
    cursor.callproc("AddVenue",["Room 2", "Block A, Floor 2", 150])
    cursor.callproc("AddVenue",["Room 3", "Block A, Floor 3", 100])
    cursor.callproc("AddVenue",["Kartal", "Istanbul,Kartal", 200])
    cursor.callproc("AddVenue",["Pendik", "Istanbul,Pendik", 150])
    cursor.callproc("AddVenue",["Kartal 2", "Istanbul,Kartal", 20])
    cursor.callproc("AddVenue",["Room 7", "Block B, Floor 1", 60])
    cursor.callproc("AddVenue",["Room 8", "Block B, Floor 2", 90])
    conn.commit()
```

- Procedure 3: AddOrganizer



```
def createProcedure3(cursor,conn):
    cursor.execute("""CREATE PROCEDURE AddOrganizer(IN name VARCHAR(50),IN id INT)
                    BEGIN
                        INSERT INTO Organizer(organizer_name,event_id) values(name,id);
                    END;""")
    conn.commit()
```



```
def addOrganizer(cursor, conn):
    cursor.callproc("AddOrganizer",["Baran",None])
    cursor.callproc("AddOrganizer",["Hüseyin",None])
    cursor.callproc("AddOrganizer",["Killa",None])
    cursor.callproc("AddOrganizer",["Hakan",None])
    cursor.callproc("AddOrganizer",["Bay K.",None])
    conn.commit()
```