

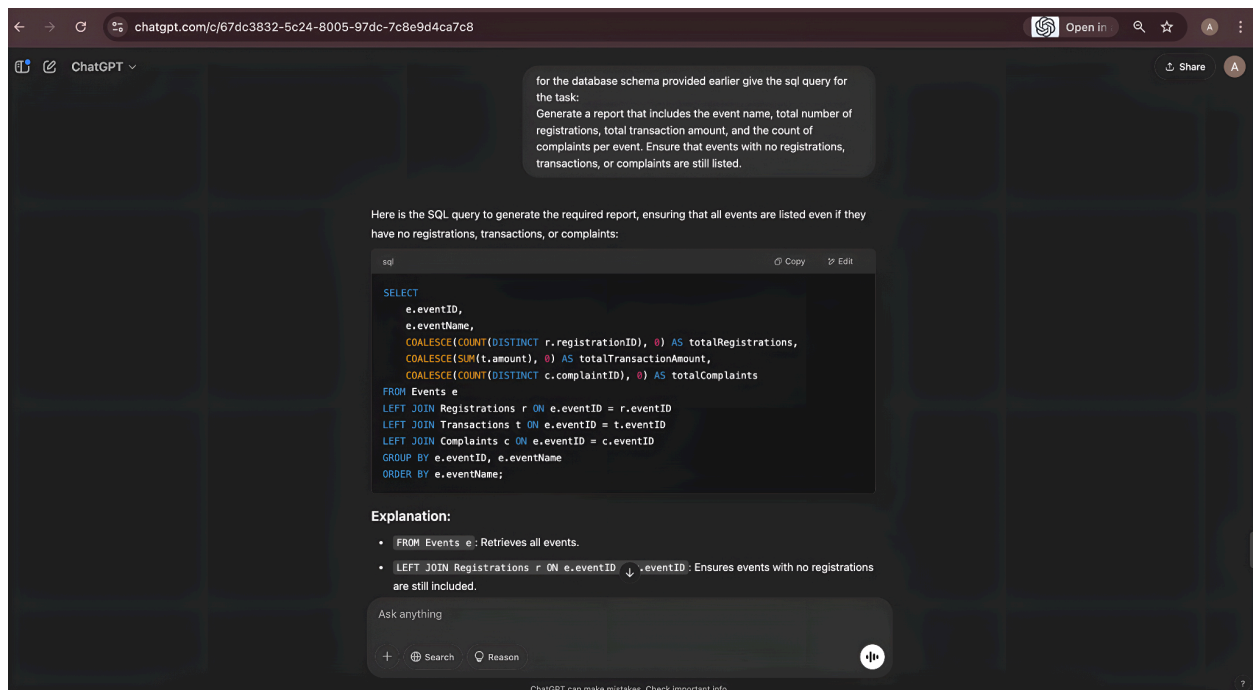
Query 1

Task: Generate a report that includes the event name, total number of registrations, total transaction amount, and the count of complaints per event. Ensure that events with no registrations, transactions, or complaints are still listed.

SQL Query Provided by ChatGPT:

```
SELECT
  e.eventID,
  e.eventName,
  COALESCE(COUNT(DISTINCT r.registrationID), 0) AS totalRegistrations,
  COALESCE(SUM(t.amount), 0) AS totalTransactionAmount,
  COALESCE(COUNT(DISTINCT c.complaintID), 0) AS totalComplaints
FROM Events e
LEFT JOIN Registrations r ON e.eventID = r.eventID
LEFT JOIN Transactions t ON e.eventID = t.eventID
LEFT JOIN Complaints c ON e.eventID = c.eventID
GROUP BY e.eventID, e.eventName
ORDER BY e.eventName;
```

Screenshot:



Issue:

Example Dataset to show the problem:

-- Insert Users

```
INSERT INTO Users (username, "password", firstName, lastName, emailID, contactNo, dob,
walletCash)
```

```
VALUES
```

```
  ('user1', 'password1', 'Alice', 'Smith', 'alice@example.com', '1234567890', '1990-01-01',
100.00),
  ('user2', 'password2', 'Bob', 'Jones', 'bob@example.com', '2345678901', '1985-05-15', 150.00);
```

-- Insert Admins

```
INSERT INTO Admins (firstName, lastName, emailID, "password", "role")
```

```
VALUES
```

```
  ('Admin', 'One', 'admin1@example.com', 'adminpass', 'Manager');
```

-- Insert Organizers

-- Here, Organizer with organizerID = 1 is verified and will be used for our events.

```
INSERT INTO Organizer (staffID, username, firstName, lastName, emailID, contactNo,
"password", organization, verificationStatus)
```

```
VALUES
```

```
  (1, 'org1', 'Organizer', 'One', 'org1@example.com', '3456789012', 'orgpass1', 'OrgOne Inc.',
TRUE),
  (1, 'org2', 'Organizer', 'Two', 'org2@example.com', '4567890123', 'orgpass2', 'OrgTwo LLC',
FALSE);
```

-- Insert Venues

```
INSERT INTO Venue ("name", street, city, "state", pincode)
```

```
VALUES
```

```
  ('Main Hall', '123 Main St', 'CityA', 'StateA', '12345'),
  ('Conference Center', '456 Center Rd', 'CityB', 'StateB', '23456');
```

-- Insert Events

-- We'll create three events.

```
INSERT INTO Events (eventName, ticketPrice, category, maxAttendees, startDate, startTime,
endDate, endTime, venueID, organizerID, ticketsSold, status)
```

```
VALUES
```

```
  ('Concert', 50.00, 'Music', 150, '2025-03-25', '20:00:00', '2025-03-25', '23:00:00', 1, 1, 120,
'Upcoming'),
  ('Conference', 100.00, 'Business', 250, '2025-03-26', '09:00:00', '2025-03-26', '17:00:00', 2, 1,
200, 'Upcoming'),
  ('Workshop', 30.00, 'Education', 60, '2025-03-27', '10:00:00', '2025-03-27', '12:00:00', 1, 1, 50,
'Upcoming');
```

-- Insert Registration records

-- Table: Registration (singular)

```

-- For "Concert" (eventID=1), we insert two registrations.
-- For "Conference" (eventID=2), we insert one registration.
-- "Workshop" (eventID=3) will have no registrations.
INSERT INTO Registration (UserID, EventID, TransactionID)
VALUES
  (1, 1, 'TXN101'),
  (2, 1, 'TXN102'),
  (1, 2, 'TXN103');

-- Insert Transactions
-- For "Concert" (eventID=1), insert two transactions.
-- For "Conference" (eventID=2), insert one transaction.
-- "Workshop" (eventID=3) will have no transactions.
INSERT INTO Transactions (eventID, userID, method_of_payment, amount)
VALUES
  (1, 1, 'Credit Card', 50.00),
  (1, 2, 'Debit Card', 100.00),
  (2, 1, 'Wallet', 150.00);

-- Insert Complaint records
-- For "Concert" (eventID=1), insert two complaints.
-- For "Conference" (eventID=2), insert one complaint.
-- "Workshop" (eventID=3) will have no complaints.
INSERT INTO Complaint (eventID, userID, staffID, Description, Status, Category)
VALUES
  (1, 1, 1, 'Complaint 1 for Concert', 'Pending', 'Event Issues'),
  (1, 2, 1, 'Complaint 2 for Concert', 'Pending', 'Ticketing Problems'),
  (2, 1, 1, 'Complaint for Conference', 'Resolved', 'Service & Hospitality');

```

Issue w.r.t above dataset:

When the query is run that joins these tables, the total transaction amount for the "Concert" event is wrong (inflated due to the multiplication effect), while the other events show correct values.

Explanation

- Event 1 ("Concert"):
 - Registration: Two rows
 - Transactions: Two rows (50 and 100)
 - Complaints: Two rows
 - Issue: When joining these tables directly, the join creates $2 \times 2 \times 2 = 8$ rows, which causes the SUM(t.amount) to be inflated (50 and 100 values repeated).
- Event 2 ("Conference"):
 - One row in each related table. No duplication effect occurs.

- Event 3 ("Workshop"):

Has no rows in the Registration, Transactions, or Complaint tables, so it will be listed with 0's in the final report.

The query is not fully correct for the task. Here are the main issues:

1. **Multiplicative Row Effect:**
Because you are joining three related tables (Registrations, Transactions, and Complaints) directly to Events, if an event has multiple rows in these tables, the join creates a multiplicative effect. Even though you're using COUNT(DISTINCT ...) for registrations and complaints, the SUM(t.amount) is applied to every joined row. For example, if an event has 2 registrations, 2 transactions, and 1 complaint, the join produces $2 \times 2 \times 1 = 4$ rows, causing the transaction amounts to be summed multiple times (inflating the total).
2. **Table Name Mismatches:**
Your schema defines the table as Registration (singular) and Complaint (singular), but the query uses plural names (Registrations, Complaints). This might cause errors unless you have synonyms or views.

Query 2:

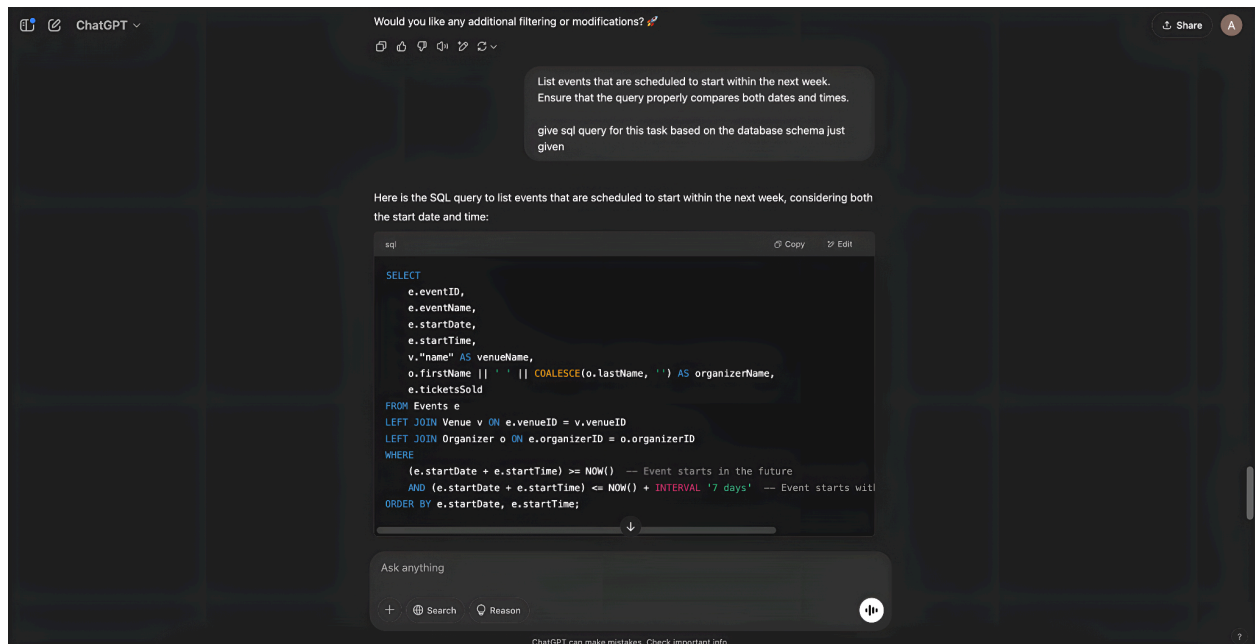
Task:

List events that are scheduled to start within the next week. Ensure that the query properly compares both dates and times.

SQL query given by ChatGPT:

```
SELECT
  e.eventID,
  e.eventName,
  e.startDate,
  e.startTime,
  v."name" AS venueName,
  o.firstName || ' ' || COALESCE(o.lastName, '') AS organizerName,
  e.ticketsSold
FROM Events e
LEFT JOIN Venue v ON e.venueID = v.venueID
LEFT JOIN Organizer o ON e.organizerID = o.organizerID
WHERE
  (e.startDate + e.startTime) >= NOW() -- Event starts in the future
  AND (e.startDate + e.startTime) <= NOW() + INTERVAL '7 days' -- Event starts within the
next 7 days
ORDER BY e.startDate, e.startTime;
```

Screenshot:



Example Dataset to show the problem:

-- Inserting data into Venue

```
INSERT INTO Venue ("name", street, city, "state", pincode)
```

```
VALUES
```

```
('Main Hall', '123 Main St', 'CityA', 'StateA', '12345');
```

-- Inserting data into Organizer

```
INSERT INTO Organizer (staffID, username, firstName, lastName, emailID, contactNo,  
"password", organization, verificationStatus)
```

```
VALUES
```

```
(1, 'org1', 'Organizer', 'One', 'org1@example.com', '1111111111', 'orgpass123', 'OrgOne Inc.',  
TRUE);
```

-- Inserting data into Events

-- We use NOW() and CURRENT_DATE with INTERVALs to simulate relative dates.

-- Note: Our schema requires startDate to be a DATE and startTime a TIME.

-- The problematic query attempts to combine these with (startDate + startTime).

```
INSERT INTO Events
```

```
(eventName, ticketPrice, category, maxAttendees, created_at, startDate, startTime, endDate,  
endTime, venueID, organizerID, ticketsSold, status)
```

```
VALUES
```

-- Event 1: "Tomorrow Event" – clearly in the future (within 7 days)

```
('Tomorrow Event', 50.00, 'Music', 100, NOW(), CURRENT_DATE + INTERVAL '1 day',  
'10:00:00', CURRENT_DATE + INTERVAL '1 day', '12:00:00', 1, 1, 50, 'Upcoming'),
```

-- Event 2: "In 5 Days Event" – within the next week

```
('In 5 Days Event', 70.00, 'Theater', 150, NOW(), CURRENT_DATE + INTERVAL '5 day',  
'15:00:00', CURRENT_DATE + INTERVAL '5 day', '18:00:00', 1, 1, 75, 'Upcoming'),
```

-- Event 3: "Past Event" – in the past (should not be returned)

```
('Past Event', 60.00, 'Conference', 200, NOW(), CURRENT_DATE - INTERVAL '1 day',  
'12:00:00', CURRENT_DATE - INTERVAL '1 day', '14:00:00', 1, 1, 100, 'Completed'),
```

-- Event 4: "Beyond Week Event" – beyond 7 days from now (should not be returned)

```
('Beyond Week Event', 80.00, 'Seminar', 120, NOW(), CURRENT_DATE + INTERVAL '8 day',  
'09:00:00', CURRENT_DATE + INTERVAL '8 day', '11:00:00', 1, 1, 90, 'Upcoming'),
```

-- Event 5: "Edge Case Today" – scheduled for later today. Because adding DATE+TIME may behave incorrectly,

-- this event can be misclassified

```
('Edge Case Today', 65.00, 'Workshop', 80, NOW(), CURRENT_DATE, '23:00:00',  
CURRENT_DATE, '23:59:00', 1, 1, 40, 'Upcoming');
```

Issue and Explanation:

Instance where the query fails: Edge Case Today is an event scheduled for later today. Because the query is written as `(e.startDate + e.startTime)`, it relies on an implicit conversion that may not produce a correct `TIMESTAMP`. Depending on your SQL engine's handling, this event might be misclassified (for example, if the addition returns a wrong value, it might not be returned even though it should).

The incorrect query uses `(e.startDate + e.startTime)` to form a timestamp for filtering. In PostgreSQL and many SQL dialects, adding a `DATE` and a `TIME` without explicit conversion is not guaranteed to produce a valid `TIMESTAMP`. This script provides both normal and edge-case data so that when the query is executed, you may observe errors or incorrect results for some events.

Conversation with ChatGPT:

<https://chatgpt.com/share/67dc39bb-9e24-8005-815d-bd481e659158>