

# Event Management

-Pranjali Datkhile



# Agenda

- Introduction
- Need of Event management
- System features & modules
- Codes
- Conclusion & Future scope

# Need of Event Management

- Streamlines planning
- Improves coordination
- Enhances attendee experience
- Saves time and reduces errors

# Technologies Used

- Python 3 – Application logic
- MySQL – Backend relational database
- MySQL Connector (Python) – For DB integration
- CSV Module – For exporting data
- Command Line Interface – User interaction



# CRUD OPERATIONS

Create (Insert)

Function: `create_event()` and `add_attendee()`

Action: Adds a new event or registers an attendee

Read (Select)

Function: `get_all_events()` and `get_attendees_by_event()`

Action: Retrieves data from the database

Update

Function: `UPDATE Events`

Action: Updates events

Delete

Function: `delete_event(event_id)`

Action: Removes an event and all its attendees



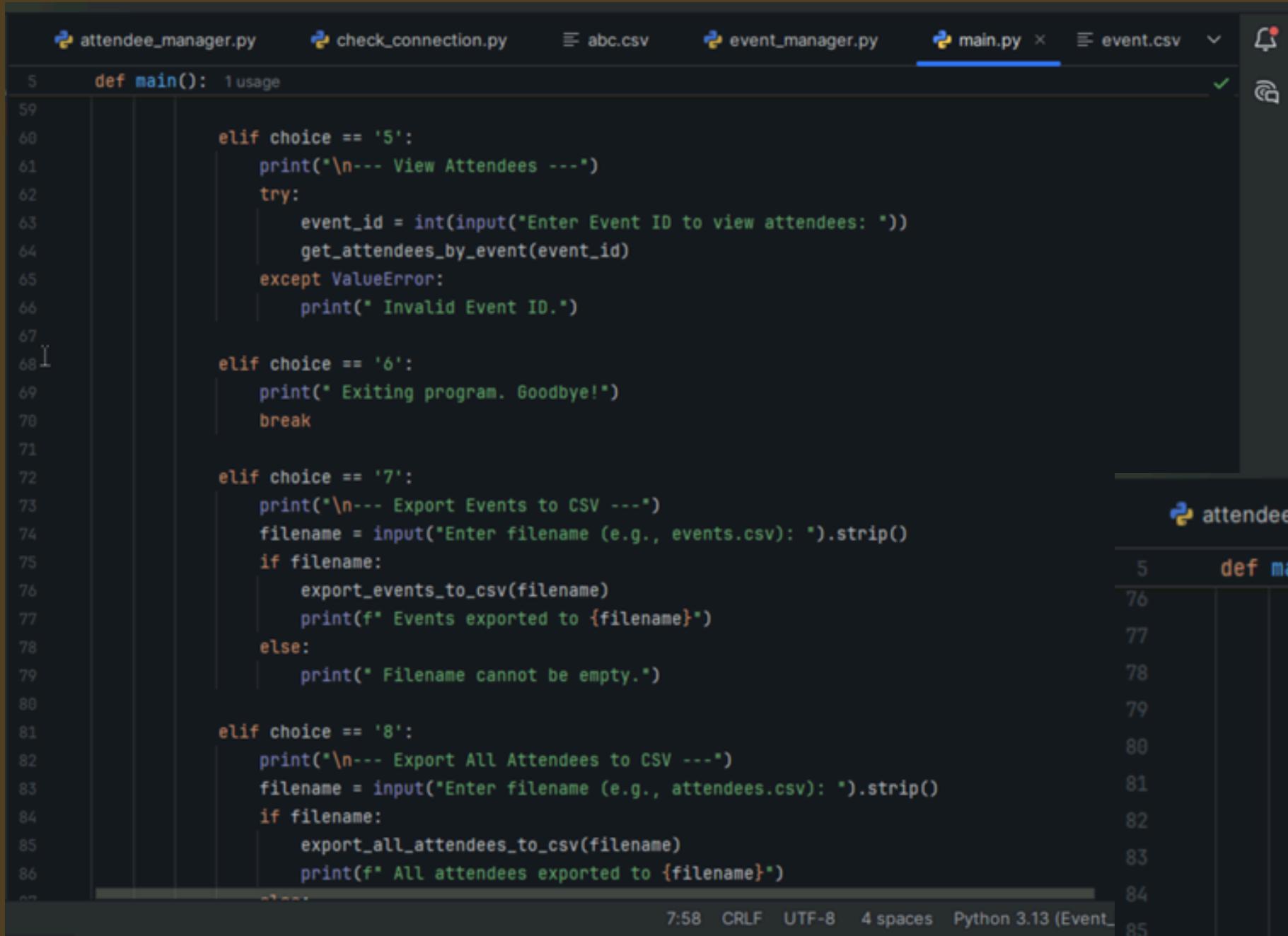
M  
A  
I  
N  
.  
P  
Y

```
4 def main(): usage
5     print("*" * 40)
6
7     print(" Welcome to Event Management System ")
8     print("*" * 40)
9
10    while True:
11        try:
12            print("\nMenu:")
13            print("1. Create Event")
14            print("2. View Events")
15            print("3. Delete Event")
16            print("4. Add Attendee")
17            print("5. View Attendees by Event")
18            print("6. Exit")
19            print("7. Export Events to CSV")
20            print("8. Export All Attendees to CSV")
21
22            choice = input("Enter your choice (1-8): ").strip()
23
24            if choice == '1':
25                print("\n--- Create Event ---")
26                name = input("Event Name: ").strip()
27                date = input("Event Date (YYYY-MM-DD): ").strip()
28                location = input("Location: ").strip()
29                description = input("Description: ").strip()
30                if name and date:
31                    create_event(name, date, location, description)
32                    print(" Event added successfully.")
33                else:
```

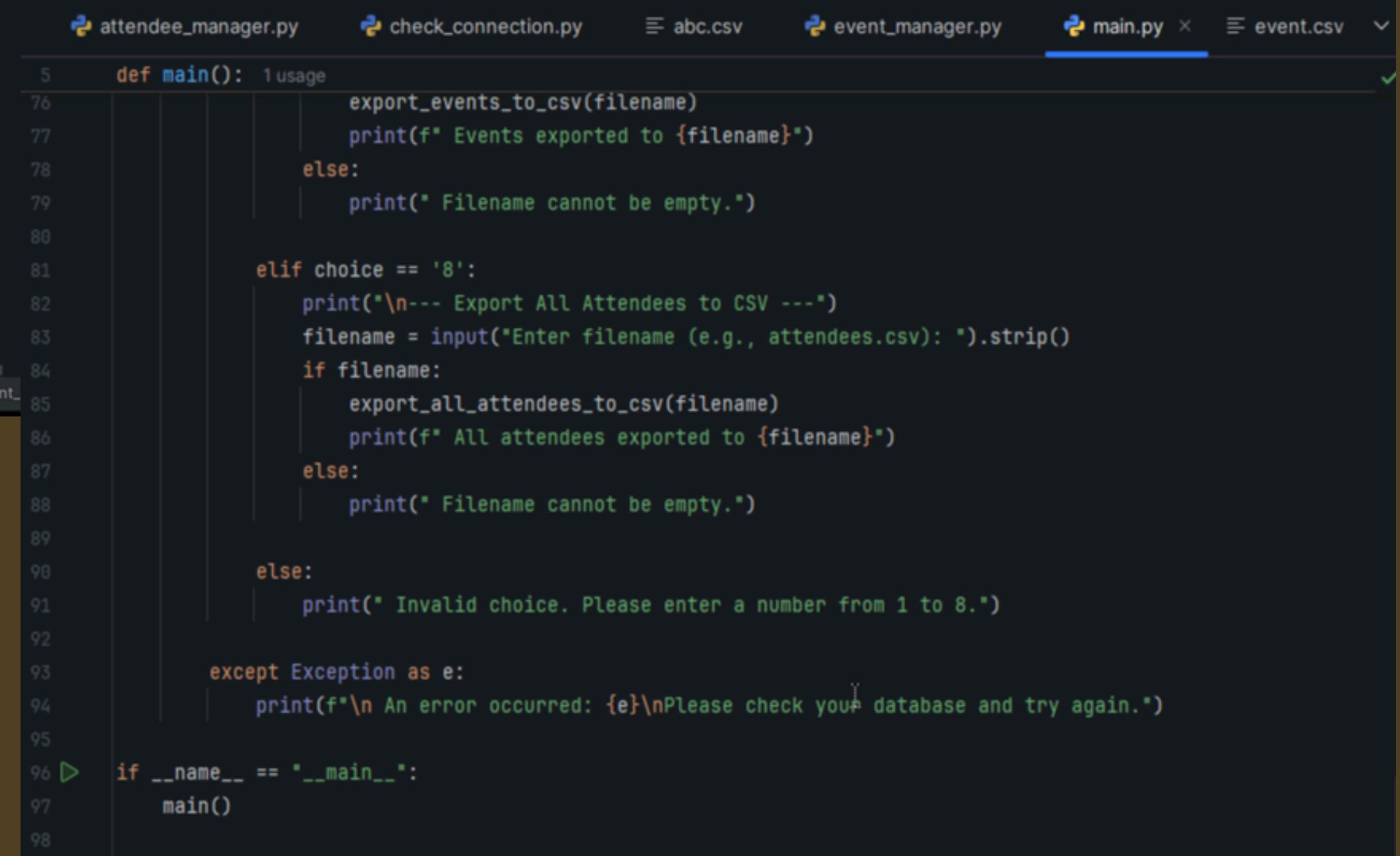
7:58 CRLF UTF-8 4 spaces Python 3.13 (Event\_M)

```
5     def main(): usage
35
36         elif choice == '2':
37             print("\n--- All Events ---")
38             get_all_events()
39
40         elif choice == '3':
41             print("\n--- Delete Event ---")
42             try:
43                 event_id = int(input("Enter Event ID to delete: "))
44                 delete_event(event_id)
45                 print(" Event deleted.")
46             except ValueError:
47                 print(" Invalid Event ID. Please enter a number.")
48
49         elif choice == '4':
50             print("\n--- Add Attendee ---")
51             name = input("Attendee Name: ").strip()
52             email = input("Email: ").strip()
53             try:
54                 event_id = int(input("Event ID to register for: "))
55                 add_attendee(name, email, event_id)
56                 print(" Attendee added.")
57             except ValueError:
58                 print(" Invalid Event ID.")
59
60         elif choice == '5':
61             print("\n--- View Attendees ---")
62             try:
```

7:58 CRLF UTF-8 4 spaces Python 3.13 (Event\_Management) ⌂



```
5     def main(): 1 usage
59
60         elif choice == '5':
61             print("\n--- View Attendees ---")
62             try:
63                 event_id = int(input("Enter Event ID to view attendees: "))
64                 get_attendees_by_event(event_id)
65             except ValueError:
66                 print(" Invalid Event ID.")
67
68         elif choice == '6':
69             print(" Exiting program. Goodbye!")
70             break
71
72         elif choice == '7':
73             print("\n--- Export Events to CSV ---")
74             filename = input("Enter filename (e.g., events.csv): ").strip()
75             if filename:
76                 export_events_to_csv(filename)
77                 print(f" Events exported to {filename}")
78             else:
79                 print(" Filename cannot be empty.")
80
81         elif choice == '8':
82             print("\n--- Export All Attendees to CSV ---")
83             filename = input("Enter filename (e.g., attendees.csv): ").strip()
84             if filename:
85                 export_all_attendees_to_csv(filename)
86                 print(f" All attendees exported to {filename}")
87
88
89
90
91
92
93
94
95
96 D  if __name__ == "__main__":
97     main()
```



```
5     def main(): 1 usage
56
57         export_events_to_csv(filename)
58         print(f" Events exported to {filename}")
59
60     elif choice == '8':
61         print("\n--- Export All Attendees to CSV ---")
62         filename = input("Enter filename (e.g., attendees.csv): ").strip()
63         if filename:
64             export_all_attendees_to_csv(filename)
65             print(f" All attendees exported to {filename}")
66         else:
67             print(" Filename cannot be empty.")
68
69
70     else:
71         print(" Invalid choice. Please enter a number from 1 to 8.")
72
73
74     except Exception as e:
75         print(f"\n An error occurred: {e}\nPlease check your database and try again.")
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96 D  if __name__ == "__main__":
97     main()
```

A  
T  
T  
E  
N  
D  
E  
—  
M  
A  
N  
A  
G  
E  
R  
. .  
P  
y

```
1  # attendee_manager.py
2  from db_connection import get_connection
3  import csv
4
5  def add_attendee(name, email, event_id):
6      conn = get_connection()
7      cursor = conn.cursor()
8      query = "INSERT INTO Attendees (name, email, event_id) VALUES (%s, %s, %s)"
9      cursor.execute(query, (name, email, event_id))
10     conn.commit()
11     cursor.close()
12     conn.close()
13
14  def get_attendees_by_event(event_id, return_data=False):
15      conn = get_connection()
16      cursor = conn.cursor(dictionary=True) # dictionary=True to get dict rows
17      query = "SELECT name, email FROM Attendees WHERE event_id = %s"
18      cursor.execute(query, (event_id,))
19      attendees = cursor.fetchall()
20      cursor.close()
21      conn.close()
22
23      if return_data:
24          return attendees
25
26      if attendees:
27          for attendee in attendees:
28              print(attendee)
29      else:
```

```
27          for attendee in attendees:
28              print(attendee)
29      else:
30          print("No attendees found for this event.")
31
32  def export_attendees_to_csv(event_id, filename=None):
33      attendees = get_attendees_by_event(event_id, return_data=True)
34      if not attendees:
35          print("No attendees to export for this event.")
36          return
37
38      # Default filename if not provided
39      if not filename:
40          filename = f"attendees_export_event_{event_id}.csv"
41
42      with open(filename, mode='w', newline='', encoding='utf-8') as file:
43          writer = csv.writer(file)
44          writer.writerow(["Name", "Email"])
45          for attendee in attendees:
46              writer.writerow([
47                  attendee.get('name', 'N/A'), # Use 'N/A' if 'name' is missing
48                  attendee.get('email', 'N/A'), # Use 'N/A' if 'email' is missing
49              ])
50
51      print(f"Attendees for event {event_id} successfully exported to {filename}")
```



E  
V  
E  
N  
T  
—  
M  
A  
N  
A  
G  
E  
R  
.  
P  
Y

```
1  # event_manager.py
2  from db_connection import get_connection
3  import csv
4  from attendee_manager import get_attendees_by_event # Import the attendee manager
5
6  def create_event(name, date, location, description):
7      conn = get_connection()
8      cursor = conn.cursor()
9      query = "INSERT INTO Events (name, date, location, description) VALUES (%s, %s, %s, %s)"
10     cursor.execute(query, (name, date, location, description))
11     conn.commit()
12     cursor.close()
13     conn.close()
14
15 def get_all_events(return_data=False):
16     conn = get_connection()
17     cursor = conn.cursor(dictionary=True) # dictionary=True to get dict results
18     query = "SELECT * FROM Events"
19     cursor.execute(query)
20     events = cursor.fetchall()
21     cursor.close()
22     conn.close()
23
24     if return_data:
25         return events
26
27     if events:
28         for event in events:
29             print(event)
```

```
29         print(event)
30     else:
31         print("No events found.")
32
33 def delete_event(event_id):
34     conn = get_connection()
35     cursor = conn.cursor()
36     query = "DELETE FROM Events WHERE event_id = %s"
37     cursor.execute(query, (event_id,))
38     conn.commit()
39     cursor.close()
40     conn.close()
41
42     # creating csv files!
43 def export_events_to_csv(filename='events_export.csv'):
44     events = get_all_events(return_data=True)
45     if not events:
46         print("No events to export.")
47         return
48
49     with open(filename, mode='w', newline='', encoding='utf-8') as file:
50         writer = csv.writer(file)
51         writer.writerow(["ID", "Name", "Date", "Location", "Description"])
52         for event in events:
53             writer.writerow([
54                 event.get('event_id'),
55                 event.get('name'),
56                 event.get('date'),
```

E

V

E

N

T

—

M

A

N

A

G

E

R

.

P

Y

```

56         event.get('date'),
57         event.get('location'),
58         event.get('description')
59     ])
60     print(f"Events successfully exported to {filename}")
61
62     def export_all_attendees_to_csv(filename='attendees_export.csv'):
63         conn = get_connection()
64         cursor = conn.cursor(dictionary=True)
65         query = "SELECT event_id, name FROM Events" # Getting event ID and Name to export with attendees
66         cursor.execute(query)
67         events = cursor.fetchall()
68         cursor.close()
69         conn.close()
70
71     if not events:
72         print("No events found to export attendees.")
73         return
74
75     with open(filename, mode='w', newline='', encoding='utf-8') as file:
76         writer = csv.writer(file)
77         writer.writerow(["Event ID", "Event Name", "Attendee Name", "Attendee Email"])
78
79     for event in events:
80         event_id = event['event_id']
81         event_name = event['name']
82         print(f"Exporting attendees for event ID: {event_id} - {event_name}") # Debugging line
83         attendees = get_attendees_by_event(event_id, return_data=True)
84

```

```

80         event_id = event['event_id']
81         event_name = event['name']
82         print(f"Exporting attendees for event ID: {event_id} - {event_name}") # Debugging line
83         attendees = get_attendees_by_event(event_id, return_data=True)
84
85     if attendees:
86         for attendee in attendees:
87             # Debugging: Print the attendee data to inspect
88             print(f"Attendee Data: {attendee}") # Debugging line
89
90             # Safely access name and email fields
91             name = attendee.get('name', 'N/A') # Use 'N/A' if 'name' is missing
92             email = attendee.get('email', 'N/A') # Use 'N/A' if 'email' is missing
93
94             writer.writerow([
95                 event_id,
96                 event_name,
97                 name,
98                 email
99             ])
100
101     else:
102         print(f"No attendees found for event ID {event_id}") # Debugging line
103

```



## Connection With Database

```
1 # db_connection.py
2 import mysql.connector
3 from db_config import config
4
5 def get_connection():
6     return mysql.connector.connect(**config)
```

```
1 # db_config.py
2 config = {
3     'host': 'localhost',
4     'user': 'root',
5     'password': 'pass@word1', #
6     'database': 'EventManagement'
7 }
```



# Database

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    department VARCHAR(50),
    salary FLOAT
);
```

Result Grid | Filter Rows:  Edit: Result Grid | Form Editor

	attendee_id	name	email	event_id
▶	1	Pranjali	abc@email.com	4
●	2	XYZ	xyz@email.com	4
*	NULL	NULL	NULL	NULL

attendees 2 x Apply Revert

1	Event ID	Event Name	Attendee Name	Attendee Email
2	4	Birthday	Pranjali	abc@email.com
3	4	Birthday	XYZ	xyz@email.com

1	ID	Name	Date	Location	Description
2	4	Birthday	2025-07-20	paris	vintage theme party
3	5	house warming	2025-08-01	New york	no gifts, only presence

The screenshot shows a Windows desktop environment with a Python application running in the Visual Studio Code (VS Code) interface. The application is titled "Event\_Management".

**Project Explorer:** Shows the project structure under "Event\_Management". The ".venv" folder contains "attendee\_manager.py", "attendees.csv", "check\_connection.py", "db\_config.py", "db\_connection.py", "event.csv" (selected), "event\_manager.py", and "main.py".

**Code Editor:** Displays the "event.csv" file content:

ID	Name	Date	Location	Description
1	4	Birthday	2025-07-20	paris
2	5	house warming	2025-08-01	New york

**Terminal:** Shows the application's output:

```
Enter your choice (1-8): 8
--- Export All Attendees to CSV ---
Enter filename (e.g., attendees.csv): abc.csv
Exporting attendees for event ID: 4 - Birthday
Attendee Data: {'name': 'Pranjali', 'email': 'abc@email.com'}
Attendee Data: {'name': 'XYZ', 'email': 'xyz@email.com'}
Exporting attendees for event ID: 5 - house warming
No attendees found for event ID 5
Attendees for all events successfully exported to abc.csv
All attendees exported to abc.csv
```

**Status Bar:** Shows "No aggregator selected" and "Python 3.13 (Event\_Management)". The system tray indicates the date as 14-09-2025.

## Conclusion

- Simple CLI-based event management tool
- Useful for small organizations or academic projects

## Future Scope

- Add a GUI or web interface
- Email notifications to attendees
- Enhanced validation and reporting
- Role-based user authentication

Thank You