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

MAIN.PY

attendee_manager.py | check_connection.py | abc.csv | event_manager.py | main.py | event.csv

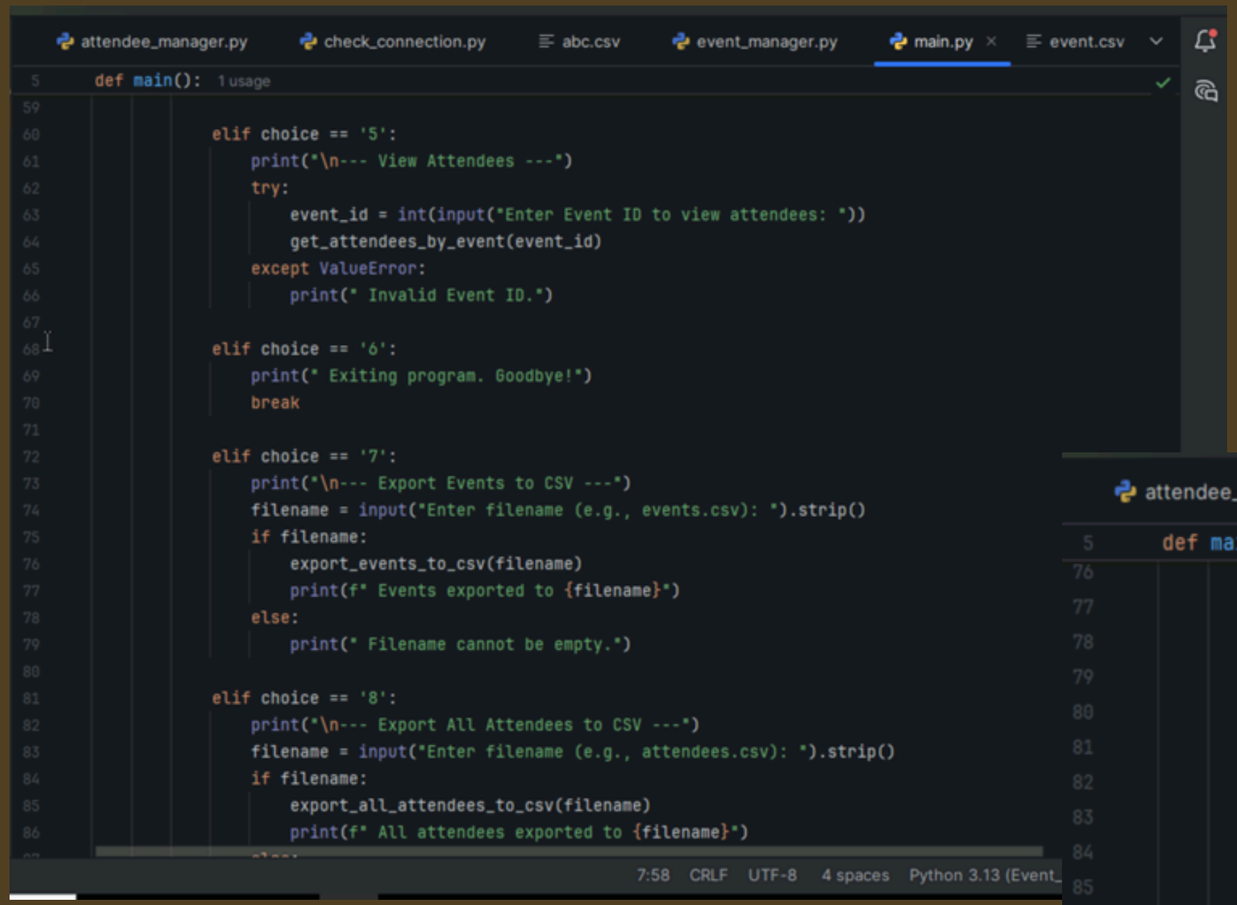
```python
def main():  1 usage
    print("=" * 40)
    print("     Welcome to Event Management System     ")
    print("=" * 40)

    while True:
        try:
            print("\nMenu:")
            print("1. Create Event")
            print("2. View Events")
            print("3. Delete Event")
            print("4. Add Attendee")
            print("5. View Attendees by Event")
            print("6. Exit")
            print("7. Export Events to CSV")
            print("8. Export All Attendees to CSV")

            choice = input("Enter your choice (1-8): ").strip()

            if choice == '1':
                print("\n--- Create Event ---")
                name = input("Event Name: ").strip()
                date = input("Event Date (YYYY-MM-DD): ").strip()
                location = input("Location: ").strip()
                description = input("Description: ").strip()
                if name and date:
                    create_event(name, date, location, description)
                    print(" Event added successfully.")
                else:
```

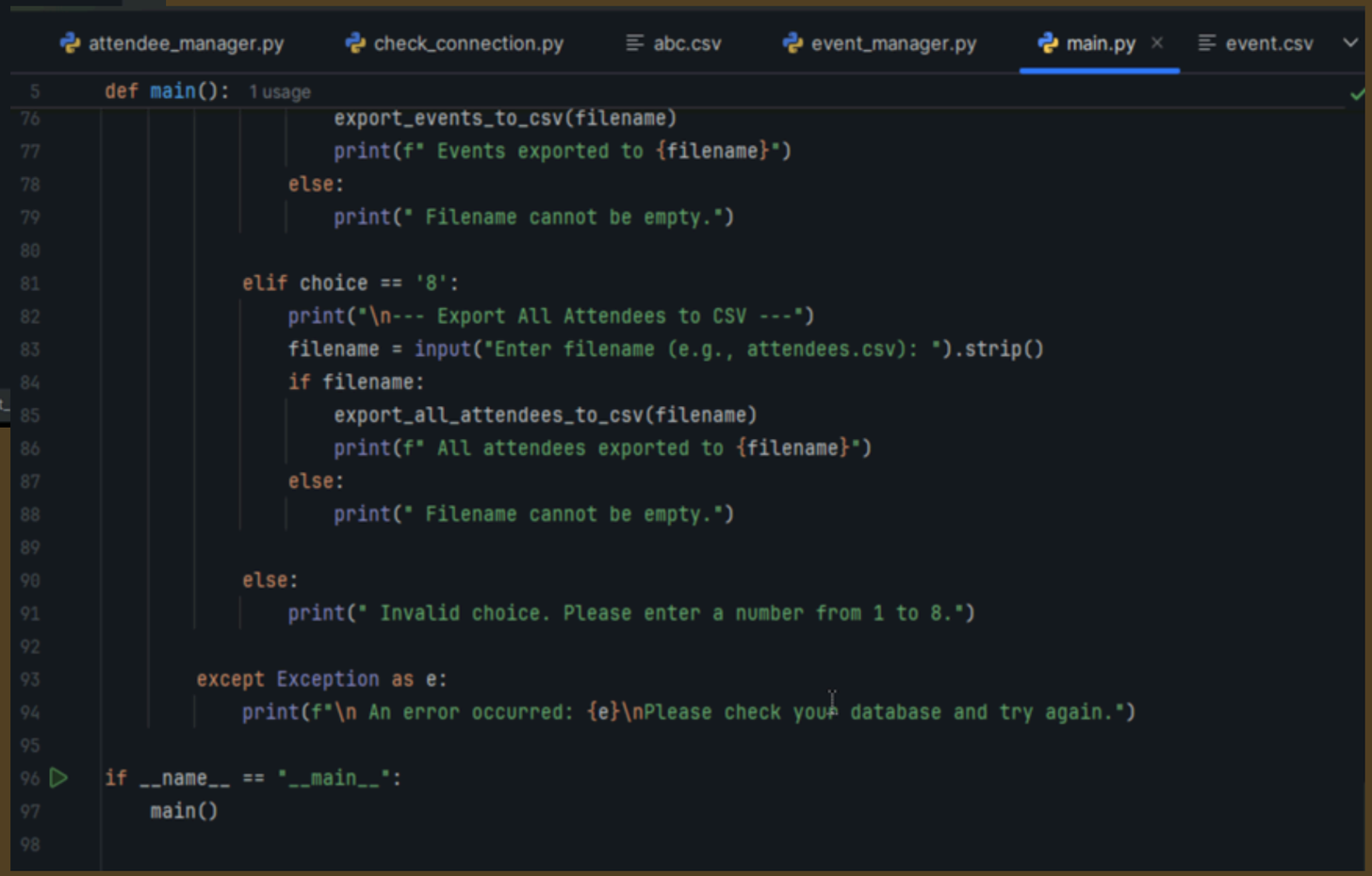
7:58 CRLF UTF-8 4 spaces Python 3.13 (Event_M

attendee_manager.py | check_connection.py | abc.csv | event_manager.py | main.py | event.csv

```python
def main():  1 usage

            elif choice == '2':
                print("\n--- All Events ---")
                get_all_events()

            elif choice == '3':
                print("\n--- Delete Event ---")
                try:
                    event_id = int(input("Enter Event ID to delete: "))
                    delete_event(event_id)
                    print(" Event deleted.")
                except ValueError:
                    print(" Invalid Event ID. Please enter a number.")

            elif choice == '4':
                print("\n--- Add Attendee ---")
                name = input("Attendee Name: ").strip()
                email = input("Email: ").strip()
                try:
                    event_id = int(input("Event ID to register for: "))
                    add_attendee(name, email, event_id)
                    print(" Attendee added.")
                except ValueError:
                    print(" Invalid Event ID.")

            elif choice == '5':
                print("\n--- View Attendees ---")
                try:
```

7:58 CRLF UTF-8 4 spaces Python 3.13 (Event_Management)


attendee_manager.py
check_connection.py
abc.csv
event_manager.py
main.py
event.csv
def main(): 1 usage
elif choice == '5':
print("\n--- View Attendees ---")
try:
event_id = int(input("Enter Event ID to view attendees: "))
get_attendees_by_event(event_id)
except ValueError:
print(" Invalid Event ID.")
elif choice == '6':
print(" Exiting program. Goodbye!")
break
elif choice == '7':
print("\n--- Export Events to CSV ---")
filename = input("Enter filename (e.g., events.csv): ").strip()
if filename:
export_events_to_csv(filename)
print(f" Events exported to {filename}")
else:
print(" Filename cannot be empty.")
elif choice == '8':
print("\n--- Export All Attendees to CSV ---")
filename = input("Enter filename (e.g., attendees.csv): ").strip()
if filename:
export_all_attendees_to_csv(filename)
print(f" All attendees exported to {filename}")
7:58 CRLF UTF-8 4 spaces Python 3.13 (Event_


attendee_manager.py
check_connection.py
abc.csv
event_manager.py
main.py
event.csv
def main(): 1 usage
export_events_to_csv(filename)
print(f" Events exported to {filename}")
else:
print(" Filename cannot be empty.")
elif choice == '8':
print("\n--- Export All Attendees to CSV ---")
filename = input("Enter filename (e.g., attendees.csv): ").strip()
if filename:
export_all_attendees_to_csv(filename)
print(f" All attendees exported to {filename}")
else:
print(" Filename cannot be empty.")
else:
print(" Invalid choice. Please enter a number from 1 to 8.")
except Exception as e:
print(f"\n An error occurred: {e}\nPlease check your database and try again.")
if __name__ == "__main__":
main()

ATTENDEE_MANAGER.PY

```python
# attendee_manager.py
from db_connection import get_connection
import csv

def add_attendee(name, email, event_id):
    conn = get_connection()
    cursor = conn.cursor()
    query = "INSERT INTO Attendees (name, email, event_id) VALUES (%s, %s, %s)"
    cursor.execute(query, (name, email, event_id))
    conn.commit()
    cursor.close()
    conn.close()

def get_attendees_by_event(event_id, return_data=False):
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)  # dictionary=True to get dict rows
    query = "SELECT name, email FROM Attendees WHERE event_id = %s"
    cursor.execute(query, (event_id,))
    attendees = cursor.fetchall()
    cursor.close()
    conn.close()

    if return_data:
        return attendees

    if attendees:
        for attendee in attendees:
            print(attendee)
    else:
        for attendee in attendees:
            print(attendee)
    else:
        print("No attendees found for this event.")

def export_attendees_to_csv(event_id, filename=None):
    attendees = get_attendees_by_event(event_id, return_data=True)
    if not attendees:
        print("No attendees to export for this event.")
        return

    # Default filename if not provided
    if not filename:
        filename = f"attendees_export_event_{event_id}.csv"

    with open(filename, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(["Name", "Email"])
        for attendee in attendees:
            writer.writerow([
                attendee.get('name', 'N/A'),  # Use 'N/A' if 'name' is missing
                attendee.get('email', 'N/A'),  # Use 'N/A' if 'email' is missing
            ])

    print(f"Attendees for event {event_id} successfully exported to {filename}")
```

EVENT_MANAGER.PY

```python
# event_manager.py
from db_connection import get_connection
import csv
from attendee_manager import get_attendees_by_event  # Import the attendee manager

def create_event(name, date, location, description):
    conn = get_connection()
    cursor = conn.cursor()
    query = "INSERT INTO Events (name, date, location, description) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (name, date, location, description))
    conn.commit()
    cursor.close()
    conn.close()

def get_all_events(return_data=False):
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)  # dictionary=True to get dict results
    query = "SELECT * FROM Events"
    cursor.execute(query)
    events = cursor.fetchall()
    cursor.close()
    conn.close()

    if return_data:
        return events

    if events:
        for event in events:
            print(event)
    else:
        print("No events found.")

def delete_event(event_id):
    conn = get_connection()
    cursor = conn.cursor()
    query = "DELETE FROM Events WHERE event_id = %s"
    cursor.execute(query, (event_id,))
    conn.commit()
    cursor.close()
    conn.close()

# creating csv files!
def export_events_to_csv(filename='events_export.csv'):
    events = get_all_events(return_data=True)
    if not events:
        print("No events to export.")
        return

    with open(filename, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(["ID", "Name", "Date", "Location", "Description"])
        for event in events:
            writer.writerow([
                event.get('event_id'),
                event.get('name'),
                event.get('date'),
```

EVENT_MANAGER.py

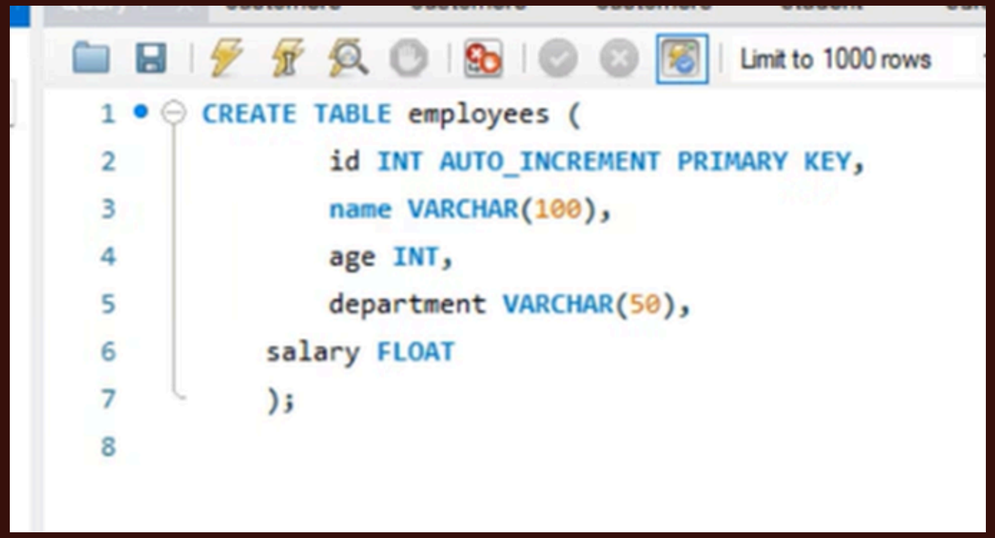
```
                event.get('date'),
                event.get('location'),
                event.get('description')
            ])
    print(f"Events successfully exported to {filename}")

def export_all_attendees_to_csv(filename='attendees_export.csv'):
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)
    query = "SELECT event_id, name FROM Events"  # Getting event ID and Name to export with attendees
    cursor.execute(query)
    events = cursor.fetchall()
    cursor.close()
    conn.close()

    if not events:
        print("No events found to export attendees.")
        return

    with open(filename, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(["Event ID", "Event Name", "Attendee Name", "Attendee Email"])

        for event in events:
            event_id = event['event_id']
            event_name = event['name']
            print(f"Exporting attendees for event ID: {event_id} - {event_name}")  # Del
            attendees = get_attendees_by_event(event_id, return_data=True)

            event_id = event['event_id']
            event_name = event['name']
            print(f"Exporting attendees for event ID: {event_id} - {event_name}")  # Debugging line
            attendees = get_attendees_by_event(event_id, return_data=True)

            if attendees:
                for attendee in attendees:
                    # Debugging: Print the attendee data to inspect
                    print(f"Attendee Data: {attendee}")  # Debugging line

                    # Safely access name and email fields
                    name = attendee.get('name', 'N/A')  # Use 'N/A' if 'name' is missing
                    email = attendee.get('email', 'N/A')  # Use 'N/A' if 'email' is missing

                    writer.writerow([
                        event_id,
                        event_name,
                        name,
                        email
                    ])
            else:
                print(f"No attendees found for event ID {event_id}")  # Debugging line

    print(f"Attendees for all events successfully exported to {filename}")
```

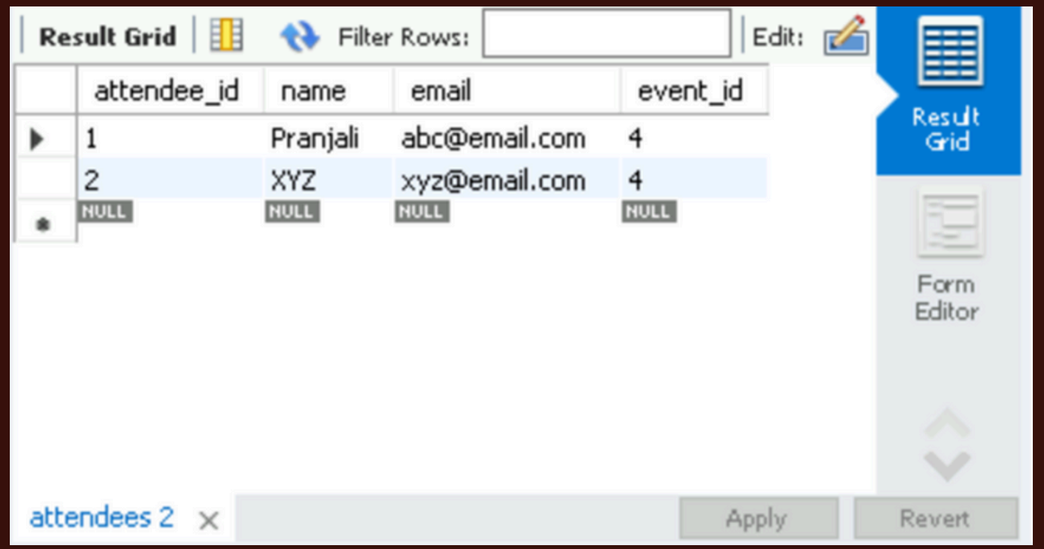
Connection With Database

```python
# db_connection.py
import mysql.connector
from db_config import config

def get_connection():
    return mysql.connector.connect(**config)
```

```python
# db_config.py
config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'pass@word1',  #
    'database': 'EventManagement'
}
```

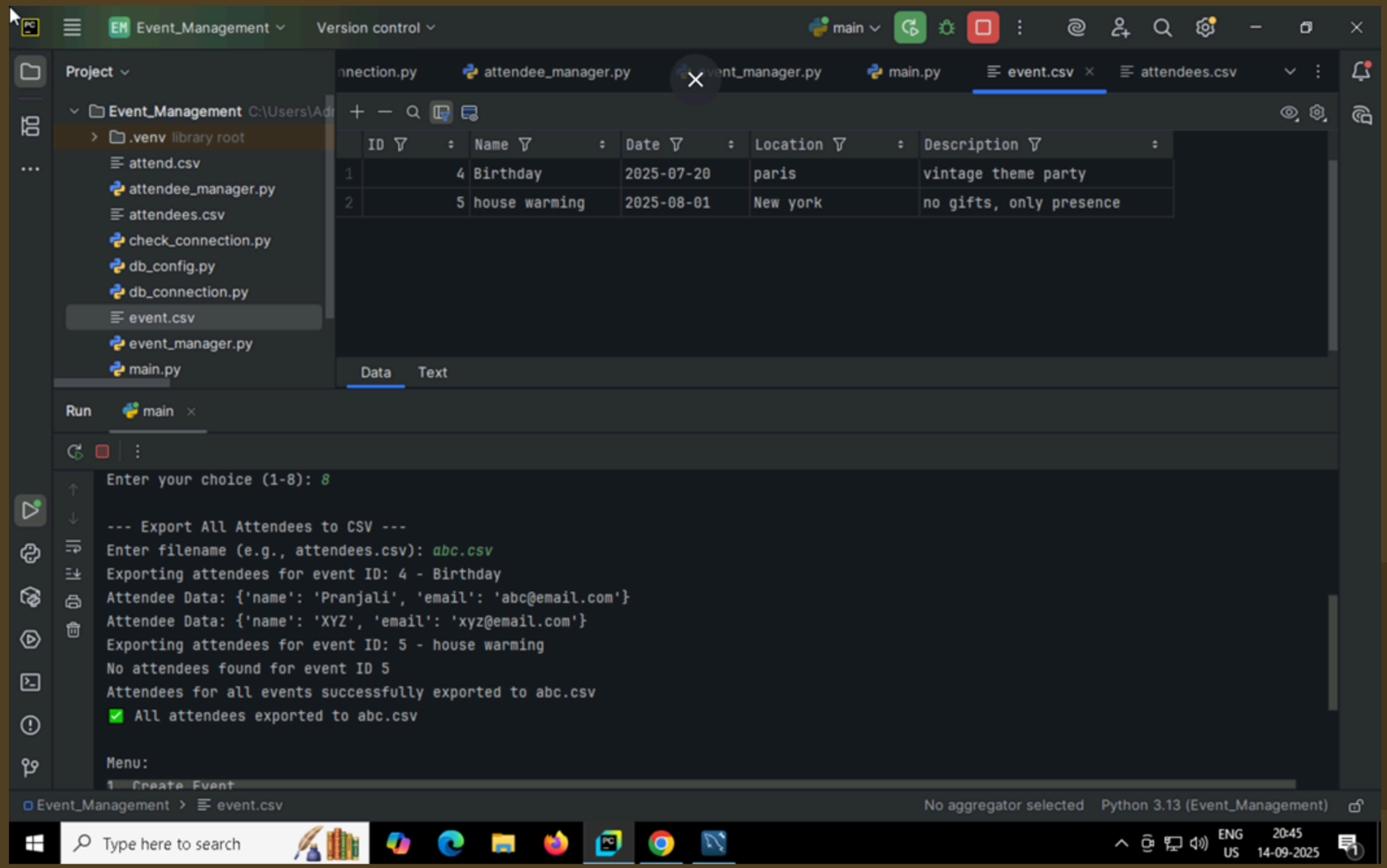
Database





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



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