DBMS Project Documentation
Synapse Fest Database System

By:

Devdutt Dinesh — 202403007
Vignes Vinod — 202403049

A detailed Software Requirements Specification and Relational Database Schema for the Synapse Annual Techno–Cultural Fest Management System

Contents

Chapter 1

SRS

1.1 Description

1.1.1 Purpose

The objective of this project is to design and maintain a database system for Synapse, the annual techno-cultural fest of Dhirubhai Ambani University (DAU), formerly DA-IICT. The database aims to store, manage, and track all aspects of the fest, including event schedules, venues, performers, ticket sales, attendees, sponsors, and vendors. By creating a centralized system, the fest committee can efficiently plan, execute, and evaluate the success of the fest each year.

The core motivation is to directly combat past problems:

- Preventing Fake Ticket Sales: In the past, fake tickets have been a huge issue for the event. By implementing a secure and verifiable ticketing system, the database will assign a unique ID to each purchased ticket, which can be validated at entry points. Once marked as "sold," the ticket with the same ticket id cannot be bought again from the fest committee unless "cancelled". This will drastically reduce fraudulent activities and ensure accurate revenue tracking.

- Improving Inter-Departmental Communication: Instead of relying on informal chats and calls, the database will provide a structured platform for communication. The "department-directory" will serve as a live directory, ensuring that every department (On-Gate, Production, Hospitality, etc.) has access to the most current and accurate contact information for their counterparts. This will streamline real-time coordination.

- Strengthening Data Integrity and Consistency: Currently, multiple departments handle their own records, which leads to duplication and inconsistencies. The centralized system ensures that all data is stored in a unified way, maintaining accuracy across departments. This will be achieved through features like unique primary keys for each record, relational constraints to prevent duplication, and controlled access so that only authorized users can update specific tables. In addition, automated daily backups and recovery protocols will safeguard critical fest data against unexpected failures.

- Crowd Management: With 40,000+ attendees, one of the biggest challenges has been managing queues, congested entry points, and chaotic movement between events. The database will integrate real-time ticket scanning and instant schedule updates, which help organizers prevent overcrowding, distribute audiences more evenly, and ensure safety. One way to implement this will be through triggers or push notifications.

- Ensuring Financial Transparency: One of the major risks in large-scale student festivals is fund mismanagement or corruption. To address this, the database will include a centralized finance table with role-based access controls. Every transaction — whether sponsorship income, ticket revenue, or expense — will be timestamped and traceable. This creates accountability, prevents misuse of funds, and provides the college administration with a clear audit trail for oversight.

1.1.2 Intended Audience and Reading Suggestions

- Synapse Committee: The primary users. They need a user-friendly interface to manage event creation, scheduling, venue allocation, participant registration, volunteer assignments, and real-time monitoring of ticket sales and on-ground operations. Suggested reading for this audience would be to understand the best practices for data entry and validation to ensure data integrity.
- Sponsors and Vendors: To gain insights into the scale of participation and revenue potential. They don't need access to the full database, but rather specific views that show the value of their involvement.
- Students and Attendees: To access schedules, register for events, purchase tickets, and stay updated.
- Admin: To monitor financials and evaluate the impact of the fest.

1.1.3 Product Scope

Synapse is Gujarat's largest techno-cultural festival, organized annually for 4 days at DAU. It attracts 40,000+ attendees, including students, performers, and industry professionals. The fest features:

- Events: These are a key part of the fest. The database must handle different event types, from individual quizzes to team-based coding challenges. It needs to track team members, judges' scores, and prize distribution for each event.
- Performances: Concerts by prominent artists, dance shows, and fashion shows.
- Exhibitions and Stalls: The database must track vendors, their stall locations, and the products/services they offer. This also ties into the finance department for stall booking fees.

Given the massive scale and diversity of activities, managing operations through manual methods often leads to inefficiency, miscommunication, and risks such as fake ticket sales or poor accommodation tracking. To ensure a smooth and secure experience, our proposed database system will provide a centralized, efficient, and reliable way to handle ticketing, event scheduling, accommodation, event details, and inter-departmental communication.

1.1.4 Description

Stakeholders

- Fest Committee Members: For planning, updating, and monitoring events.
- Sponsors and Vendors: To access participation metrics and visibility opportunities.
- Students & Attendees: To register, purchase passes, and track events of interest.
- College Administration: To oversee finances, safety, and compliance.

Database Overview

The system will consist of interrelated tables covering the following domains:

1. Events Table: Stores event details
2. Participants Table: Records participants, team details, and event enrollment.
3. Judges/Artists Table: Maintains profiles of judges and guest performers.
4. Ticketing Table: Tracks ticket prices, availability, and sales transactions.
5. Attendees Table: Stores attendee details linked with purchased tickets.
6. Vendors Table: Lists food stalls, merchandise outlets, and startup kiosks.
7. Sponsors Table: Records sponsor names, categories, and contribution amounts.
8. Volunteers Table: Assigns volunteers to specific events and roles and provides their phone numbers.
9. Transaction Table: Manages sponsorship funds, ticket revenue, expenses, and budget allocations. Each entry will be linked with unique identifiers (such as transaction ID and timestamp) to ensure accountability.
10. Venue Table: Tracks all venues that events take place in.
11. Merchandise Table: Keeps track of number of purchases, what size each purchase is, and whether the item has been collected.
12. Accommodation table: keeps track of all rooms and all relevant information of the person/people staying at the rooms.
13. Department table/directory: Helps the various departments that are the backbone of the event to help communicate better. Also useful for attendees.
14. Inventory table: To ensure that any inventory, that is either rented or bought, is all archived properly to make sure all rented items are returned or items bought are kept in storage for next year's fest.

Overall Fest Flow and Description

Synapse is the annual techno-cultural fest of Dhirubhai Ambani University (DAU), and for four days the entire campus comes alive with energy, creativity, and celebration. It's one of Gujarat's biggest student-run fests, drawing over 40,000 people – from college students and alumni to industry professionals and cultural enthusiasts.

Daytime Competitions Throughout the day, different clubs at DAU host competitions in academics, technology, and art. Each club is allotted a budget for prize distribution, ensuring fair recognition of winners. For example:

- The Headrush Club hosts challenging quizzes.
- The Debate Society engages students in lively debates, encouraging critical thinking.
- The Press Club curates poetry slams and creative writing contests.

Evening Performances and Showcases As the sun sets, the spotlight shifts to performative and expressive arts. Competitions and showcases include:

- Battle of the Bands, where student bands from various colleges compete with electrifying performances.
- Dance Showcases and Fashion Shows that are judged by industry professionals.
- Drama and Theatre Performances.

Night Shows The nights at Synapse are the most anticipated, as the stage comes alive with renowned artists and performers. In the past, famous artists such as Mohit Chauhan, Shaan, and many DJs have captivated audiences with unforgettable performances. These concerts are the highlight of Synapse, where thousands gather to sing the night away.

Campus Experience and Engagement Beyond competitions and performances, there is a plethora of activities that attendees can engage themselves in:

- Food and Beverage Stalls: The fest hosts a diverse range of food trucks, local eateries, and national chains, catering to all tastes.
- Merchandise and Lifestyle Stalls: Pop-up shops offer clothing, jewelry, accessories, and DAU-branded souvenirs.
- Sponsors are able to showcase their product to a large audience.

The Organizing Committee and Departments Behind the seamless execution of Synapse lies a core organizing committee, which is divided into six key departments. These departments work in close coordination, constantly communicating to ensure the fest runs smoothly:

1. On-Gate Department – Manages security and entry verification at all gates, ensuring that only authorized ticket holders, guests, and performers are admitted. They are also the first point of contact for attendees arriving at the fest.
2. On-Call Department – Acts as the live communication backbone. This team manages urgent requests, coordinates between departments, and ensures that information flows quickly during the fest.
3. Accommodation Department – Handles housing for outstation participants and attendees. Many visiting students and performers stay in DAU's accommodation facilities, which this team organizes, manages, and supports throughout their stay.
4. Production Department – Oversees the stage, lights, sound systems, and overall technical setup for performances and shows. Their role is critical for creating a professional and engaging experience for both artists and the audience.
5. Events Department – Coordinates with clubs and societies to manage the schedule of competitions, ensures judges and participants are present, and monitors the smooth conduct of all daytime and evening events.
6. Hospitality Department – Ensures that performers, judges, sponsors, and special guests are treated with care. They manage logistics such as transportation, refreshments, green rooms, and VIP support.

Usefulness of the database to each department

1. The On-Gate department benefits from the database by being able to make sure the legitimacy of the individuals entering the college during Synapse i.e. making sure the ticket is real and the individual entering is the one associated with the Identifier unique to each ticket.

2. The On-call department will be able to coordinate much more efficiently with the help of the call directory table which helps them keep track of each coordinator and fest committee members, while making sure that other departments also benefit from constant updation of the call directory database.

3. The Accommodation department benefits from the accommodation table, being to keep track who is being accommodated where, while constantly being able to update it in case of changes.

4. The Production department benefits from the inventory table, enabling them to efficiently keep track of the inventory like stage infrastructure, lights and sound systems etc while simultaneously updating the table according to whether the item is in use or packed into storage.

5. The Events Department benefits from the events table, where they are able to track each event and its progress while being able to see details like judges,participants and the host club. This information can be relayed to the On call department to make sure that coordinators/ members can be reassigned from one event to another according to the events and its progress.The will also be able to make sure each event was held fairly via the Judges evaluation table.

6. The Hospitality department benefits from the judges/artist table, by being able to oversee their concerned event timings and necessary arrangements to be made accordingly.This information can also be relayed to the events department and on-call department in case of any delays/issues. This also helps with efficient coordination with concerned persons of contact(POC's).

Together, these departments ensure that every part of Synapse – from competitions to concerts – operates like clockwork. Each department maintains a dedicated contact person with details such as phone number and location, allowing the teams to stay connected and handle issues in real time.

Technical Overview

The database will follow a simple three-layer structure: a user interface (mobile/web for attendees and admins), an application layer to process requests, and a centralized database to store and manage all records. This design ensures that data remains consistent across departments. To handle large crowds, the system will be hosted on reliable servers with backup and recovery features so that operations can continue even if there is a failure.

The system will enforce strong consistency across departments using database transaction isolation levels. Triggers and stored procedures will ensure that updates in one module (e.g., a ticket scanned at the On-Gate department) are instantly reflected in related modules (e.g., attendee entry logs and finance records). This guarantees accuracy and up-to-date information for all departments.

Synapse is truly a special event that allows various students to showcase their talent and improve their skills through healthy competition. It's an event where countless unforgettable memories are forged.

1.2 Fact Finding Phase

1.2.1 Background Readings

Description of Each Reading

- Official Synapse Website (synapse.daiict.ac.in):
  Provided an overview of the scale, events, and structure of Synapse. Helped identify categories of events (competitions, workshops, concerts, exhibitions),and sponsorship needs.
- Database Design in Event Management Systems :
  Highlighted best practices for designing event databases, such as normalization of event data, handling multiple venues, and managing many-to-many relationships between participants and events.
- News Coverage of Past Synapse Fests:
  Gave real-life insights into event flow, large crowd handling, and vendor involvement. Helped identify requirements around ticket validation, security, and vendor management.
- IEEE Software Requirement Specification (SRS) Template Guidelines:
  Used to structure Purpose, Scope, Audience, and Requirements sections formally.

References

- Official DAU synapse website:
  Synapse
- Geeksforgeeks article to design a database for event management:
  How to Design a Database for Event Management - GeeksforGeeks
- Article on synapse by Shiksha:
  Dhirubhai Ambani University (formerly DA-IICT) News & Events - ,Gandhinagar
- SRS iee format template and example:
  IEEE Software Requirements Specification Template
  Software Requirements Specification document with example - Krazytech

1.2.2 Interview

DBMS project: Interview plan

System: Synapse Database
Participants: Arya Murg (Synapse committee member & former volunteer)
Devdutt Dinesh
Vignes Vinod
Date: 1/9/2025 Time: 9:00 pm
Duration: 15 mins Venue: Canteen

Purpose of Interview: To obtain a better Idea regarding the annual fest of synapse from a member of the annual fest committee regarding the logistics and planning behind Synapse. Also to find and understand the experience when they were a volunteer.

Agenda: Overview of Festival Structure
Sponsorship and finance of Synapse
Any difficulties faced by the committee
Fake ticket issue

Documents to be brought to the interview : Not applicable

DBMS project: Interview Summary

System: Synapse Database
Participants: Devdutt Dinesh
Vignes Vinod
Arya Murg (Synapse Committee Member)
Date: 2/9/2025 Time: 9:00 pm
Duration: 15 mins Venue: Canteen

Purpose of Interview: To obtain a better Idea regarding the annual fest of synapse from a member of the annual fest committee regarding the logistics and planning behind Synapse.

1. The festival structure is formed by annual fest committee core members and extended core-members, forming 6 Departments covering each and every aspect of organizing a fest of this magnitude, multitude and size. This includes the On-Gate, On-call, accomodation, production, events and Hospitality departments.

2. Sponsors are classified into categories based on their level of exclusivity and brand exposure. The primary categories include Title, Co-Title, Platinum, and Associate Sponsors. Additionally, sponsors may opt to distinguish themselves by their specific domain or industry. For example, a bank may choose to be recognized as the "Banking Partner" instead of being placed under the standard sponsorship categories.

3. There are majorly 2 issues faced when organizing an event of this scale – chaotic communication and fake tickets. Tickets printed have proper identifiers as in unique codes to make sure fake tickets are not being sold. No mitigation measures have been introduced as such to combat the chaotic communication/ miscommunication.

4. All the data regarding synapse is stored in excel sheets. There is no additional security measures taken.

1.2.3 Questionnaire

Questionnaire (Google form):

Name*:
Phone number*:
Student ID Number (if DAU student):

1. Have you attended Synapse before?*

• Yes
• No

2. If yes, how many times have you attended?*

• 1 time
• 2-3 times
• More than 3 times

3. Which category best describes you?*

• Synapse Committee member
• DAU student (if not synapse member)
• Alumni
• Faculty
• Guest/Judge

4. Which part of Synapse do you enjoy the most?*

Day Competitions (Quizzes, Debates, Arts, etc.)
Evening Performances (Dance, Fashion Shows)
Night Concerts (Artists/DJs)
Food & Merchandise Stalls
Other: __________

5. During the fest, how do you usually get updates about schedules or competitions?*

Word of mouth
WhatsApp groups
Official Website
Social Media

6. Have you ever faced issue with ticketing or entry at Synapse *

• Yes
• No
If yes, please specify briefly:

7. How important would you rate the following for a smoother Synapse experience?*

(1 = Not Important, 5 = Very Important)
1 2 3 4 5
Secure Ticketing System: * * * * *
Accommodation Management: * * * * *
Real-Time Event Updates: * * * * *
Clear Contact Information of Committee: * * * * *

8. If you stayed in Synapse accommodation, how would you rate your experience?*

- Remarkable
- Good
- Average
- Poor
- Did not use accommodation

9. What do you think is the biggest problem in organizing or attending Synapse?*

10. What features would you like to see in a database system that manages Synapse?*

Summary of Questionnaire

1. Most people who filled the form were DAU students
2. Most of them have attended from 1 to 3 times
3. The most enjoyable part of synapse according to most people are night concerts and the various food stalls and vendors.
4. Almost everyone gets updates about events and performances through word of mouth or social media like instagram.
5. There hasn't been much issue of entry but that might be because the majority who filled this Questionnaire were DAU students.
6. Most people find the real time updates of events and performances and also clear contact information of committee members as the important aspects needed for smooth running of Synapse.
7. The accommodation isn't up to standard.
8. Most people have the sentiment that the biggest issue of attending Synapse is the crowd and how chaotic it is.

1.2.4 Observations

Attending Synapse in person is a one-of-a-kind experience. The campus becomes a gateway to an event filled with lights, music, and excitement. Students, performers, and guests all contribute to an atmosphere that feels larger than life. For many attendees, Synapse is not just an event but a memory-making experience — late-night concerts, cheering crowds, and lively competitions create stories people carry for a long time. However, the scale of the event also brings operational challenges.

Requirements gathered:

- Crowd Management Issues: Entry points and ticket checks often become congested due to the high volume of attendees.
- Chaotic Movement Between Events: Navigating across different venues on campus becomes difficult, leading to delays and missed experiences.
- Communication Gaps: Participants sometimes lack access to updated schedules, locations, or emergency contacts, making coordination harder.
- Exhausting for Staff and Attendees: Volunteers and departments spend extra time handling repetitive tasks that could be streamlined with better systems.

1.2.5 Fact Finding Chart

OBJECTIVE	**TECHNIQUE**	**SUBJECT**	**TIME COMMITMENT**
Background info on Synapse News article	Background reading 0.5 day	Official event website	
Scope of the event	Background reading	Official event website	2 hours
Event operations	Interview	Committee member	1 hour
Event structure	Background reading/observations	Social media	0.5 day
To establish databases required Committee member	Background reading/interview 3 hours	Official event website	
Issue to be solved	Interview	Committee member	1 hour
Overall experience of attendees/members	Questionnaire/ Interview	Attendees and committee members	2 hour
Pricing of tickets and accomodation	Interview	Committee member	1 hour
Types of sponsors Observations	Interview/ Committee member	1.5 hours	
Mobile interface needs	Observations	Attendees	1 hour
Understand volunteer duties and challenges	Interview	Former volunteer	1 hour
How data is documented for previous fests	Interview	Committee member	1 hour

1.3 Requirements

1.3.1 Functional Requirements

- Store and retrieve detailed event information covering competitions, performances, exhibitions, and stalls.
- Track event participants including individual and team details, registrations, and volunteer assignments.
- Implement a secure and unique ticketing system to prevent fake ticket sales and accurately track ticket transactions.
- Maintain attendee information linked to ticket purchases for verification and communication.
- Store and track judges' and artists' profiles along with event schedules and scoring evaluations.
- Manage sponsors, including categories, brand exposure levels, and contribution amounts.
- Track vendor details, stall locations, and products or services offered.

- Manage finance data covering sponsorship funds, ticket revenue, expenses, and budgeting.
- Maintain merchandise sales tracking including quantity, sizes, and collection status.
- Accommodation management to track room assignments and occupant details.
- Facilitate real-time inter-departmental communication via contact directory for various departments (On-Gate, Production, Hospitality, etc.).
- Provide backup and recovery functionality to restore data in case of accidental loss, system crash, or malicious activity.
- Archive past year's fest data (events, ticket sales, finances, participants) for reference and planning of future editions.

1.3.2 Non-Functional Requirements

- Ensure scalability to handle 40,000+ attendees with high concurrency during sales and events. The system should support horizontal scaling using distributed database technologies or cloud-native services to accommodate peak loads up to and exceeding 40,000 concurrent users, especially during ticket sales and entry check-in periods.
- Guarantee data security, encrypt sensitive information, and implement access control so volunteers, attendees, and sponsors see only appropriate data.
- Deliver a user-friendly administrative dashboard for committee members and a clean public-facing interface for attendees.
- The mobile interface will be designed with responsive web principles ensuring compatibility across Android and iOS devices, enabling smooth performance under heavy concurrent mobile connections. The backend APIs must be optimized for low latency and efficient data transfer to mobile clients to support real-time event updates and ticket validation.
- Maintain reliability and consistency of the database system to avoid failures during the fest.
- Support real-time updates for event schedules, contacts, and notifications to enhance crowd and chaos management.
- The system must maintain archives of past fests for long-term use and allow quick recovery from unexpected failures.
- Security is very important. We will protect sensitive data by encrypting it, both when it's stored and when it's sent over the internet. We will also limit who can see or change data based on their role, like volunteer, attendee, or admin.
- Testing & Deployment: Critical modules (such as ticketing, transactions, and real-time updates) will undergo automated unit, integration, and stress testing to ensure stability during peak usage periods. Load testing will also be performed to verify that the system can support 40,000+ concurrent users without performance degradation.

Key Issues to Address (based on interviews and observations):

- Combat fake ticket sales using a unique ID system per ticket.
- Improve chaotic communication by providing a structured, live contact directory.
- Streamline coordination between departments to manage large crowds and event movement.

- Enhance accommodation quality tracking and management..
- Make sure all finances are transparent to prevent financial fraud by committee members.
- To prevent losing important data, regular backups will be taken automatically. Some backups will save only changes made since the last backup, while others will save everything, stored safely both locally and on secure cloud servers. We will regularly test to make sure backups work correctly and can be restored quickly in an emergency, so the festival can continue without issues.

1.4 User Classes and Characteristics

USER ROLE	**USE OF DATABASE**
Synapse Committee Members	Full access to event schedules, participant registration, ticketing, accommodation, volunteers, vendors, and sponsors. Can add, update, and delete records (CRUD). Generate detailed financial and operational reports.
Sponsors	Access depends on contribution tier: Title/Co-title sponsor: View comprehensive reports (brand exposure metrics, ticket sales numbers, event reach). Platinum Sponsors: View limited reports related to their sponsored events or branding. Associate Sponsors: View only acknowledgment details (where their brand/logo appears). No modification rights.
Vendors	Manage their stall details (location, category, items/services offered). View financial dues and settlement records linked to their stall.
Students and Attendees	Register for events, purchase tickets, and view schedules. Access only their own tickets, accommodation bookings, and merchandise purchases. No modification privileges outside their own data.
Volunteers	View duty matters, event assignments, and committee contact directory. Limited update rights
Judges / Artists / Performers	Judges: Input evaluation/score sheets directly into the database for assigned events. Artists/Performers: View their performance schedule, accommodation details, and hospitality contacts.
College Admin	Monitor overall finances, sponsorship data, and generate reports for compliance and safety but do not manage day-to-day event details.
System Administrator / IT Support	Full control over the database including managing user accounts, data security, backups, and system maintenance. No role in event/finance decisions, purely technical oversight.

1.5 Operating environment

i. Hardware/Software requirements

Hardware:

- Any standard PC or laptop for committee/admin use.
- Smartphones (Android/iOS) for attendees and volunteers to access schedules, updates, and ticket validation.

Software:

- Database: MySQL / PostgreSQL / Oracle DB for backend storage.
- Mobile Interface: Progressive Web App (PWA) or mobile-friendly website for attendees to view schedules, tickets, and announcements.

Operating System:

- Server: Linux (preferred for hosting, stability, and scalability).
- Client/Admin: Windows/Linux for committee PCs.
- Mobile: Android/iOS for attendees and volunteers.

Connectivity: Reliable Wi-Fi/4G/5G across campus during the fest.

ii. External Interface requirements

Input Interfaces:

- Admin Panel: SQL queries, database dashboards, and data entry forms
- Attendee Interface: Web/mobile forms for ticket purchase, accommodation booking, and event registration.
- Volunteer Interface: Mobile access for schedules, duty rosters, and live updates.

Output Interfaces:

- Query results and structured reports.
- Real-time notifications and event updates sent to attendee mobile devices.
- Accommodation allocations displayed for attendees.
- Judges' scoring outputs for event evaluations.

Possible Third-Party Tools/APIs:

- Google Forms (used for Questionnaire).
- WhatsApp/Email for sending schedule updates, emergency alerts, or confirmations.

1.6 Product functions

- CRUD Operations:
  - Add, update, and delete attendee records, event schedules, and accommodation details.
  - Maintain records of clubs, events, and prize allocations.
- Triggers:
  - Auto-update seat availability when tickets are booked or cancelled.
  - Auto-notify concerned committee members when accommodation capacity is nearing full.
- Views:
  - Public view of events and schedules accessible to attendees.
  - Views of accommodation and contact directory are accessible for attendees, judges and performers.
- Stored Procedures:
  - Calculate total prize distribution per club.
  - Generate reports for ticket sales, event participation, and accommodation usage.
  - Validate attendee entry against registered tickets.

1.7 Privileges

Function	**Committee**	**Volunteer**	**Attendee/ Student**	**Sponsor/ Vendor**	**Judge/ Performer**
Add/ Delete Users	Yes	No	No	No	No
Create/ Update Event Details	Yes	Yes	View only	View only	View only
Manage Ticketing	Yes	Yes	No	No	No
Manage Accommodation	Yes	Yes	View only	No	View only
View Full Schedule	Yes	Yes	Yes	Yes	Yes
Access Finance Table	Yes	No	No	Limited view	No
Judges' Evaluation Entry	Yes	No	No	No	Yes
Generate Reports	Yes	No	No	Limited view	No
View/Update Contact Directory	Yes	Yes	View only	View only	View only

1.8 Assumptions

- Attendees, judges, performers and everyone else are assumed to have access to a smartphone to check schedules, ticket details, or accommodation information via the system's mobile-accessible interface.
- Users (committee members, volunteers, and attendees) are assumed to have basic computer and smartphone skills, sufficient to interact with forms, dashboards, and mobile-friendly web pages.
- A stable internet connection is available to enable access to the centralized database and event tables remotely through laptops or mobile devices.
- All event departments (Gate, On-Call, Accommodation, Production, Events, Hospitality) will cooperate and update their data regularly to ensure smooth fest operations.

1.9 Business Constraints

- Time Constraint: The database system must be designed and implemented within the yearly timeline, limiting the scope for advanced features or long-term testing.
- Cost Constraint: The system must rely on free and open-source technologies (e.g., MySQL, PostgreSQL) as no budget is allocated for proprietary software or licenses.
- Skill Constraint: Development is restricted to the knowledge base of students from the DBMS course, which limits the use of advanced tools or enterprise-level solutions. The system must be simple enough for students to use, but also robust enough to provide transparency to reduce chances of corruption or financial mismanagement.
- Fest-Specific Constraint: The system must be capable of handling large attendee traffic and high-volume ticket validation during peak fest hours, but does not need to be maintained throughout the year once the event concludes.

Chapter 2

ER Diagram

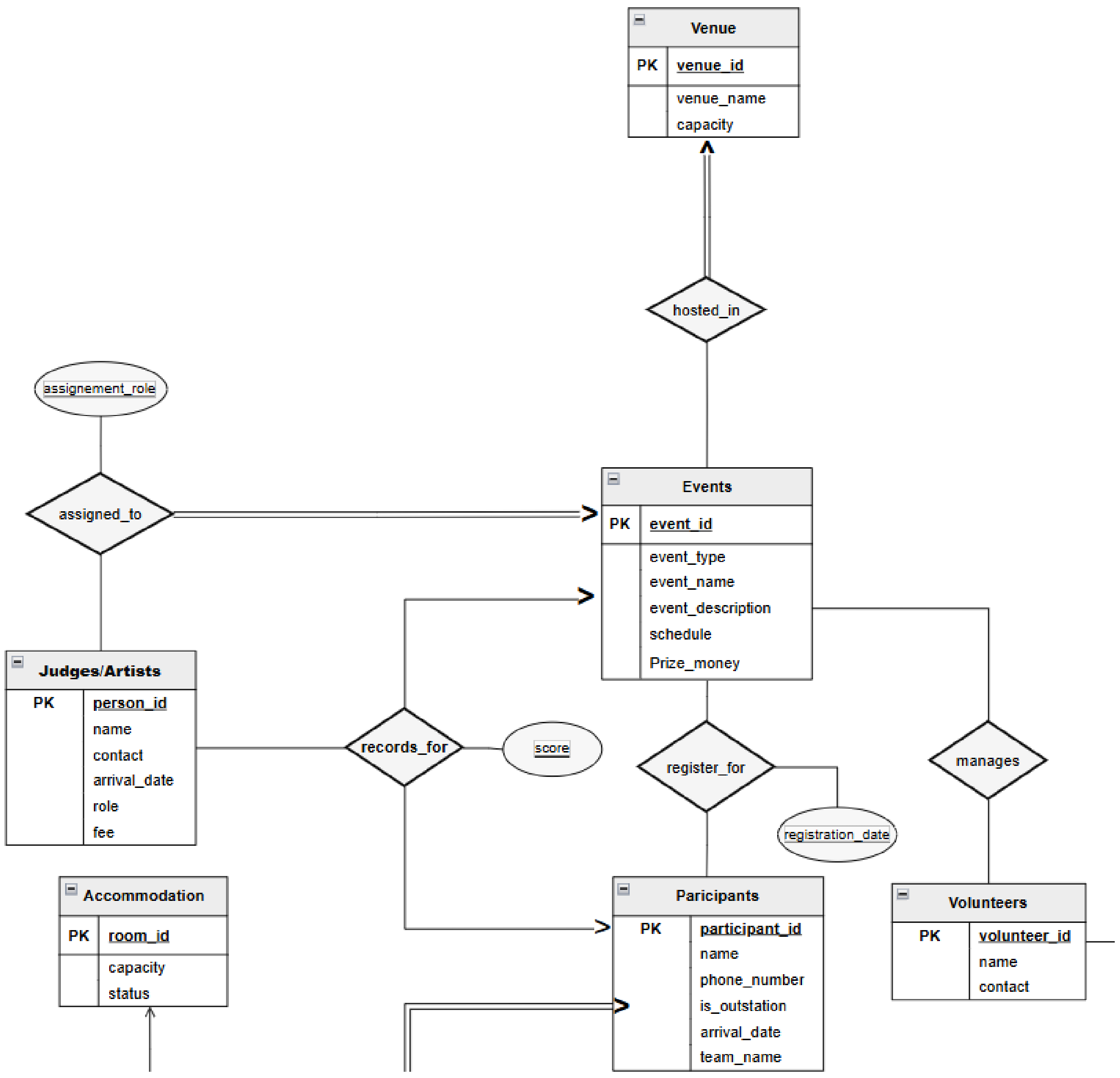


Figure 2.1: ER Diagram (Part 1)

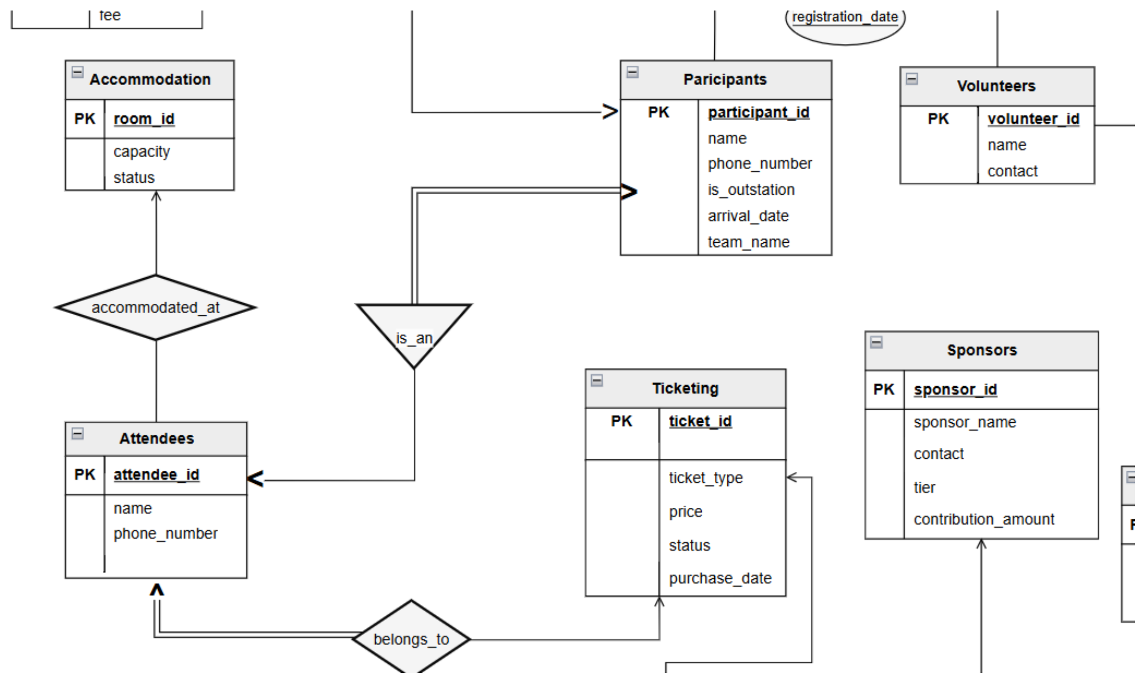


Figure 2.2: ER Diagram (Part 2)

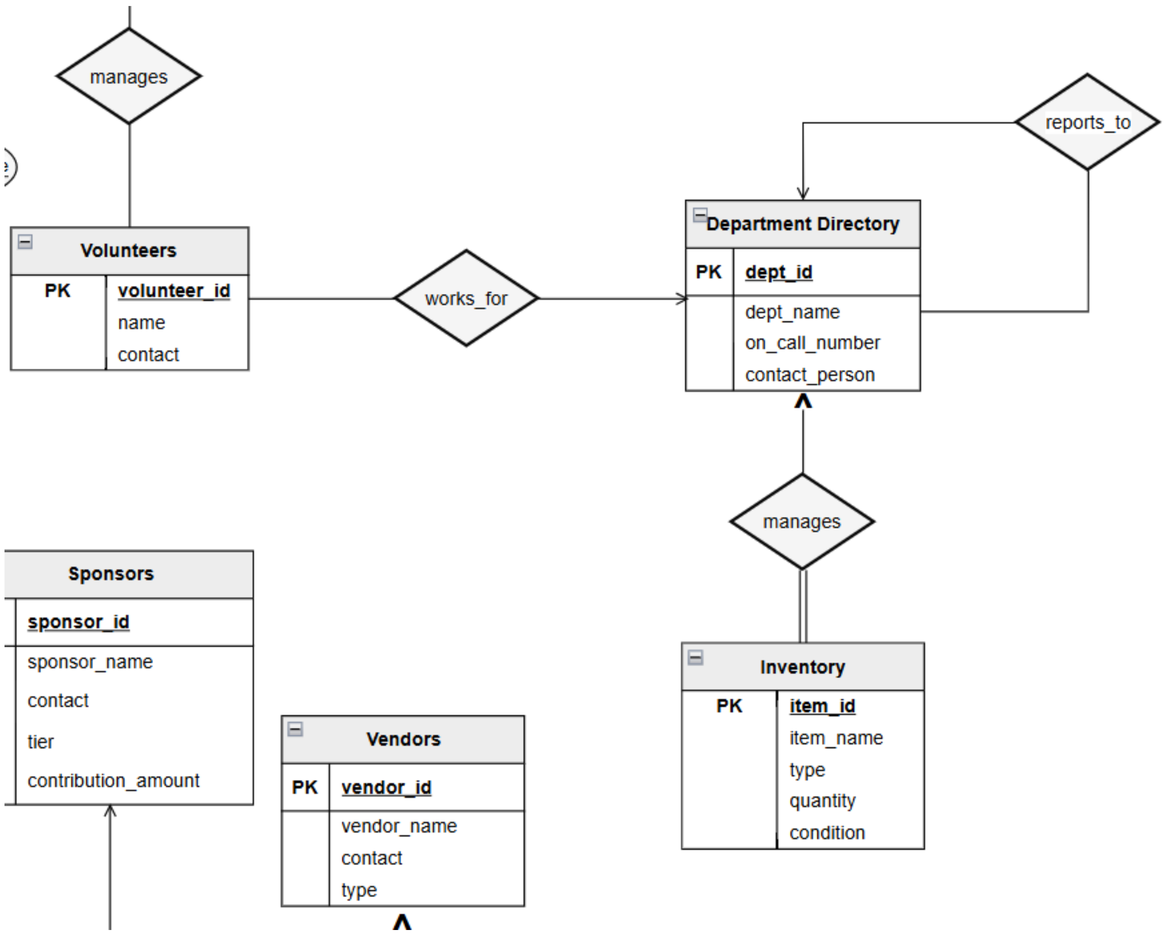


Figure 2.3: ER Diagram (Part 3)

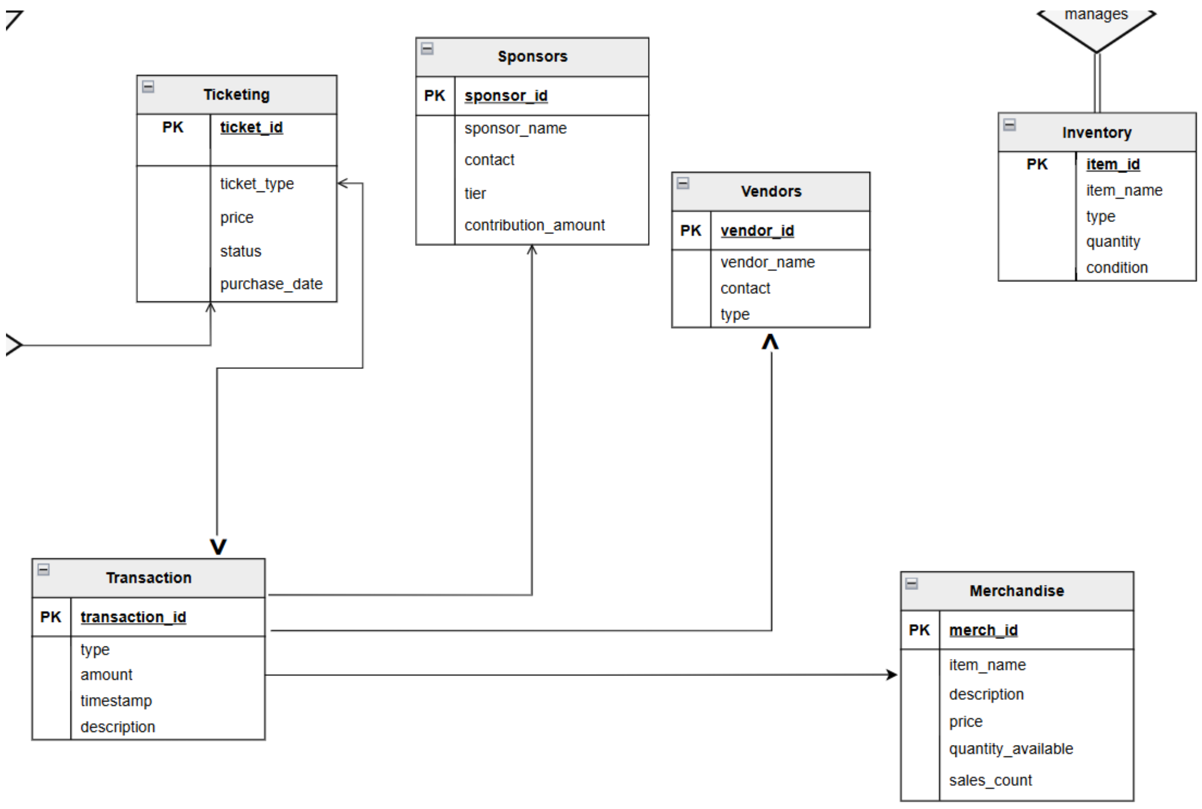


Figure 2.4: ER Diagram (Part 4)

You can access the full ER Diagram here: **View ER Diagram**

Chapter 3

Relational Schema

3.1 Normalised Relational Tables

The following relational schemas are already in normalized form and include all necessary PK, FK, UNIQUE, and CHECK constraints.

Table	**Schema + Constraints**
Attendees	`attendee_id (PK), name, phone_number (UNIQUE), room_id (FK → Accommodation.room_id)`
Participants	`participant_id (PK), attendee_id (FK, UNIQUE), is_outstation, arrival_date, team_name`
Events	`event_id (PK), event_name, event_description, event_type, schedule, prize_money, venue_id (FK → Venue.venue_id)`
Judges/Artists	`person_id (PK), name, contact, arrival_date, role, fee, assignment_role, event_id (FK → Events.event_id)`
Ticketing	`ticket_id (PK), ticket_type, price, status, purchase_date, attendee_id (FK → Attendees.attendee_id)`
Vendors	`vendor_id (PK), vendor_name, contact, type`
Sponsors	`sponsor_id (PK), sponsor_name, contact, tier, contribution_amount`
Volunteers	`volunteer_id (PK), name, contact, dept_id (FK → DepartmentDirectory.dept_id)`
Merchandise	`item_id (PK), item_name, description, price, quantity_available, dept_id (FK → DepartmentDirectory.dept_id)`
Accommodation	`room_id (PK), capacity, status`
DepartmentDirectory	`dept_id (PK), dept_name (UNIQUE), contact_person, on_call_number, reports_to_dept_id (FK → self)`
Inventory	`item_id (PK), item_name, type, quantity, condition, dept_id (FK → DepartmentDirectory.dept_id)`
Transaction	`transaction_id (PK), type, amount, timestamp, description, merch_id (FK), sponsor_id (FK), vendor_id (FK), ticket_id (FK),` *+ XOR source CHECK constraint ensures only 1 source is non-NULL*
Registration	`participant_id, event_id, registration_date (Composite PK, FK constraints)`

Table	Schema + Constraints
JudgesEvaluation	`person_id, participant_id, event_id, score (Composite PK, FK constraints)`
EventManagement	`volunteer_id, event_id (Composite PK, FK constraints)`

Chapter 4

DDL Script

4.1 Database Definition Language (DDL)

The following script defines all relational tables in their normalized form, including all integrity constraints such as **Primary Keys (PK)**, **Foreign Keys (FK)**, **UNIQUE**, and **CHECK** constraints.

```
create table venue (
   venue_id serial primary key,
   venue_name varchar(255) not null,
   capacity int not null check (capacity > 0)
);

create table accommodation (
   room_id serial primary key,
   capacity int not null check (capacity > 0),
   status varchar(50) not null check (status in ('available', 'occupied', 'maintenance'))
);

create table departmentdirectory (
   dept_id serial primary key,
   dept_name varchar(100) not null unique,
   contact_person varchar(100),
   on_call_number varchar(20),
   reports_to_dept_id int
);

create table vendors (
   vendor_id serial primary key,
   vendor_name varchar(255) not null,
   contact varchar(255),
   type varchar(100)
);

create table sponsors (
   sponsor_id serial primary key,
   sponsor_name varchar(255) not null,
   contact varchar(255),
   tier varchar(50) check (tier in ('title', 'cotitle', 'platinum', 'associate')),
   contribution_amount numeric(15, 2) check (contribution_amount >= 0)
);

create table merchandise (
   merch_id serial primary key,
   item_name varchar(255) not null,
   description text,
   price numeric(15, 2) not null check (price >= 0),
   quantity_available int not null check (quantity_available >= 0)
);

create table events (
   event_id serial primary key,
   event_name varchar(255) not null,
   event_description text,
   event_type varchar(100) not null,
   schedule timestamp not null,
   prize_money numeric(15, 2) check (prize_money >= 0),
   venue_id int not null,
   constraint fk_events_venue
       foreign key (venue_id) references venue(event_id) on delete restrict
);

create table attendees (
   attendee_id serial primary key,
   name varchar(255) not null,
   phone_number varchar(20) unique,
   room_id int,
   constraint fk_attendees_accommodation
       foreign key (room_id) references accommodation(room_id) on delete set null
);

create table participants (
   participant_id serial primary key,
   attendee_id int not null unique,
   is_outstation boolean not null default false,
   arrival_date date,
   team_name varchar(255),
   constraint fk_participants_attendees
       foreign key (attendee_id) references attendees(attendee_id) on delete cascade
);

create table judgesartists (
   person_id serial primary key,
   name varchar(255) not null,
   contact varchar(255),
   arrival_date date,
   role varchar(50) not null check (role in ('judge', 'artist')),
   fee numeric(15, 2) check (fee >= 0),
   assignment_role varchar(255),
   event_id int,
   constraint fk_judges_events
       foreign key (event_id) references events(event_id) on delete set null
);

create table ticketing (
   ticket_id serial primary key,
   ticket_type varchar(50) not null check (ticket_type in ('vip', 'general')),
   price numeric(15, 2) not null check (price >= 0),
   status varchar(50) not null check (status in ('sold', 'available', 'cancelled')),
   purchase_date date,
   attendee_id int,
   constraint fk_ticketing_attendees
       foreign key (attendee_id) references attendees(attendee_id) on delete set null
);
create table transaction (
   transaction_id serial primary key,
   type varchar(100) not null check (type in ('sponsorship', 'vendor payment', 'ticket sale
   amount numeric(15, 2) not null,
   timestamp timestamp not null default current_timestamp,
   description text,
   sponsor_id int,
   vendor_id int,
   ticket_id int,
   merch_id int,
   constraint fk_transaction_sponsor
       foreign key (sponsor_id) references sponsors(sponsor_id),
   constraint fk_transaction_vendor
       foreign key (vendor_id) references vendors(vendor_id),
   constraint fk_transaction_ticketing
       foreign key (ticket_id) references ticketing(ticket_id),
   constraint fk_transaction_merchandise
       foreign key (merch_id) references merchandise(merch_id),
   constraint chk_transaction_source
       check (
           (case when sponsor_id is not null then 1 else 0 end) +
           (case when vendor_id is not null then 1 else 0 end) +
           (case when ticket_id is not null then 1 else 0 end) +
           (case when merch_id is not null then 1 else 0 end)
           = 1
       )
);
create table volunteers (
   volunteer_id serial primary key,
   name varchar(255) not null,
   contact varchar(255),
   dept_id int not null,
   constraint fk_volunteers_department
       foreign key (dept_id) references departmentdirectory(dept_id) on delete restrict
);
create table inventory (
   item_id serial primary key,
   item_name varchar(255) not null,
   type varchar(100),
   quantity int not null check (quantity >= 0),
   condition text,
   dept_id int not null,
   constraint fk_inventory_department
       foreign key (dept_id) references departmentdirectory(dept_id) on delete cascade
);
create table registration (
   participant_id int not null,
   event_id int not null,
   registration_date date not null default current_date,
   primary key (participant_id, event_id),
   constraint fk_registration_participants
       foreign key (participant_id) references participants(participant_id) on delete cascac
   constraint fk_registration_events
       foreign key (event_id) references events(event_id) on delete cascade
);
create table judgesevaluation (
   person_id int not null,
   participant_id int not null,
   event_id int not null,
   score numeric(5, 2) check (score >= 0 and score <= 100),
   primary key (person_id, participant_id, event_id),
   constraint fk_evaluation_judges
       foreign key (person_id) references judgesartists(person_id) on delete cascade,
   constraint fk_evaluation_participants
       foreign key (participant_id) references participants(participant_id) on delete cascac
   constraint fk_evaluation_events
       foreign key (event_id) references events(event_id) on delete cascade
);
create table eventmanagement (
   volunteer_id int not null,
   event_id int not null,
   primary key (volunteer_id, event_id),
   constraint fk_management_volunteers
       foreign key (volunteer_id) references volunteers(volunteer_id) on delete cascade,
   constraint fk_management_events
       foreign key (event_id) references events(event_id) on delete cascade
);
alter table departmentdirectory
add constraint fk_dept_reports_to
foreign key (reports_to_dept_id)
references departmentdirectory(dept_id)
on delete set null;
```

(Continue to the next chapter for database queries and implementation)

Chapter 5

Queries

This chapter presents the SQL queries designed for the Synapse fest database, along with their corresponding outputs. The queries are grouped into:

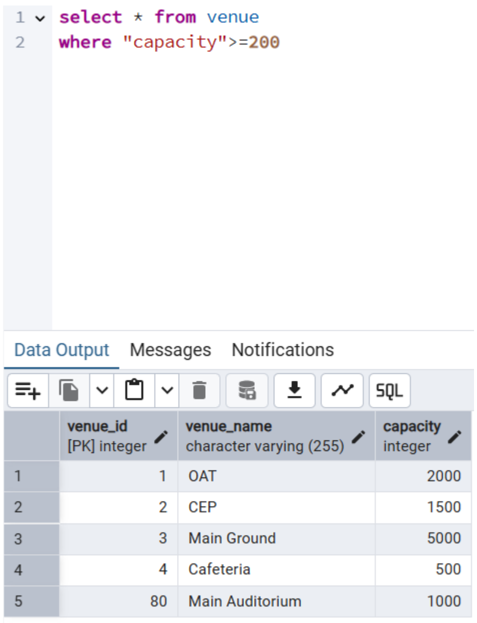
- Simple Queries
- Complex Queries
- Triggers
- Functions

Each subsection states the requirement in words and then shows the SQL (and result) via the attached images exported from the DBMS environment.

5.1 Simple Queries

5.1.1 Query 1: Venues with capacity $\geq$ 200

Requirement: Give all venues with capacity greater than or equal to 200.



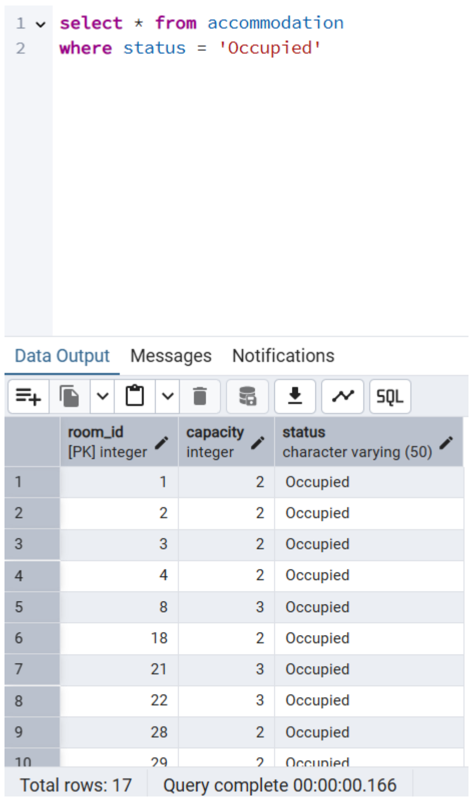
```sql
select * from venue
where "capacity">=200
```

	venue_id [PK] integer	venue_name character varying (255)	capacity integer
1	1	OAT	2000
2	2	CEP	1500
3	3	Main Ground	5000
4	4	Cafeteria	500
5	80	Main Auditorium	1000

Figure 5.1: Simple Query 1: Venues with capacity greater than or equal to 200

5.1.2 Query 2: Occupied rooms from Accommodation

Requirement: Give all tables that are occupied from accommodation.



```sql
select * from accommodation
where status = 'Occupied'
```

Data Output Messages Notifications

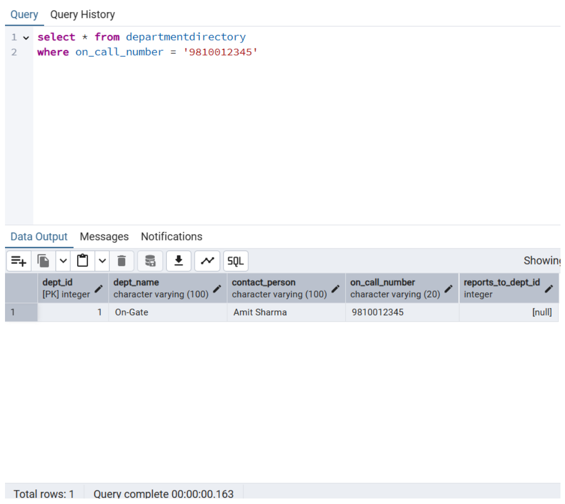
	room_id [PK] integer	capacity integer	status character varying (50)
1	1	2	Occupied
2	2	2	Occupied
3	3	2	Occupied
4	4	2	Occupied
5	8	3	Occupied
6	18	2	Occupied
7	21	3	Occupied
8	22	3	Occupied
9	28	2	Occupied
10	29	2	Occupied

Total rows: 17 Query complete 00:00:00.166

Figure 5.2: Simple Query 2: All occupied rooms from Accommodation

5.1.3 Query 3: Contact person by phone number

Requirement: To find contact person with a specific number '9810012345' from `departmentdirectory`.



Query Query History

```sql
select * from departmentdirectory
where on_call_number = '9810012345'
```

Data Output Messages Notifications

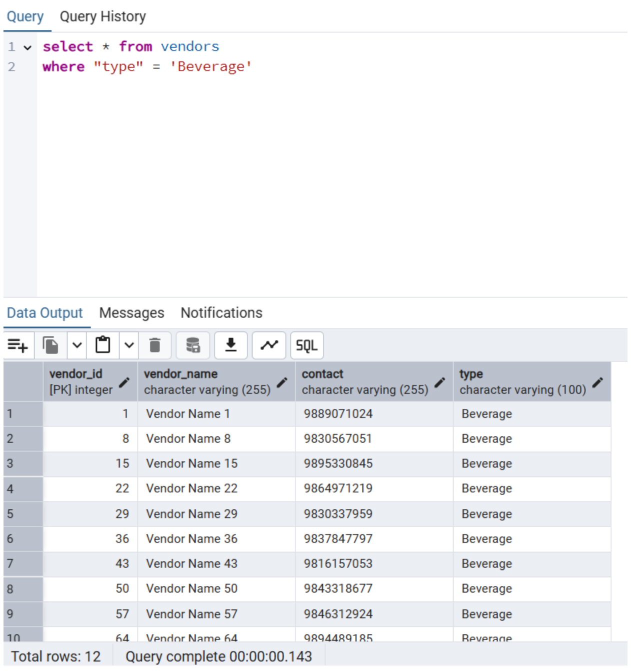
	dept_id [PK] integer	dept_name character varying (100)	contact_person character varying (100)	on_call_number character varying (20)	reports_to_dept_id integer
1	1	On-Gate	Amit Sharma	9810012345	[null]

Total rows: 1 Query complete 00:00:00.163

Figure 5.3: Simple Query 3: Contact person with phone number 9810012345

5.1.4 Query 4: Vendors of type beverage

Requirement: All vendors that belong to the beverage type.



Query Query History

```sql
select * from vendors
where "type" = 'Beverage'
```

Data Output Messages Notifications

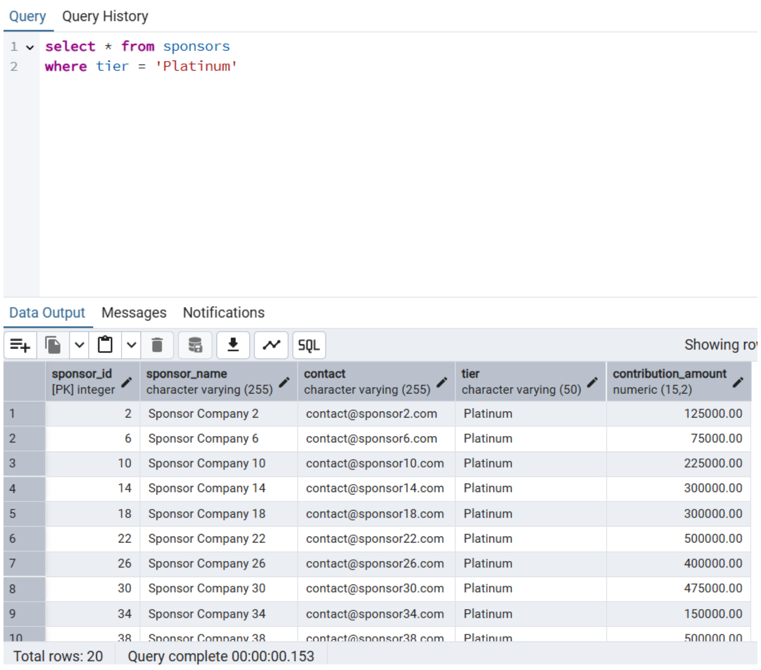
	vendor_id [PK] integer	vendor_name character varying (255)	contact character varying (255)	type character varying (100)
1	1	Vendor Name 1	9889071024	Beverage
2	8	Vendor Name 8	9830567051	Beverage
3	15	Vendor Name 15	9895330845	Beverage
4	22	Vendor Name 22	9864971219	Beverage
5	29	Vendor Name 29	9830337959	Beverage
6	36	Vendor Name 36	9837847797	Beverage
7	43	Vendor Name 43	9816157053	Beverage
8	50	Vendor Name 50	9843318677	Beverage
9	57	Vendor Name 57	9846312924	Beverage
10	64	Vendor Name 64	9894489185	Beverage

Total rows: 12 Query complete 00:00:00.143

Figure 5.4: Simple Query 4: Vendors of type "beverage"

5.1.5 Query 5: All platinum sponsors

Requirement: Give all platinum sponsors.



Query Query History

```sql
select * from sponsors
where tier = 'Platinum'
```

Data Output Messages Notifications

Showing ro

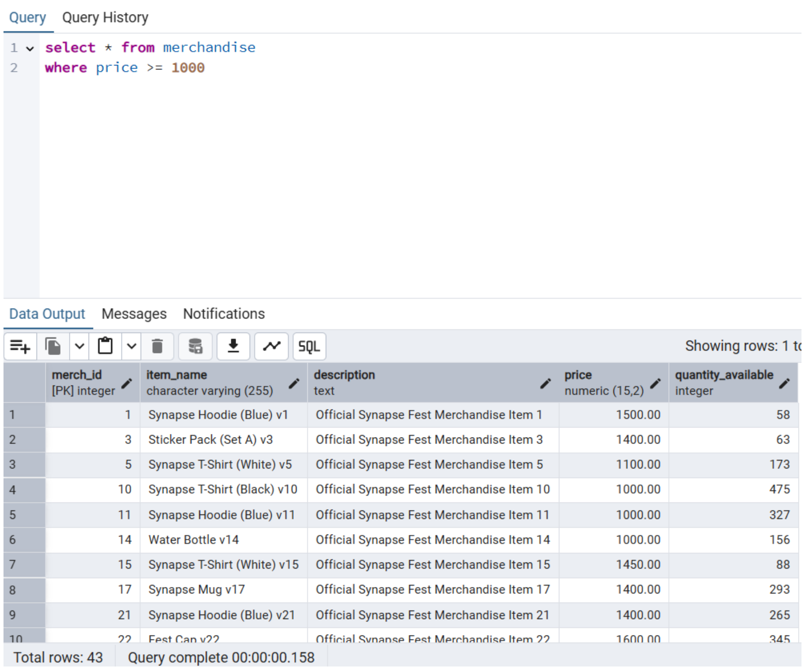
	sponsor_id [PK] integer	sponsor_name character varying (255)	contact character varying (255)	tier character varying (50)	contribution_amount numeric (15,2)
1	2	Sponsor Company 2	contact@sponsor2.com	Platinum	125000.00
2	6	Sponsor Company 6	contact@sponsor6.com	Platinum	75000.00
3	10	Sponsor Company 10	contact@sponsor10.com	Platinum	225000.00
4	14	Sponsor Company 14	contact@sponsor14.com	Platinum	300000.00
5	18	Sponsor Company 18	contact@sponsor18.com	Platinum	300000.00
6	22	Sponsor Company 22	contact@sponsor22.com	Platinum	500000.00
7	26	Sponsor Company 26	contact@sponsor26.com	Platinum	400000.00
8	30	Sponsor Company 30	contact@sponsor30.com	Platinum	475000.00
9	34	Sponsor Company 34	contact@sponsor34.com	Platinum	150000.00
10	38	Sponsor Company 38	contact@sponsor38.com	Platinum	500000.00

Total rows: 20 Query complete 00:00:00.153

Figure 5.5: Simple Query 5: Platinum sponsors

5.1.6 Query 6: Merchandise priced $\geq$ 1000

Requirement: All merchandise above or equal to the price of 1000 rupees.



```sql
select * from merchandise
where price >= 1000
```

Showing rows: 1 t

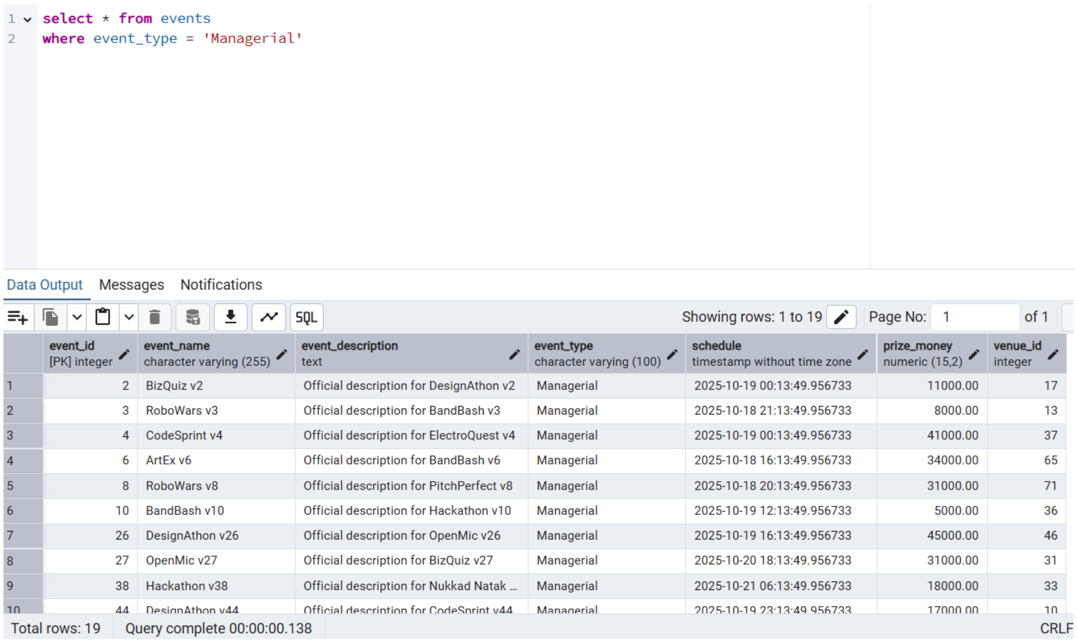
	merch_id [PK] integer	item_name character varying (255)	description text	price numeric (15,2)	quantity_available integer
1	1	Synapse Hoodie (Blue) v1	Official Synapse Fest Merchandise Item 1	1500.00	58
2	3	Sticker Pack (Set A) v3	Official Synapse Fest Merchandise Item 3	1400.00	63
3	5	Synapse T-Shirt (White) v5	Official Synapse Fest Merchandise Item 5	1100.00	173
4	10	Synapse T-Shirt (Black) v10	Official Synapse Fest Merchandise Item 10	1000.00	475
5	11	Synapse Hoodie (Blue) v11	Official Synapse Fest Merchandise Item 11	1000.00	327
6	14	Water Bottle v14	Official Synapse Fest Merchandise Item 14	1000.00	156
7	15	Synapse T-Shirt (White) v15	Official Synapse Fest Merchandise Item 15	1450.00	88
8	17	Synapse Mug v17	Official Synapse Fest Merchandise Item 17	1400.00	293
9	21	Synapse Hoodie (Blue) v21	Official Synapse Fest Merchandise Item 21	1400.00	265
10	22	Fest Cap v22	Official Synapse Fest Merchandise Item 22	1600.00	345

Total rows: 43 Query complete 00:00:00.158

Figure 5.6: Simple Query 6: Merchandise with price $\geq$ 1000

5.1.7 Query 7: Managerial events

Requirement: All events that are managerial in nature.



```sql
select * from events
where event_type = 'Managerial'
```

Showing rows: 1 to 19 Page No: 1 of 1

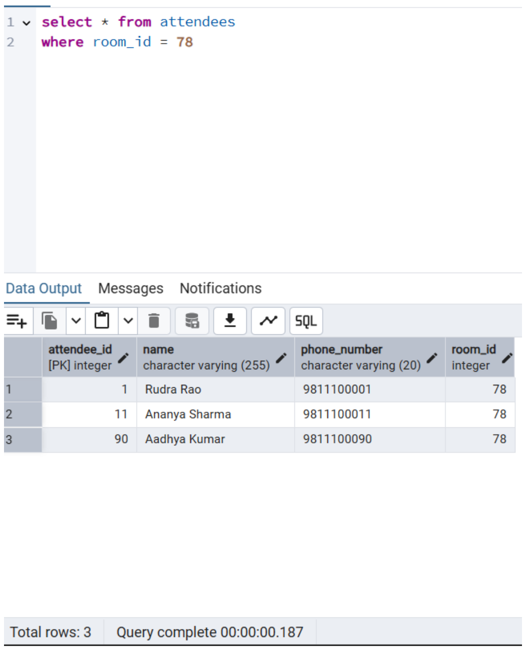
	event_id [PK] integer	event_name character varying (255)	event_description text	event_type character varying (100)	schedule timestamp without time zone	prize_money numeric (15,2)	venue_id integer
1	2	BizQuiz v2	Official description for DesignAthon v2	Managerial	2025-10-19 00:13:49.956733	11000.00	17
2	3	RoboWars v3	Official description for BandBash v3	Managerial	2025-10-18 21:13:49.956733	8000.00	13
3	4	CodeSprint v4	Official description for ElectroQuest v4	Managerial	2025-10-19 00:13:49.956733	41000.00	37
4	6	ArtEx v6	Official description for BandBash v6	Managerial	2025-10-18 16:13:49.956733	34000.00	65
5	8	RoboWars v8	Official description for PitchPerfect v8	Managerial	2025-10-18 20:13:49.956733	31000.00	71
6	10	BandBash v10	Official description for Hackathon v10	Managerial	2025-10-19 12:13:49.956733	5000.00	36
7	26	DesignAthon v26	Official description for OpenMic v26	Managerial	2025-10-19 16:13:49.956733	45000.00	46
8	27	OpenMic v27	Official description for BizQuiz v27	Managerial	2025-10-20 18:13:49.956733	31000.00	31
9	38	Hackathon v38	Official description for Nukkad Natak ...	Managerial	2025-10-21 06:13:49.956733	18000.00	33
10	44	DesignAthon v44	Official description for CodeSprint v44	Managerial	2025-10-19 23:13:49.956733	17000.00	10

Total rows: 19 Query complete 00:00:00.138 CRLF

Figure 5.7: Simple Query 7: Events that are managerial in nature

5.1.8 Query 8: Attendees in room 78

Requirement: All attendees that live in the accommodation with `room_id` 78.



```
select * from attendees
where room_id = 78
```

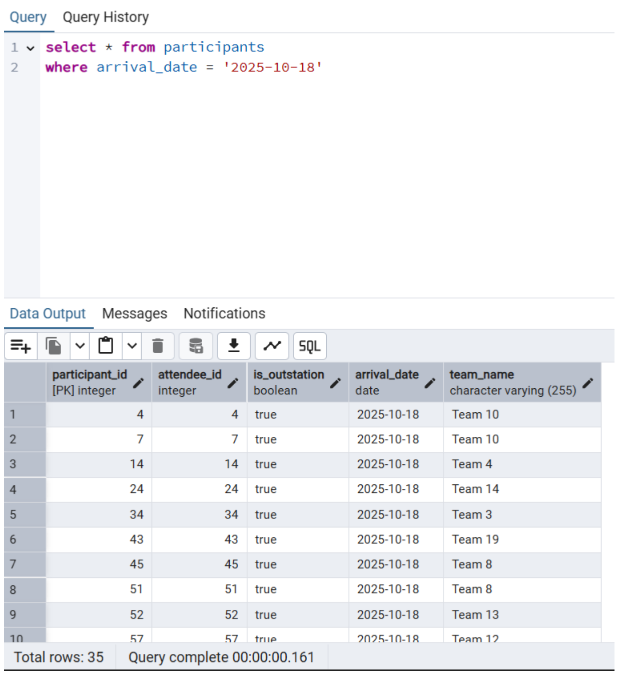
	attendee_id [PK] integer	name character varying (255)	phone_number character varying (20)	room_id integer
1	1	Rudra Rao	9811100001	78
2	11	Ananya Sharma	9811100011	78
3	90	Aadhya Kumar	9811100090	78

```
Total rows: 3    Query complete 00:00:00.187
```

Figure 5.8: Simple Query 8: Attendees staying in room 78

5.1.9 Query 9: Participants arrived on 18 October 2025

Requirement: All participants who arrived at 18th of October, 2025.



```
select * from participants
where arrival_date = '2025-10-18'
```

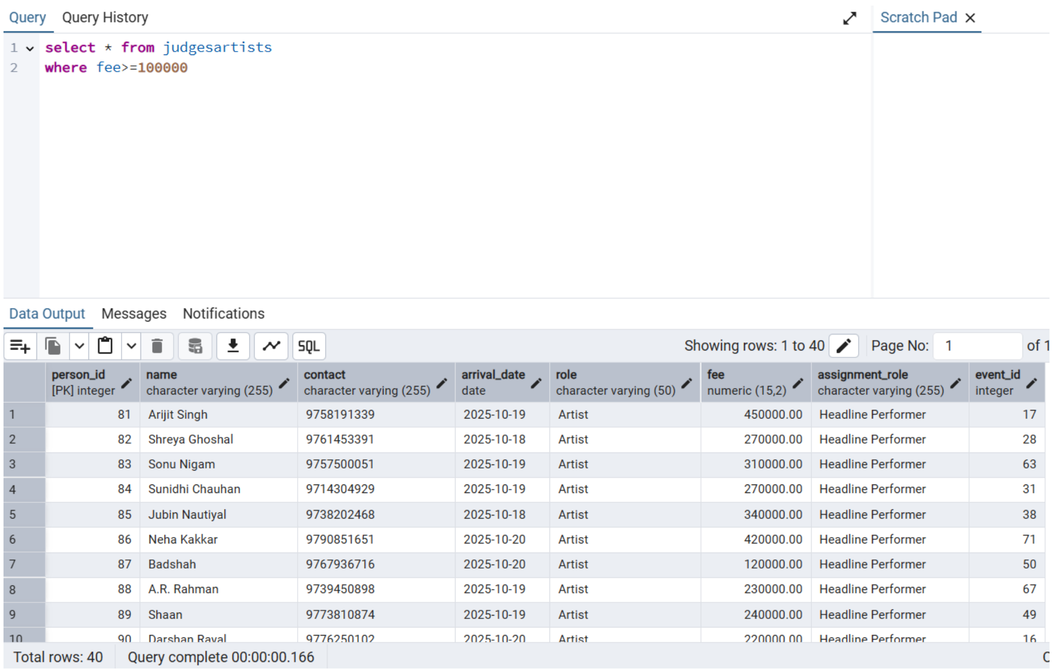
	participant_id [PK] integer	attendee_id integer	is_outstation boolean	arrival_date date	team_name character varying (255)
1	4	4	true	2025-10-18	Team 10
2	7	7	true	2025-10-18	Team 10
3	14	14	true	2025-10-18	Team 4
4	24	24	true	2025-10-18	Team 14
5	34	34	true	2025-10-18	Team 3
6	43	43	true	2025-10-18	Team 19
7	45	45	true	2025-10-18	Team 8
8	51	51	true	2025-10-18	Team 8
9	52	52	true	2025-10-18	Team 13
10	57	57	true	2025-10-18	Team 12

```
Total rows: 35    Query complete 00:00:00.161
```

Figure 5.9: Simple Query 9: Participants who arrived on 18-10-2025

5.1.10 Query 10: Judges and artists with high fee

Requirement: All judges and artists who have a fee above or equal to 100000.



```sql
select * from judgesartists
where fee>=100000
```

Showing rows: 1 to 40 Page No: 1 of 1

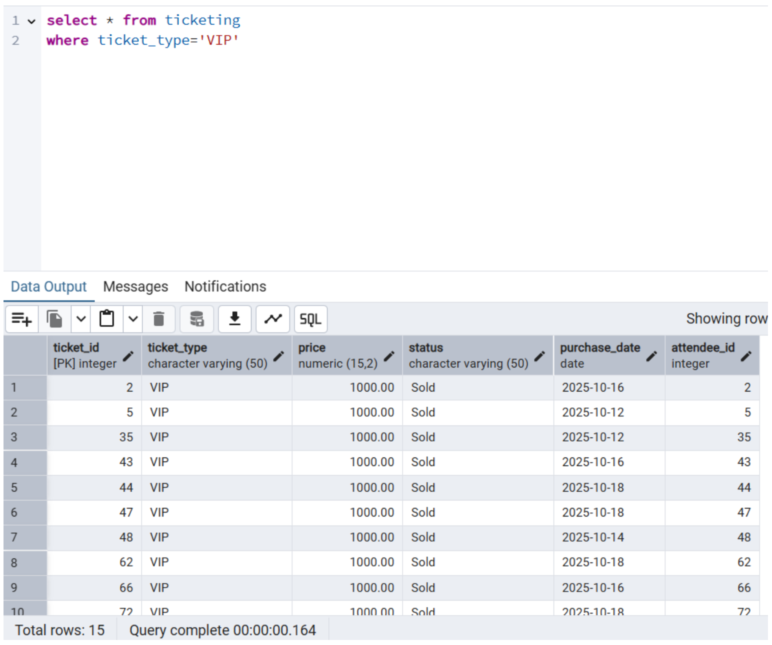
	person_id [PK] integer	name character varying (255)	contact character varying (255)	arrival_date date	role character varying (50)	fee numeric (15,2)	assignment_role character varying (255)	event_id integer
1	81	Arijit Singh	9758191339	2025-10-19	Artist	450000.00	Headline Performer	17
2	82	Shreya Ghoshal	9761453391	2025-10-18	Artist	270000.00	Headline Performer	28
3	83	Sonu Nigam	9757500051	2025-10-19	Artist	310000.00	Headline Performer	63
4	84	Sunidhi Chauhan	9714304929	2025-10-19	Artist	270000.00	Headline Performer	31
5	85	Jubin Nautiyal	9738202468	2025-10-18	Artist	340000.00	Headline Performer	38
6	86	Neha Kakkar	9790851651	2025-10-20	Artist	420000.00	Headline Performer	71
7	87	Badshah	9767936716	2025-10-20	Artist	120000.00	Headline Performer	50
8	88	A.R. Rahman	9739450898	2025-10-19	Artist	230000.00	Headline Performer	67
9	89	Shaan	9773810874	2025-10-19	Artist	240000.00	Headline Performer	49
10	90	Darshan Raval	9776250102	2025-10-20	Artist	220000.00	Headline Performer	16

Total rows: 40 Query complete 00:00:00.166

Figure 5.10: Simple Query 10: Judges/Artists with fee $\geq$ 100000

5.1.11 Query 11: VIP tickets

Requirement: All tickets that are VIP.



```sql
select * from ticketing
where ticket_type='VIP'
```

Showing row

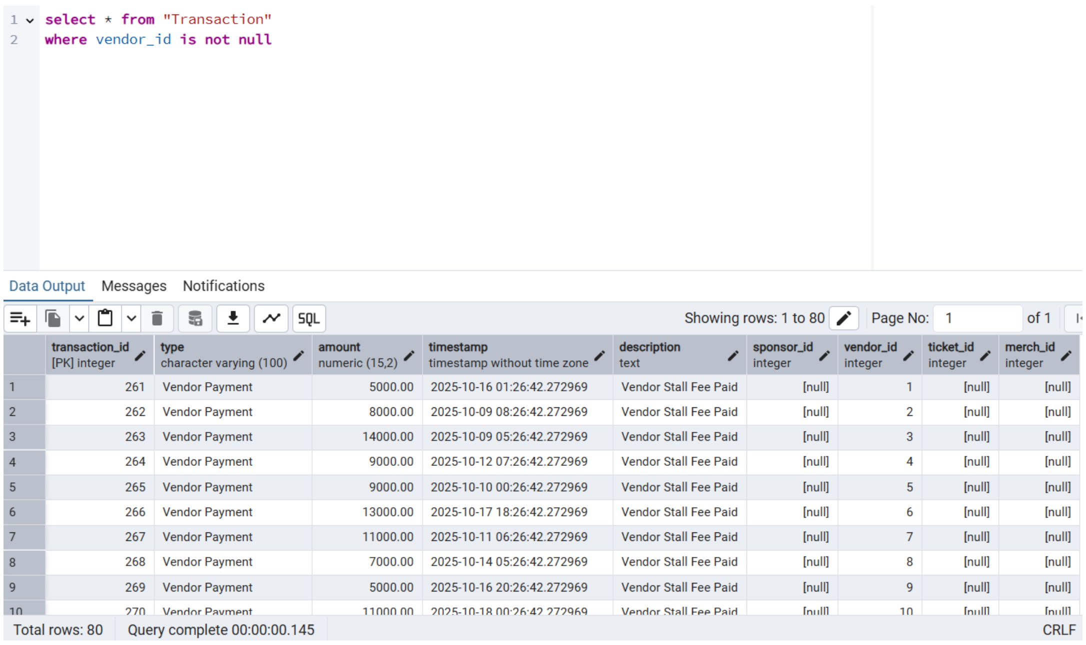
	ticket_id [PK] integer	ticket_type character varying (50)	price numeric (15,2)	status character varying (50)	purchase_date date	attendee_id integer
1	2	VIP	1000.00	Sold	2025-10-16	2
2	5	VIP	1000.00	Sold	2025-10-12	5
3	35	VIP	1000.00	Sold	2025-10-12	35
4	43	VIP	1000.00	Sold	2025-10-16	43
5	44	VIP	1000.00	Sold	2025-10-18	44
6	47	VIP	1000.00	Sold	2025-10-18	47
7	48	VIP	1000.00	Sold	2025-10-14	48
8	62	VIP	1000.00	Sold	2025-10-18	62
9	66	VIP	1000.00	Sold	2025-10-16	66
10	72	VIP	1000.00	Sold	2025-10-18	72

Total rows: 15 Query complete 00:00:00.164

Figure 5.11: Simple Query 11: VIP tickets

5.1.12 Query 12: Vendor transactions

Requirement: Give all vendor transactions.



```
select * from "Transaction"
where vendor_id is not null
```

Data Output Messages Notifications

Showing rows: 1 to 80 Page No: 1 of 1

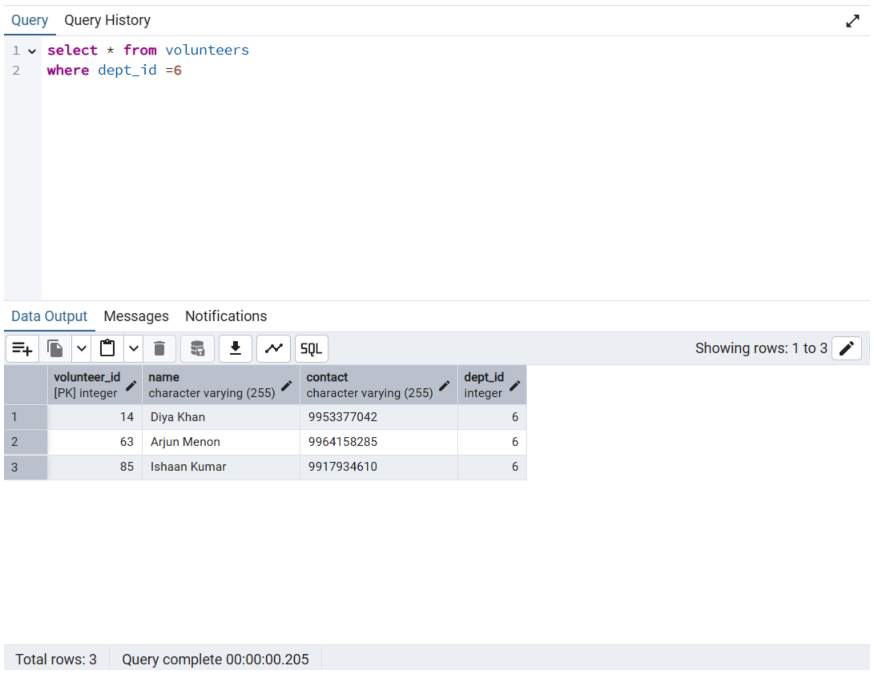
	transaction_id [PK] integer	type character varying (100)	amount numeric (15,2)	timestamp timestamp without time zone	description text	sponsor_id integer	vendor_id integer	ticket_id integer	merch_id integer
1	261	Vendor Payment	5000.00	2025-10-16 01:26:42.272969	Vendor Stall Fee Paid	[null]	1	[null]	[null]
2	262	Vendor Payment	8000.00	2025-10-09 08:26:42.272969	Vendor Stall Fee Paid	[null]	2	[null]	[null]
3	263	Vendor Payment	14000.00	2025-10-09 05:26:42.272969	Vendor Stall Fee Paid	[null]	3	[null]	[null]
4	264	Vendor Payment	9000.00	2025-10-12 07:26:42.272969	Vendor Stall Fee Paid	[null]	4	[null]	[null]
5	265	Vendor Payment	9000.00	2025-10-10 00:26:42.272969	Vendor Stall Fee Paid	[null]	5	[null]	[null]
6	266	Vendor Payment	13000.00	2025-10-17 18:26:42.272969	Vendor Stall Fee Paid	[null]	6	[null]	[null]
7	267	Vendor Payment	11000.00	2025-10-11 06:26:42.272969	Vendor Stall Fee Paid	[null]	7	[null]	[null]
8	268	Vendor Payment	7000.00	2025-10-14 05:26:42.272969	Vendor Stall Fee Paid	[null]	8	[null]	[null]
9	269	Vendor Payment	5000.00	2025-10-16 20:26:42.272969	Vendor Stall Fee Paid	[null]	9	[null]	[null]
10	270	Vendor Payment	11000.00	2025-10-18 00:26:42.272969	Vendor Stall Fee Paid	[null]	10	[null]	[null]

Total rows: 80 Query complete 00:00:00.145 CRLF

Figure 5.12: Simple Query 12: Vendor transactions

5.1.13 Query 13: Volunteers of department 6

Requirement: All volunteers that belong to department 6.



Query Query History

```
select * from volunteers
where dept_id =6
```

Data Output Messages Notifications

Showing rows: 1 to 3

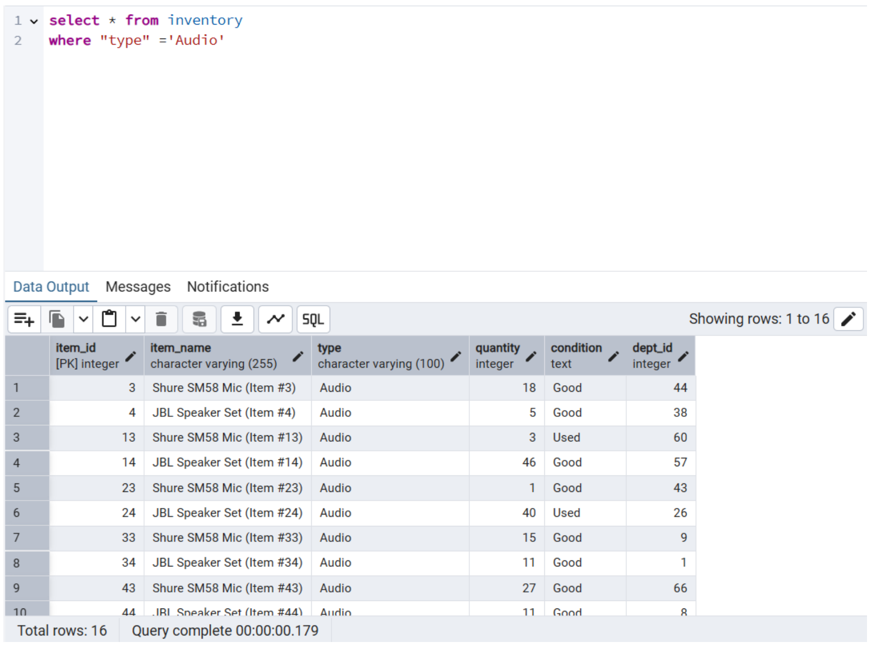
	volunteer_id [PK] integer	name character varying (255)	contact character varying (255)	dept_id integer
1	14	Diya Khan	9953377042	6
2	63	Arjun Menon	9964158285	6
3	85	Ishaan Kumar	9917934610	6

Total rows: 3 Query complete 00:00:00.205

Figure 5.13: Simple Query 13: Volunteers belonging to department 6

5.1.14 Query 14: Inventory items of type audio

Requirement: All inventory items that are of type audio.



```
select * from inventory
where "type" ='Audio'
```

Data Output Messages Notifications

Showing rows: 1 to 16

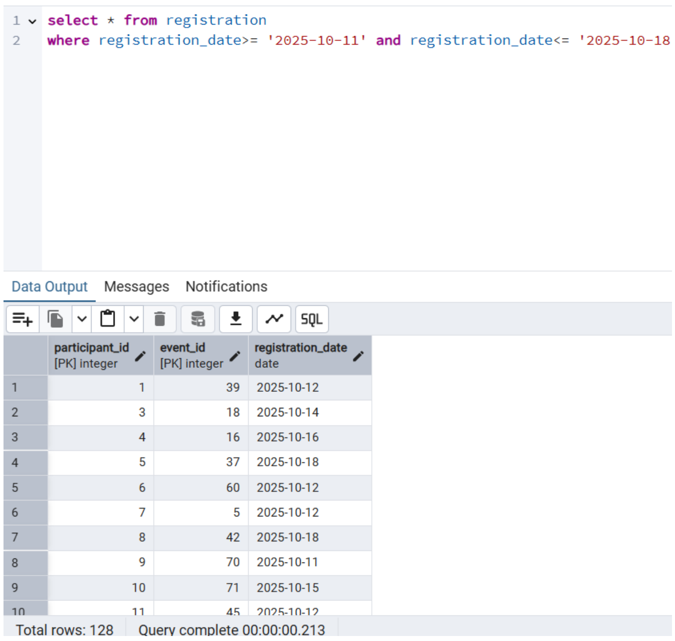
	item_id [PK] integer	item_name character varying (255)	type character varying (100)	quantity integer	condition text	dept_id integer
1	3	Shure SM58 Mic (Item #3)	Audio	18	Good	44
2	4	JBL Speaker Set (Item #4)	Audio	5	Good	38
3	13	Shure SM58 Mic (Item #13)	Audio	3	Used	60
4	14	JBL Speaker Set (Item #14)	Audio	46	Good	57
5	23	Shure SM58 Mic (Item #23)	Audio	1	Good	43
6	24	JBL Speaker Set (Item #24)	Audio	40	Used	26
7	33	Shure SM58 Mic (Item #33)	Audio	15	Good	9
8	34	JBL Speaker Set (Item #34)	Audio	11	Good	1
9	43	Shure SM58 Mic (Item #43)	Audio	27	Good	66
10	44	JBL Speaker Set (Item #44)	Audio	11	Good	8

Total rows: 16 Query complete 00:00:00.179

Figure 5.14: Simple Query 14: Inventory items of type “audio”

5.1.15 Query 15: Registrations between 11th and 16th October

Requirement: All registrations between the dates 11th and 16th October.



```
select * from registration
where registration_date>= '2025-10-11' and registration_date<= '2025-10-18
```

Data Output Messages Notifications

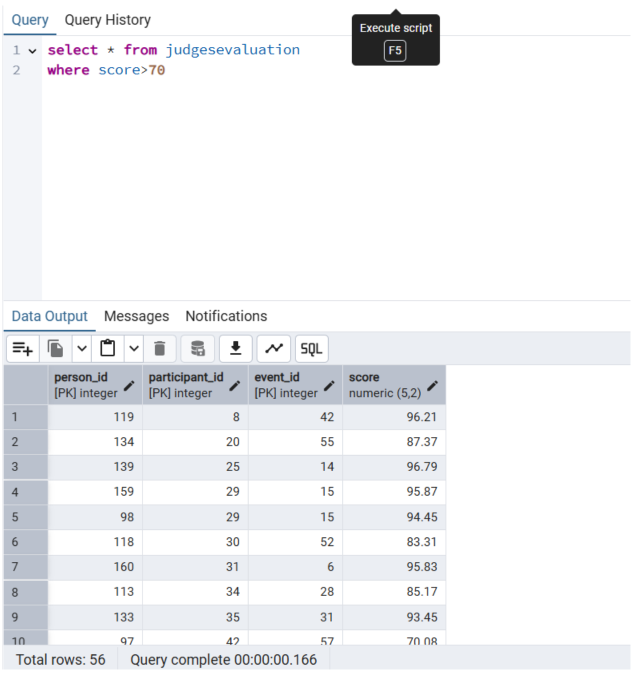
	participant_id [PK] integer	event_id [PK] integer	registration_date date
1	1	39	2025-10-12
2	3	18	2025-10-14
3	4	16	2025-10-16
4	5	37	2025-10-18
5	6	60	2025-10-12
6	7	5	2025-10-12
7	8	42	2025-10-18
8	9	70	2025-10-11
9	10	71	2025-10-15
10	11	45	2025-10-12

Total rows: 128 Query complete 00:00:00.213

Figure 5.15: Simple Query 15: Registrations between 11 and 16 October

5.1.16 Query 16: Scores above 70

Requirement: For all events and for any participants, all scores that are above 70.



```sql
select * from judgesevaluation
where score>70
```

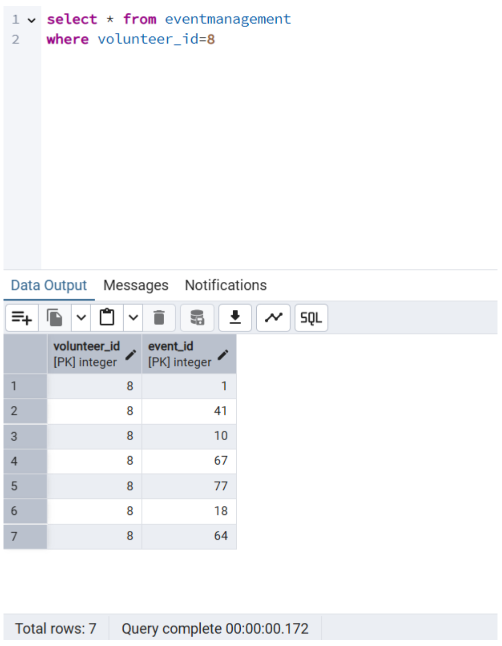
	person_id [PK] integer	participant_id [PK] integer	event_id [PK] integer	score numeric (5,2)
1	119	8	42	96.21
2	134	20	55	87.37
3	139	25	14	96.79
4	159	29	15	95.87
5	98	29	15	94.45
6	118	30	52	83.31
7	160	31	6	95.83
8	113	34	28	85.17
9	133	35	31	93.45
10	97	42	57	70.08

Total rows: 56 Query complete 00:00:00.166

Figure 5.16: Simple Query 16: Scores above 70

5.1.17 Query 17: Events managed by volunteer 8

Requirement: All events managed by volunteer with volunteer id 8.



```sql
select * from eventmanagement
where volunteer_id=8
```

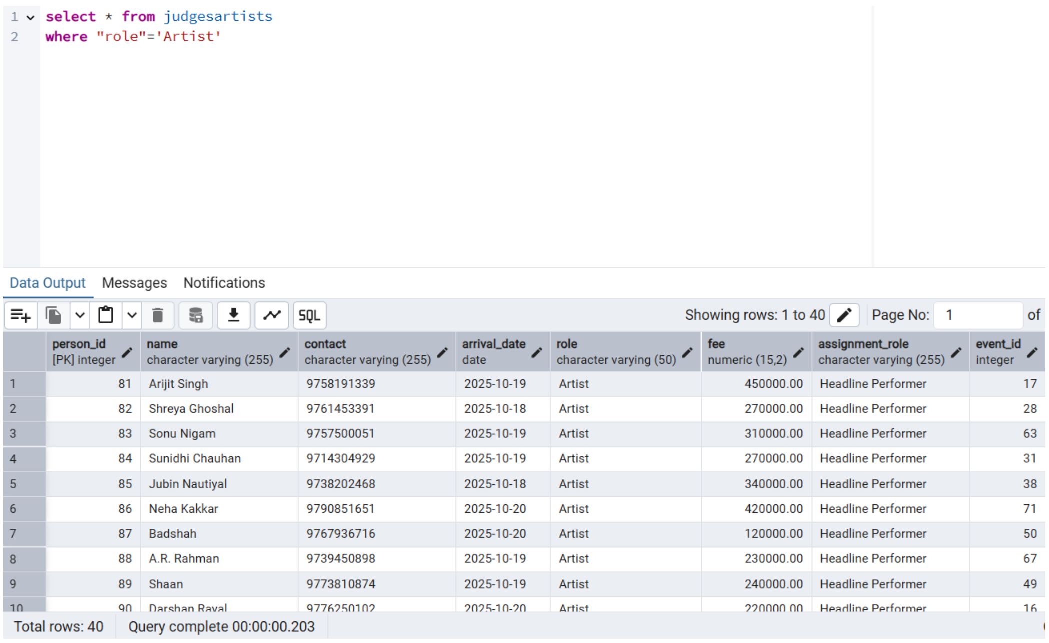
	volunteer_id [PK] integer	event_id [PK] integer
1	8	1
2	8	41
3	8	10
4	8	67
5	8	77
6	8	18
7	8	64

Total rows: 7 Query complete 00:00:00.172

Figure 5.17: Simple Query 17: Events managed by volunteer 8

5.1.18 Query 18: Guests that are artists

Requirement: All guests that are artists.



```sql
select * from judgesartists
where "role"='Artist'
```

Data Output | Messages | Notifications

Showing rows: 1 to 40 Page No: 1 of

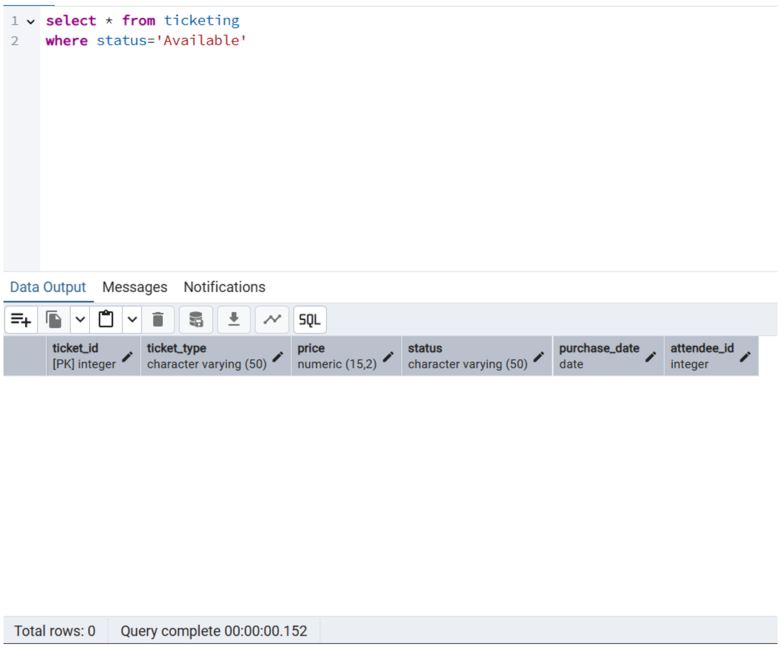
	person_id [PK] integer	name character varying (255)	contact character varying (255)	arrival_date date	role character varying (50)	fee numeric (15,2)	assignment_role character varying (255)	event_id integer
1	81	Arijit Singh	9758191339	2025-10-19	Artist	450000.00	Headline Performer	17
2	82	Shreya Ghoshal	9761453391	2025-10-18	Artist	270000.00	Headline Performer	28
3	83	Sonu Nigam	9757500051	2025-10-19	Artist	310000.00	Headline Performer	63
4	84	Sunidhi Chauhan	9714304929	2025-10-19	Artist	270000.00	Headline Performer	31
5	85	Jubin Nautiyal	9738202468	2025-10-18	Artist	340000.00	Headline Performer	38
6	86	Neha Kakkar	9790851651	2025-10-20	Artist	420000.00	Headline Performer	71
7	87	Badshah	9767936716	2025-10-20	Artist	120000.00	Headline Performer	50
8	88	A.R. Rahman	9739450898	2025-10-19	Artist	230000.00	Headline Performer	67
9	89	Shaan	9773810874	2025-10-19	Artist	240000.00	Headline Performer	49
10	90	Darshan Raval	9776250102	2025-10-20	Artist	220000.00	Headline Performer	16

Total rows: 40 Query complete 00:00:00.203

Figure 5.18: Simple Query 18: Guests that are artists

5.1.19 Query 19: Tickets available for purchase

Requirement: All tickets that are available for purchase.



```sql
select * from ticketing
where status='Available'
```

Data Output | Messages | Notifications

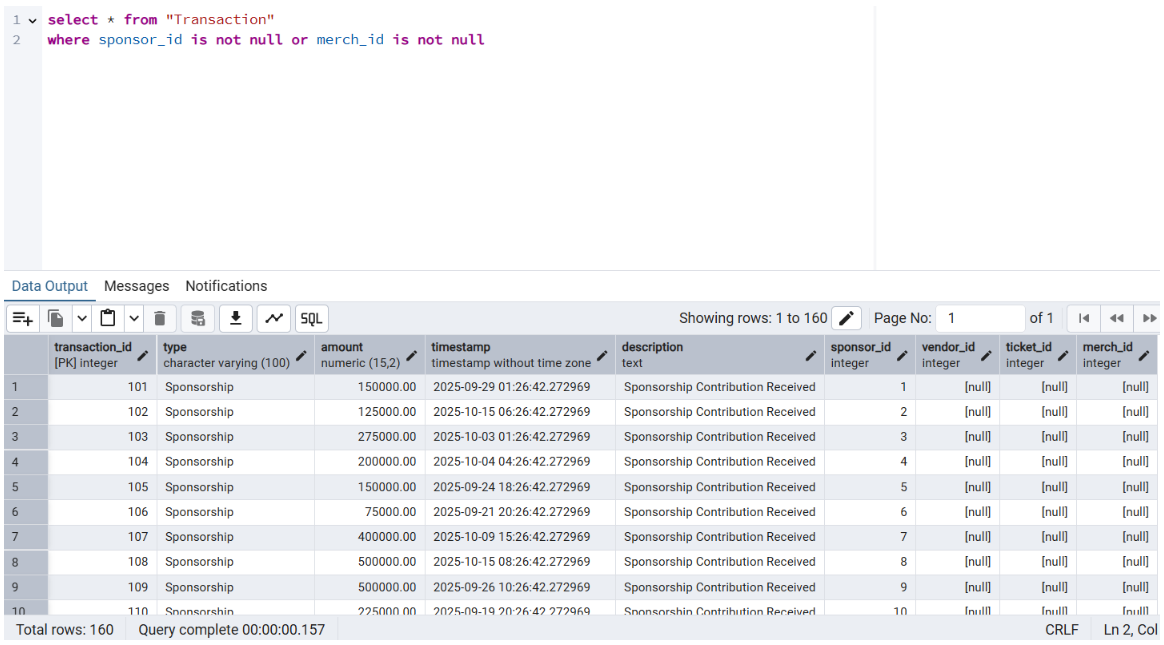
ticket_id [PK] integer	ticket_type character varying (50)	price numeric (15,2)	status character varying (50)	purchase_date date	attendee_id integer

Total rows: 0 Query complete 00:00:00.152

Figure 5.19: Simple Query 19: Tickets with status “Available”

5.1.20 Query 20: Sponsor and merchandise transactions

Requirement: All sponsor and merchandise transactions.



```sql
select * from "Transaction"
where sponsor_id is not null or merch_id is not null
```

Showing rows: 1 to 160 Page No: 1 of 1

	transaction_id [PK] integer	type character varying (100)	amount numeric (15,2)	timestamp timestamp without time zone	description text	sponsor_id integer	vendor_id integer	ticket_id integer	merch_id integer
1	101	Sponsorship	150000.00	2025-09-29 01:26:42.272969	Sponsorship Contribution Received	1	[null]	[null]	[null]
2	102	Sponsorship	125000.00	2025-10-15 06:26:42.272969	Sponsorship Contribution Received	2	[null]	[null]	[null]
3	103	Sponsorship	275000.00	2025-10-03 01:26:42.272969	Sponsorship Contribution Received	3	[null]	[null]	[null]
4	104	Sponsorship	200000.00	2025-10-04 04:26:42.272969	Sponsorship Contribution Received	4	[null]	[null]	[null]
5	105	Sponsorship	150000.00	2025-09-24 18:26:42.272969	Sponsorship Contribution Received	5	[null]	[null]	[null]
6	106	Sponsorship	75000.00	2025-09-21 20:26:42.272969	Sponsorship Contribution Received	6	[null]	[null]	[null]
7	107	Sponsorship	400000.00	2025-10-09 15:26:42.272969	Sponsorship Contribution Received	7	[null]	[null]	[null]
8	108	Sponsorship	500000.00	2025-10-15 08:26:42.272969	Sponsorship Contribution Received	8	[null]	[null]	[null]
9	109	Sponsorship	500000.00	2025-09-26 10:26:42.272969	Sponsorship Contribution Received	9	[null]	[null]	[null]
10	110	Sponsorship	225000.00	2025-09-19 20:26:42.272969	Sponsorship Contribution Received	10	[null]	[null]	[null]

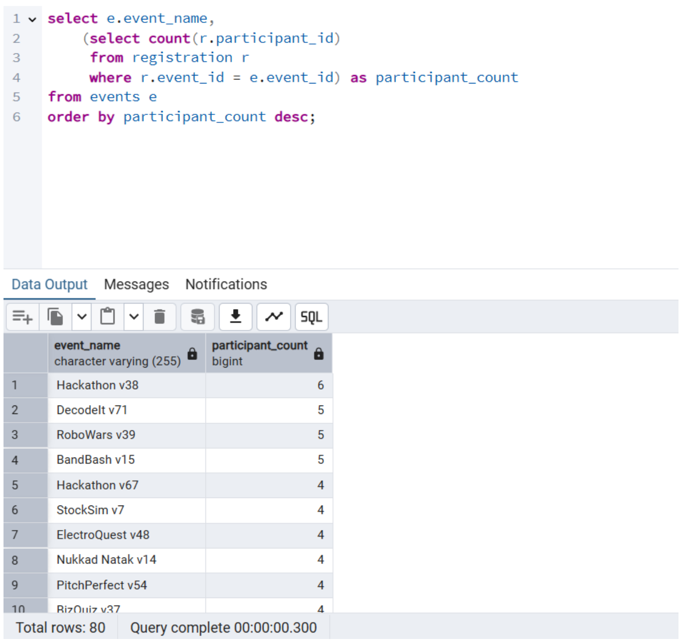
Total rows: 160 Query complete 00:00:00.157 CRLF Ln 2, Col

Figure 5.20: Simple Query 20: Sponsor and merchandise related transactions

5.2 Complex Queries

5.2.1 Complex Query 1: Participants per event

Requirement: Show each event's name and the total number of participants registered for it.



```sql
select e.event_name,
    (select count(r.participant_id)
     from registration r
     where r.event_id = e.event_id) as participant_count
from events e
order by participant_count desc;
```

Data Output   Messages   Notifications

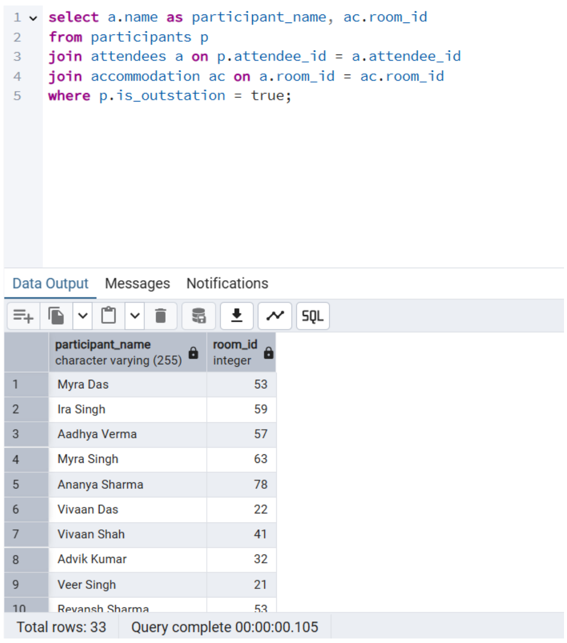
	event_name character varying (255)	participant_count bigint
1	Hackathon v38	6
2	DecodeIt v71	5
3	RoboWars v39	5
4	BandBash v15	5
5	Hackathon v67	4
6	StockSim v7	4
7	ElectroQuest v48	4
8	Nukkad Natak v14	4
9	PitchPerfect v54	4
10	BizQuiz v37	4

Total rows: 80   Query complete 00:00:00.300

Figure 5.21: Complex Query 1: Event name with total participants

5.2.2 Complex Query 2: Out-of-station participants and rooms

Requirement: List the names of all out-of-station participants and the `room_id` of their accommodation.



```sql
select a.name as participant_name, ac.room_id
from participants p
join attendees a on p.attendee_id = a.attendee_id
join accommodation ac on a.room_id = ac.room_id
where p.is_outstation = true;
```

Data Output   Messages   Notifications

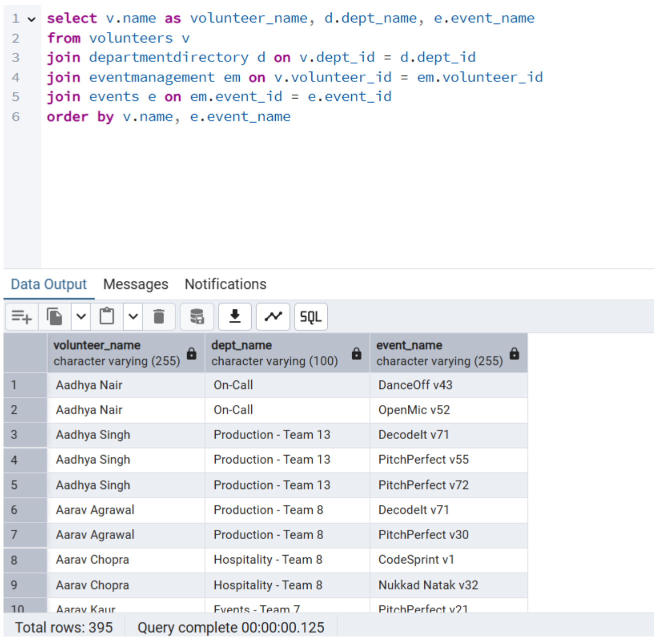
	participant_name character varying (255)	room_id integer
1	Myra Das	53
2	Ira Singh	59
3	Aadhya Verma	57
4	Myra Singh	63
5	Ananya Sharma	78
6	Vivaan Das	22
7	Vivaan Shah	41
8	Advik Kumar	32
9	Veer Singh	21
10	Revansh Sharma	53

Total rows: 33   Query complete 00:00:00.105

Figure 5.22: Complex Query 2: Out-of-station participants with room details

5.2.3 Complex Query 3: Volunteers, departments, and events

Requirement: Show a list of all volunteers, their department name, and the name of the event they are managing.



```sql
select v.name as volunteer_name, d.dept_name, e.event_name
from volunteers v
join departmentdirectory d on v.dept_id = d.dept_id
join eventmanagement em on v.volunteer_id = em.volunteer_id
join events e on em.event_id = e.event_id
order by v.name, e.event_name
```

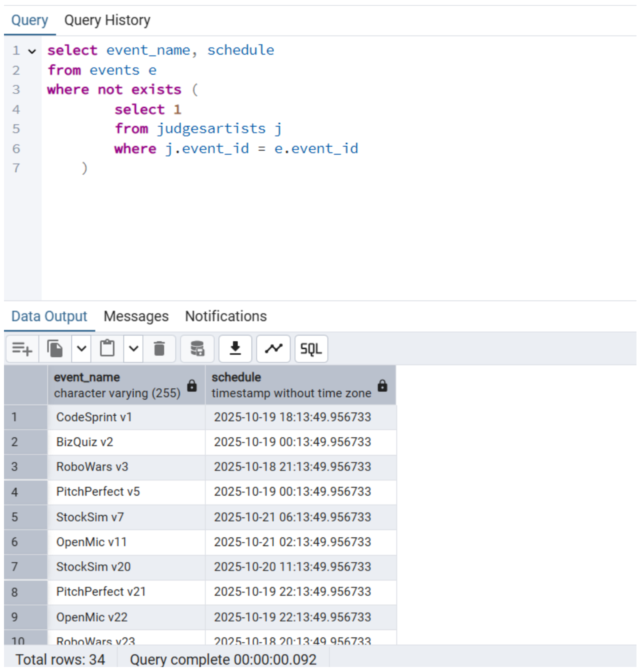
	volunteer_name character varying (255)	dept_name character varying (100)	event_name character varying (255)
1	Aadhya Nair	On-Call	DanceOff v43
2	Aadhya Nair	On-Call	OpenMic v52
3	Aadhya Singh	Production - Team 13	DecodeIt v71
4	Aadhya Singh	Production - Team 13	PitchPerfect v55
5	Aadhya Singh	Production - Team 13	PitchPerfect v72
6	Aarav Agrawal	Production - Team 8	DecodeIt v71
7	Aarav Agrawal	Production - Team 8	PitchPerfect v30
8	Aarav Chopra	Hospitality - Team 8	CodeSprint v1
9	Aarav Chopra	Hospitality - Team 8	Nukkad Natak v32
10	Aarav Kaur	Events - Team 7	PitchPerfect v21

```
Total rows: 395    Query complete 00:00:00.125
```

Figure 5.23: Complex Query 3: Volunteers with department and event

5.2.4 Complex Query 4: Events without judges or artists

Requirement: Find all events that do not have any judges or artists assigned to them yet.



```sql
select event_name, schedule
from events e
where not exists (
        select 1
        from judgesartists j
        where j.event_id = e.event_id
    )
```

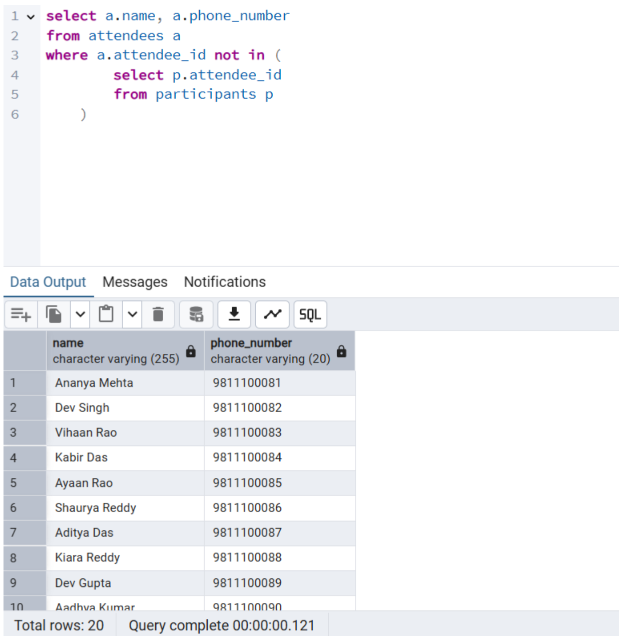
	event_name character varying (255)	schedule timestamp without time zone
1	CodeSprint v1	2025-10-19 18:13:49.956733
2	BizQuiz v2	2025-10-19 00:13:49.956733
3	RoboWars v3	2025-10-18 21:13:49.956733
4	PitchPerfect v5	2025-10-19 00:13:49.956733
5	StockSim v7	2025-10-21 06:13:49.956733
6	OpenMic v11	2025-10-21 02:13:49.956733
7	StockSim v20	2025-10-20 11:13:49.956733
8	PitchPerfect v21	2025-10-19 22:13:49.956733
9	OpenMic v22	2025-10-19 22:13:49.956733
10	RoboWars v23	2025-10-18 20:13:49.956733

```
Total rows: 34    Query complete 00:00:00.092
```

Figure 5.24: Complex Query 4: Events without assigned judges/artists

5.2.5 Complex Query 5: Ticket buyers not registered as participants

Requirement: List all attendees who have bought a ticket but have not registered as a participant in any event.



```
select a.name, a.phone_number
from attendees a
where a.attendee_id not in (
        select p.attendee_id
        from participants p
    )
```

Data Output Messages Notifications

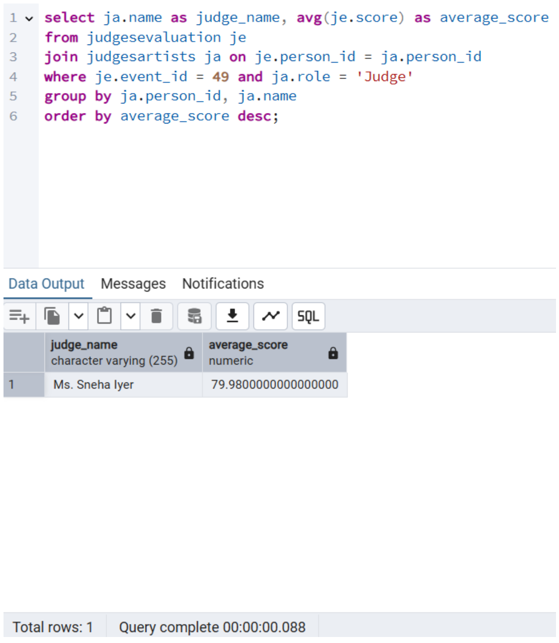
	name character varying (255)	phone_number character varying (20)
1	Ananya Mehta	9811100081
2	Dev Singh	9811100082
3	Vihaan Rao	9811100083
4	Kabir Das	9811100084
5	Ayaan Rao	9811100085
6	Shaurya Reddy	9811100086
7	Aditya Das	9811100087
8	Kiara Reddy	9811100088
9	Dev Gupta	9811100089
10	Aadhya Kumar	9811100090

Total rows: 20 Query complete 00:00:00.121

Figure 5.25: Complex Query 5: Attendees with tickets but no event registration

5.2.6 Complex Query 6: Judges and average scores for event 49

Requirement: For a specific event, say event with event id 49, show each judge's name and the average score they gave.



```
select ja.name as judge_name, avg(je.score) as average_score
from judgesevaluation je
join judgesartists ja on je.person_id = ja.person_id
where je.event_id = 49 and ja.role = 'Judge'
group by ja.person_id, ja.name
order by average_score desc;
```

Data Output Messages Notifications

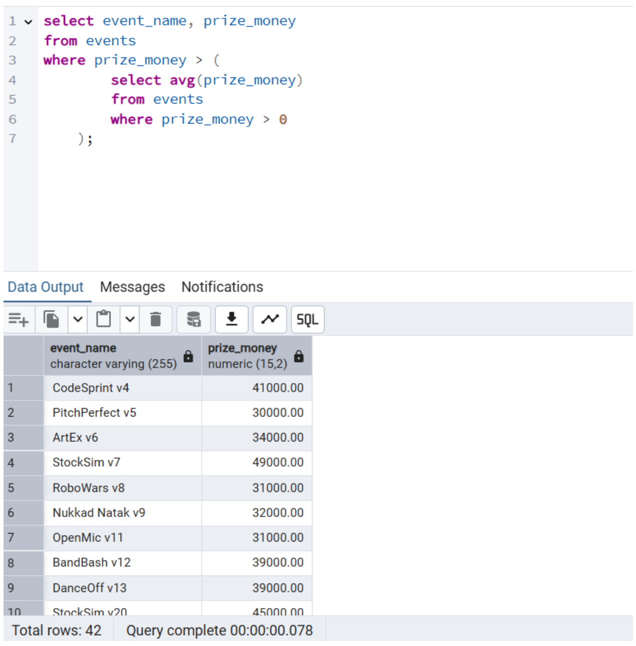
	judge_name character varying (255)	average_score numeric
1	Ms. Sneha Iyer	79.9800000000000000

Total rows: 1 Query complete 00:00:00.088

Figure 5.26: Complex Query 6: Judges and their average scores for event 49

5.2.7 Complex Query 7: Events with prize money above average

Requirement: List all events whose prize money is greater than the average prize money of all events.



```sql
select event_name, prize_money
from events
where prize_money > (
        select avg(prize_money)
        from events
        where prize_money > 0
    );
```

Data Output | Messages | Notifications

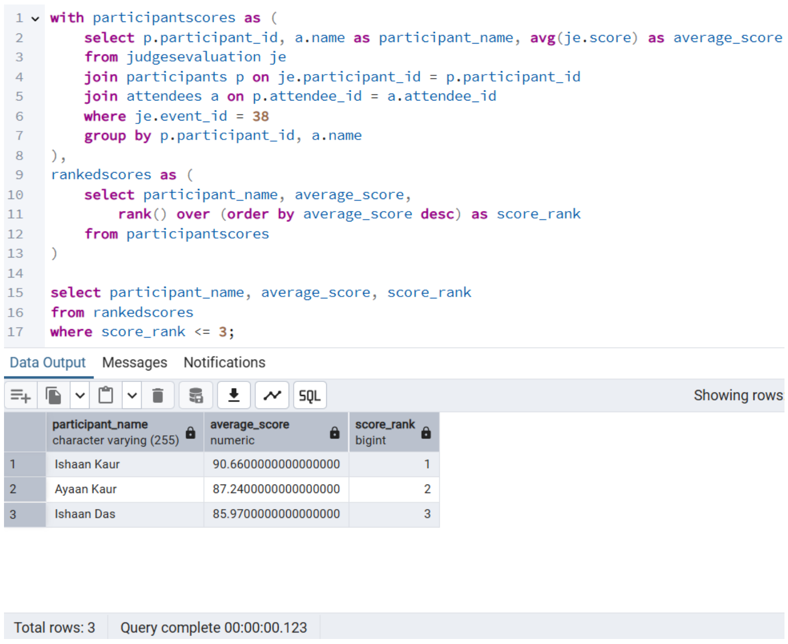
	event_name character varying (255)	prize_money numeric (15,2)
1	CodeSprint v4	41000.00
2	PitchPerfect v5	30000.00
3	ArtEx v6	34000.00
4	StockSim v7	49000.00
5	RoboWars v8	31000.00
6	Nukkad Natak v9	32000.00
7	OpenMic v11	31000.00
8	BandBash v12	39000.00
9	DanceOff v13	39000.00
10	StockSim v20	45000.00

Total rows: 42 Query complete 00:00:00.078

Figure 5.27: Complex Query 7: Events with prize money greater than average

5.2.8 Complex Query 8: Top 3 participants for event 38

Requirement: Show the names of the top 3 highest-scoring participants for a specific event, say event 38, based on their average score from all judges.



```sql
with participantscores as (
    select p.participant_id, a.name as participant_name, avg(je.score) as average_score
    from judgesevaluation je
    join participants p on je.participant_id = p.participant_id
    join attendees a on p.attendee_id = a.attendee_id
    where je.event_id = 38
    group by p.participant_id, a.name
),
rankedscores as (
    select participant_name, average_score,
        rank() over (order by average_score desc) as score_rank
    from participantscores
)

select participant_name, average_score, score_rank
from rankedscores
where score_rank <= 3;
```

Data Output | Messages | Notifications

Showing rows

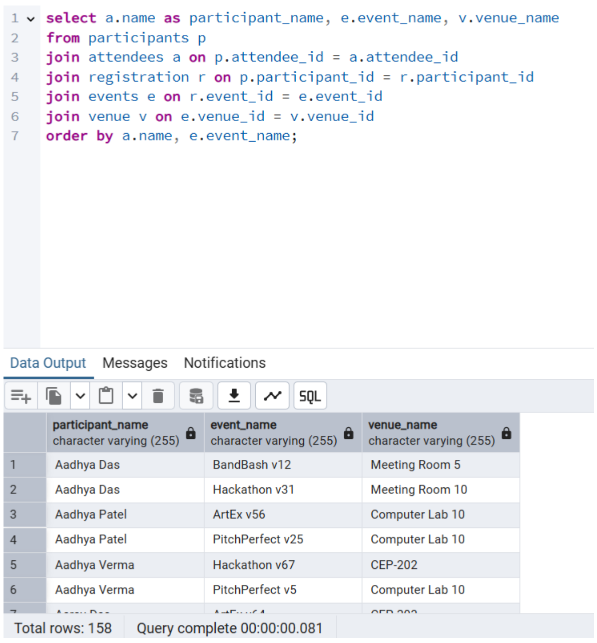
	participant_name character varying (255)	average_score numeric	score_rank bigint
1	Ishaan Kaur	90.6600000000000000	1
2	Ayaan Kaur	87.2400000000000000	2
3	Ishaan Das	85.9700000000000000	3

Total rows: 3 Query complete 00:00:00.123

Figure 5.28: Complex Query 8: Top 3 highest-scoring participants for event 38

5.2.9 Complex Query 9: Participants and venues

Requirement: List all participant names and the venues of the events they are registered for.



```sql
select a.name as participant_name, e.event_name, v.venue_name
from participants p
join attendees a on p.attendee_id = a.attendee_id
join registration r on p.participant_id = r.participant_id
join events e on r.event_id = e.event_id
join venue v on e.venue_id = v.venue_id
order by a.name, e.event_name;
```

Data Output Messages Notifications

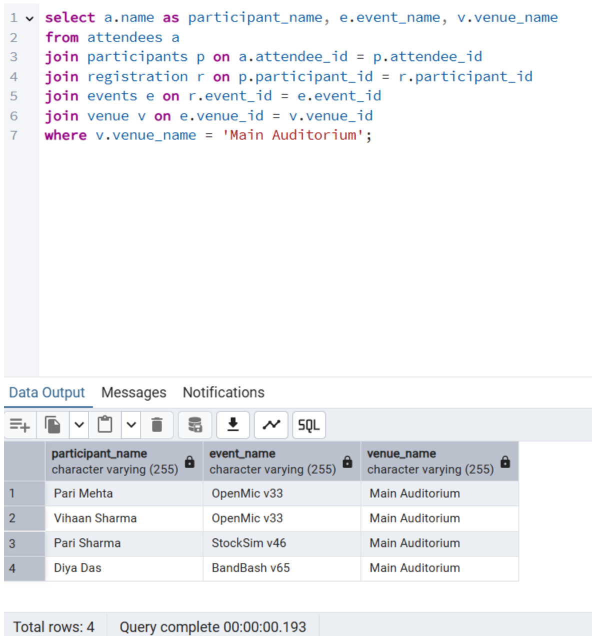
	participant_name character varying (255)	event_name character varying (255)	venue_name character varying (255)
1	Aadhya Das	BandBash v12	Meeting Room 5
2	Aadhya Das	Hackathon v31	Meeting Room 10
3	Aadhya Patel	ArtEx v56	Computer Lab 10
4	Aadhya Patel	PitchPerfect v25	Computer Lab 10
5	Aadhya Verma	Hackathon v67	CEP-202
6	Aadhya Verma	PitchPerfect v5	Computer Lab 10

Total rows: 158 Query complete 00:00:00.081

Figure 5.29: Complex Query 9: Participants with their event venues

5.2.10 Complex Query 10: Participants in Main Auditorium events

Requirement: Get the names of all participants who are registered for any event held in the 'Main Auditorium'.



```sql
select a.name as participant_name, e.event_name, v.venue_name
from attendees a
join participants p on a.attendee_id = p.attendee_id
join registration r on p.participant_id = r.participant_id
join events e on r.event_id = e.event_id
join venue v on e.venue_id = v.venue_id
where v.venue_name = 'Main Auditorium';
```

Data Output Messages Notifications

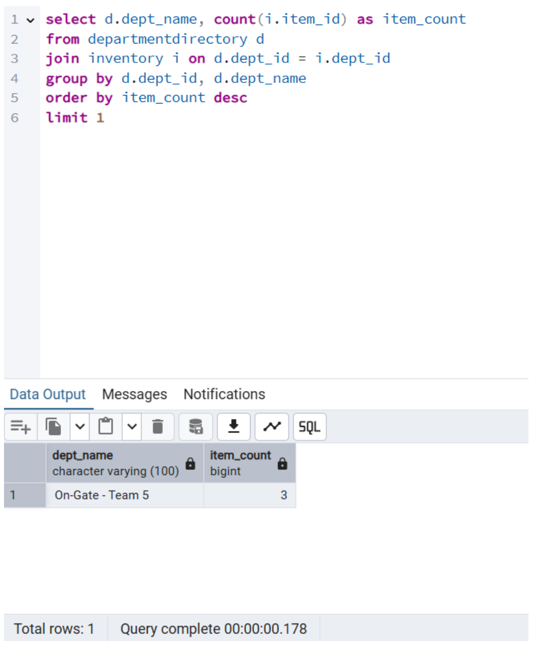
	participant_name character varying (255)	event_name character varying (255)	venue_name character varying (255)
1	Pari Mehta	OpenMic v33	Main Auditorium
2	Vihaan Sharma	OpenMic v33	Main Auditorium
3	Pari Sharma	StockSim v46	Main Auditorium
4	Diya Das	BandBash v65	Main Auditorium

Total rows: 4 Query complete 00:00:00.193

Figure 5.30: Complex Query 10: Participants registered for events in the Main Auditorium

5.2.11 Complex Query 11: Department with maximum inventory items

Requirement: Find the name of the department that has the highest number of inventory items.



```sql
select d.dept_name, count(i.item_id) as item_count
from departmentdirectory d
join inventory i on d.dept_id = i.dept_id
group by d.dept_id, d.dept_name
order by item_count desc
limit 1
```

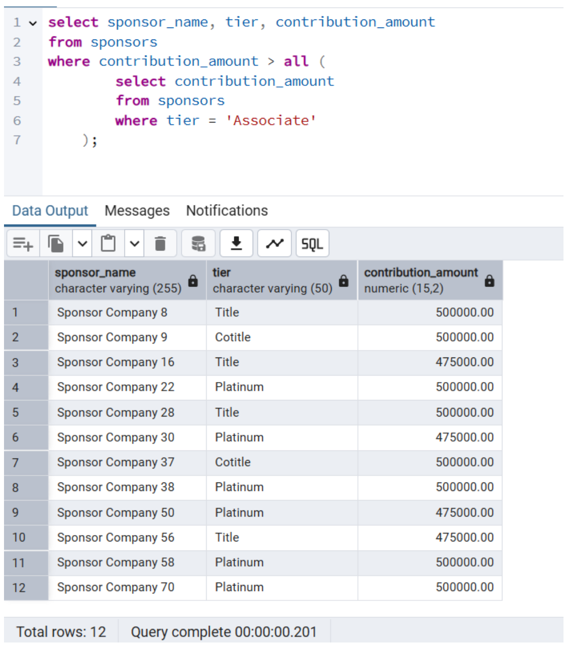
	dept_name character varying (100)	item_count bigint
1	On-Gate - Team 5	3

Total rows: 1 Query complete 00:00:00.178

Figure 5.31: Complex Query 11: Department with highest number of inventory items

5.2.12 Complex Query 12: Sponsors above all Associates

Requirement: Show all sponsors (and their tiers) who contributed an amount greater than every single 'Associate' tier sponsor.



```sql
select sponsor_name, tier, contribution_amount
from sponsors
where contribution_amount > all (
        select contribution_amount
        from sponsors
        where tier = 'Associate'
    );
```

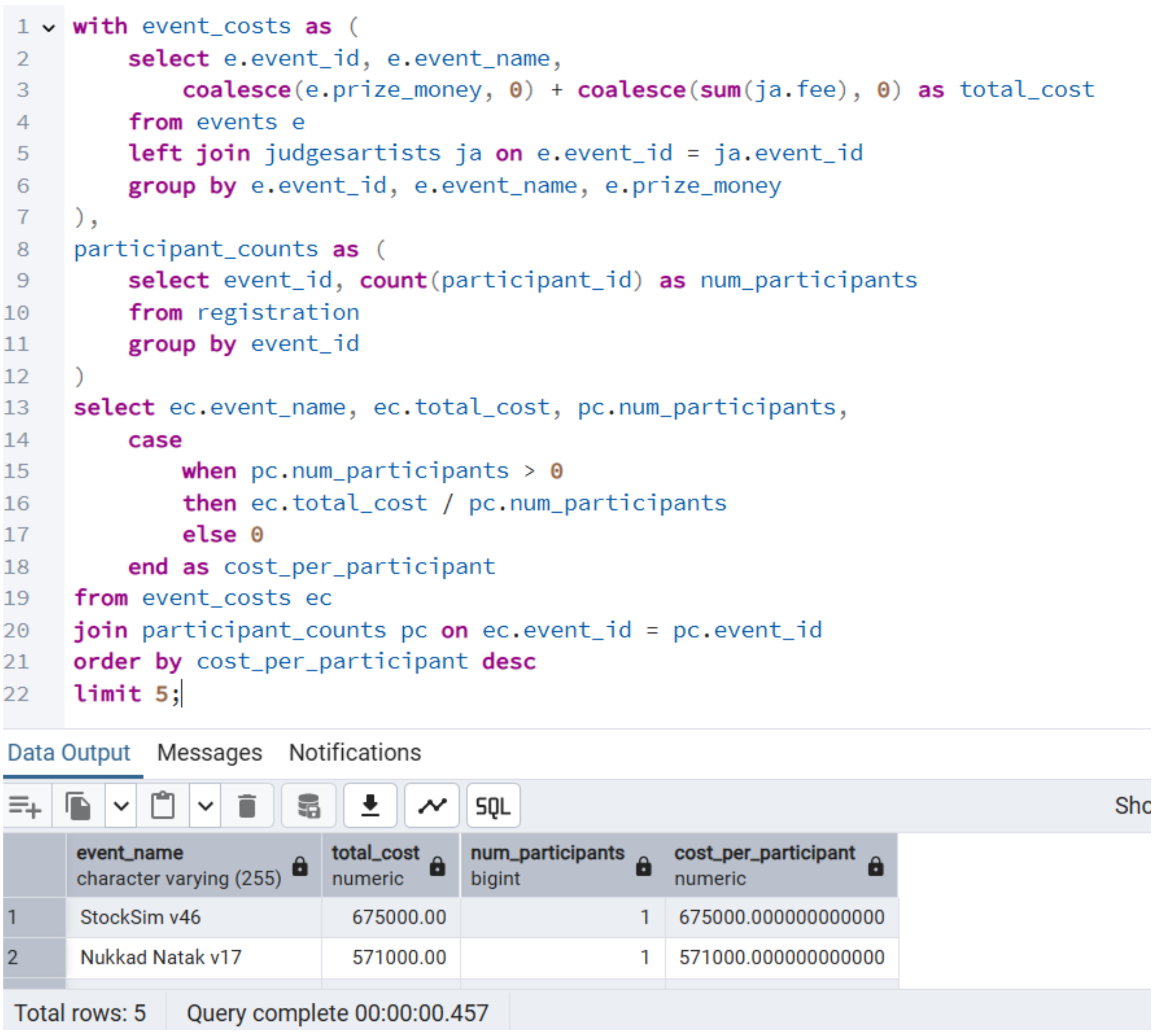
	sponsor_name character varying (255)	tier character varying (50)	contribution_amount numeric (15,2)
1	Sponsor Company 8	Title	500000.00
2	Sponsor Company 9	Cotitle	500000.00
3	Sponsor Company 16	Title	475000.00
4	Sponsor Company 22	Platinum	500000.00
5	Sponsor Company 28	Title	500000.00
6	Sponsor Company 30	Platinum	475000.00
7	Sponsor Company 37	Cotitle	500000.00
8	Sponsor Company 38	Platinum	500000.00
9	Sponsor Company 50	Platinum	475000.00
10	Sponsor Company 56	Title	475000.00
11	Sponsor Company 58	Platinum	500000.00
12	Sponsor Company 70	Platinum	500000.00

Total rows: 12 Query complete 00:00:00.201

Figure 5.32: Complex Query 12: Sponsors contributing more than all Associate sponsors

5.2.13 Complex Query 13: Top 5 most expensive events by cost per participant

Requirement: Find the top 5 most expensive events, ranked by their 'cost per participant'.



```sql
with event_costs as (
    select e.event_id, e.event_name,
        coalesce(e.prize_money, 0) + coalesce(sum(ja.fee), 0) as total_cost
    from events e
    left join judgesartists ja on e.event_id = ja.event_id
    group by e.event_id, e.event_name, e.prize_money
),
participant_counts as (
    select event_id, count(participant_id) as num_participants
    from registration
    group by event_id
)
select ec.event_name, ec.total_cost, pc.num_participants,
    case
        when pc.num_participants > 0
        then ec.total_cost / pc.num_participants
        else 0
    end as cost_per_participant
from event_costs ec
join participant_counts pc on ec.event_id = pc.event_id
order by cost_per_participant desc
limit 5;
```

	event_name character varying (255)	total_cost numeric	num_participants bigint	cost_per_participant numeric
1	StockSim v46	675000.00	1	675000.000000000000
2	Nukkad Natak v17	571000.00	1	571000.000000000000

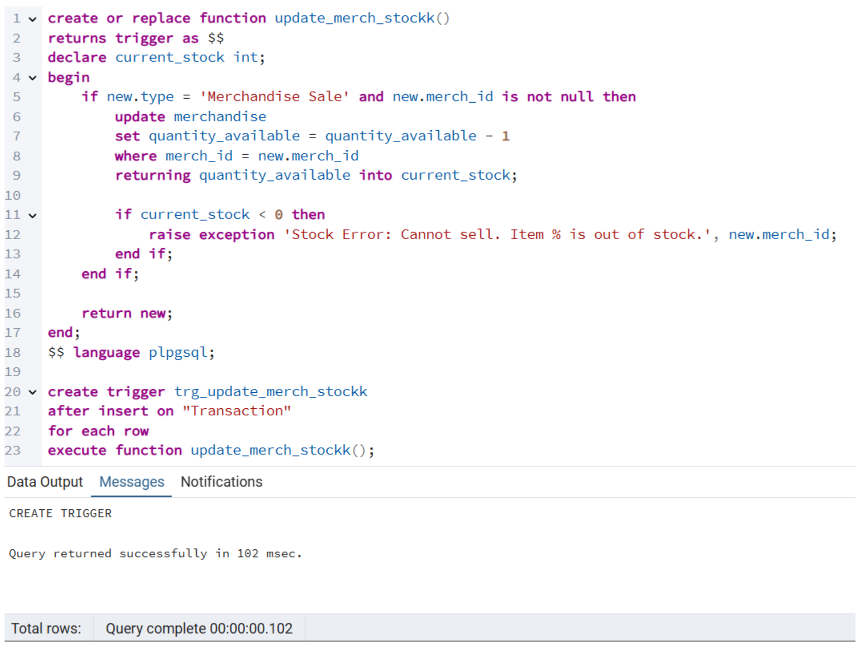
Total rows: 5   Query complete 00:00:00.457

Figure 5.33: Complex Query 13: Top 5 events by cost per participant

5.3 Triggers

5.3.1 Trigger 14: Decrease merchandise quantity on sale

Requirement: Create a trigger that automatically decreases the `quantity_available` in the `merchandise` table each time a ’Merchandise Sale’ is recorded in the `transaction` table.



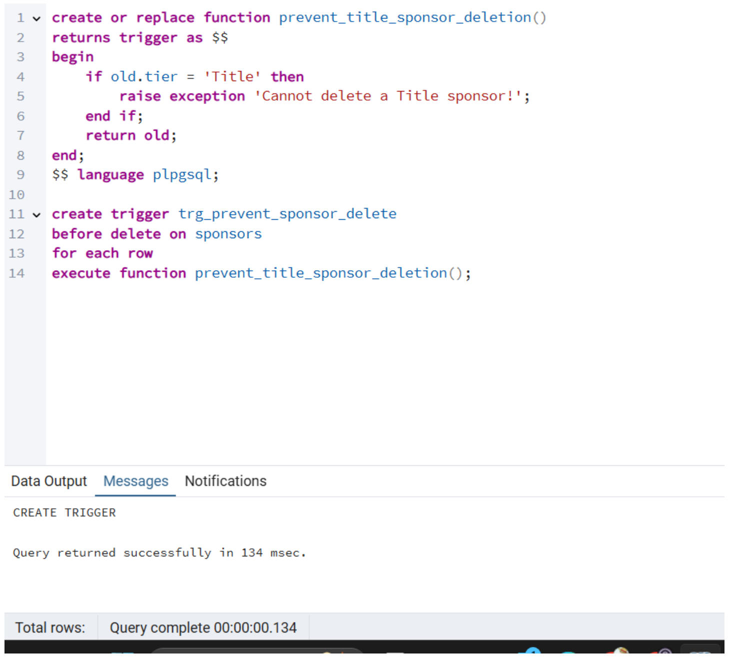
```
create or replace function update_merch_stockk()
returns trigger as $$
declare current_stock int;
begin
    if new.type = 'Merchandise Sale' and new.merch_id is not null then
        update merchandise
        set quantity_available = quantity_available - 1
        where merch_id = new.merch_id
        returning quantity_available into current_stock;

        if current_stock < 0 then
            raise exception 'Stock Error: Cannot sell. Item % is out of stock.', new.merch_id;
        end if;
    end if;

    return new;
end;
$$ language plpgsql;

create trigger trg_update_merch_stockk
after insert on "Transaction"
for each row
execute function update_merch_stockk();
```

```
CREATE TRIGGER

Query returned successfully in 102 msec.
```

Figure 5.34: Trigger 14: Decreasing merchandise quantity on merchandise sale

5.3.2 Trigger 15: Prevent deletion of Title sponsors

Requirement: Create a trigger that prevents anyone from accidentally deleting a sponsor who is in the top ’Title’ tier.



```
create or replace function prevent_title_sponsor_deletion()
returns trigger as $$
begin
    if old.tier = 'Title' then
        raise exception 'Cannot delete a Title sponsor!';
    end if;
    return old;
end;
$$ language plpgsql;

create trigger trg_prevent_sponsor_delete
before delete on sponsors
for each row
execute function prevent_title_sponsor_deletion();
```

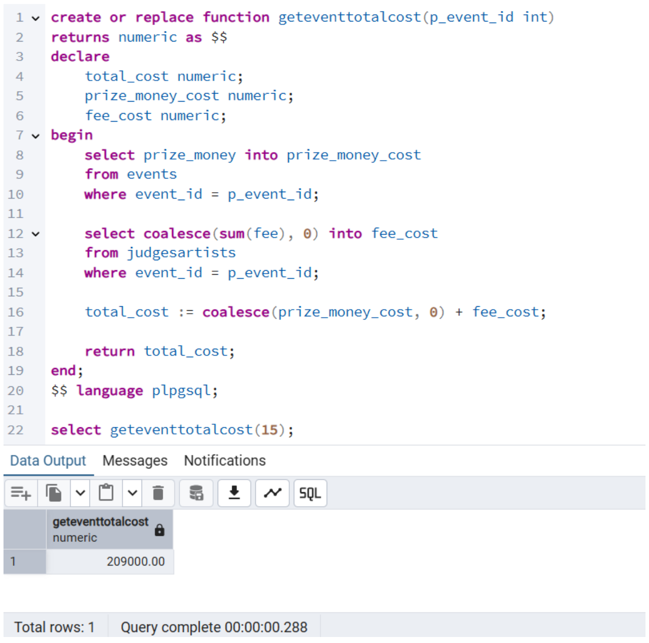
```
CREATE TRIGGER

Query returned successfully in 134 msec.
```

Figure 5.35: Trigger 15: Prevent deletion of Title-tier sponsors

5.4 Functions

5.4.1 Function 16: Total cost of an event

Requirement: Create a function that takes an `event_id` and returns its total cost (Prize Money + all Judge/Artist Fees).



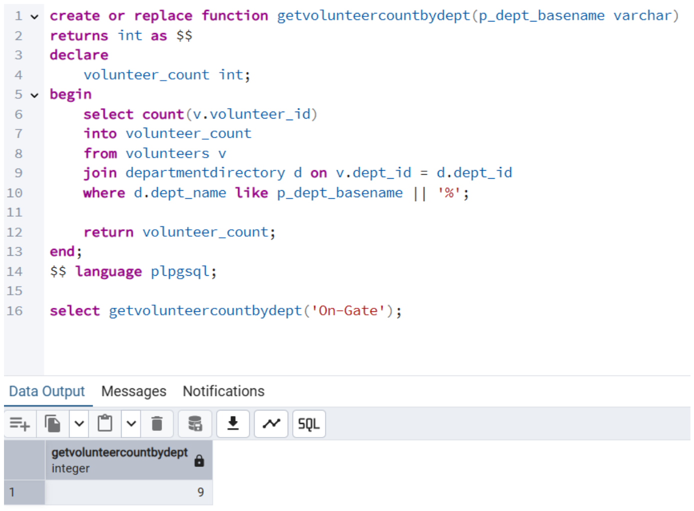
```
create or replace function geteventtotalcost(p_event_id int)
returns numeric as $$
declare
    total_cost numeric;
    prize_money_cost numeric;
    fee_cost numeric;
begin
    select prize_money into prize_money_cost
    from events
    where event_id = p_event_id;

    select coalesce(sum(fee), 0) into fee_cost
    from judgesartists
    where event_id = p_event_id;

    total_cost := coalesce(prize_money_cost, 0) + fee_cost;

    return total_cost;
end;
$$ language plpgsql;

select geteventtotalcost(15);
```

```
Data Output   Messages   Notifications

   geteventtotalcost
   numeric
1        209000.00

Total rows: 1   Query complete 00:00:00.288
```

Figure 5.36: Function 16: Total cost of an event

5.4.2 Function 17: Number of volunteers in a department

Requirement: Create a function that takes a `dept_name` and returns the total number of volunteers in that department.



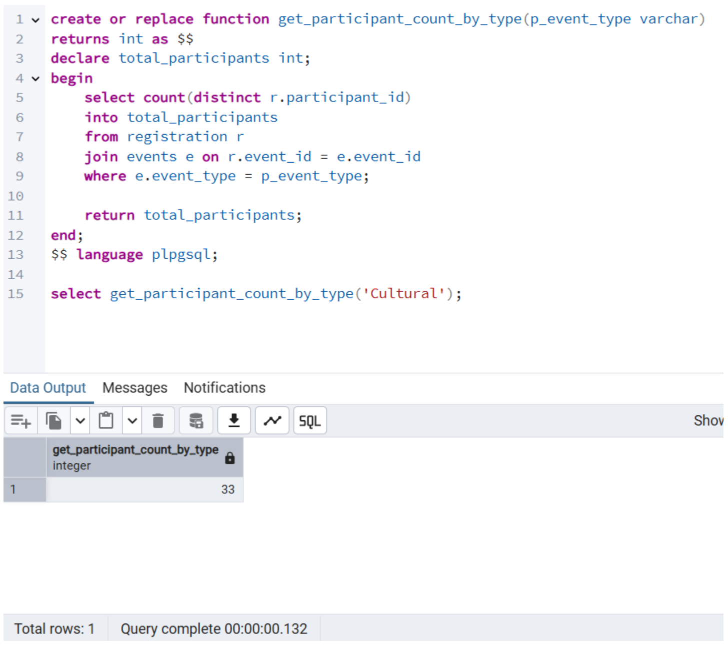
```
create or replace function getvolunteercountbydept(p_dept_basename varchar)
returns int as $$
declare
    volunteer_count int;
begin
    select count(v.volunteer_id)
    into volunteer_count
    from volunteers v
    join departmentdirectory d on v.dept_id = d.dept_id
    where d.dept_name like p_dept_basename || '%';

    return volunteer_count;
end;
$$ language plpgsql;

select getvolunteercountbydept('On-Gate');
```

```
Data Output   Messages   Notifications

   getvolunteercountbydept
   integer
1                        9
```

Figure 5.37: Function 17: Total volunteers in a given department

5.4.3 Function 18: Unique participants for an event type

Requirement: Create a function that takes an `event_type` and returns the total number of unique participants registered for all events of that type.



```
create or replace function get_participant_count_by_type(p_event_type varchar)
returns int as $$
declare total_participants int;
begin
    select count(distinct r.participant_id)
    into total_participants
    from registration r
    join events e on r.event_id = e.event_id
    where e.event_type = p_event_type;

    return total_participants;
end;
$$ language plpgsql;

select get_participant_count_by_type('Cultural');
```

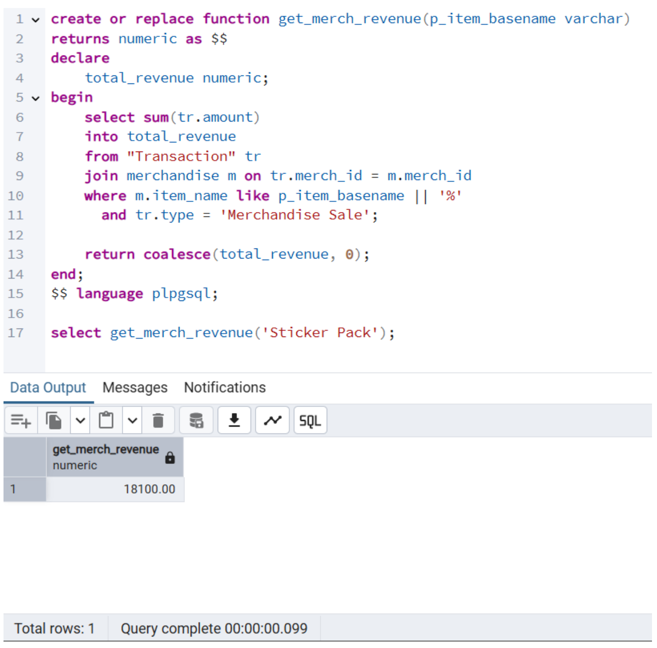
Data Output Messages Notifications

```
  get_participant_count_by_type
  integer
1                            33

Total rows: 1    Query complete 00:00:00.132
```

Figure 5.38: Function 18: Unique participants for a given event type

5.4.4 Function 19: Revenue from a specific merchandise item

Requirement: Create a function that takes an `item_name` and returns the total revenue from sales of that specific merchandise item.



```
create or replace function get_merch_revenue(p_item_basename varchar)
returns numeric as $$
declare
    total_revenue numeric;
begin
    select sum(tr.amount)
    into total_revenue
    from "Transaction" tr
    join merchandise m on tr.merch_id = m.merch_id
    where m.item_name like p_item_basename || '%'
      and tr.type = 'Merchandise Sale';

    return coalesce(total_revenue, 0);
end;
$$ language plpgsql;

select get_merch_revenue('Sticker Pack');
```

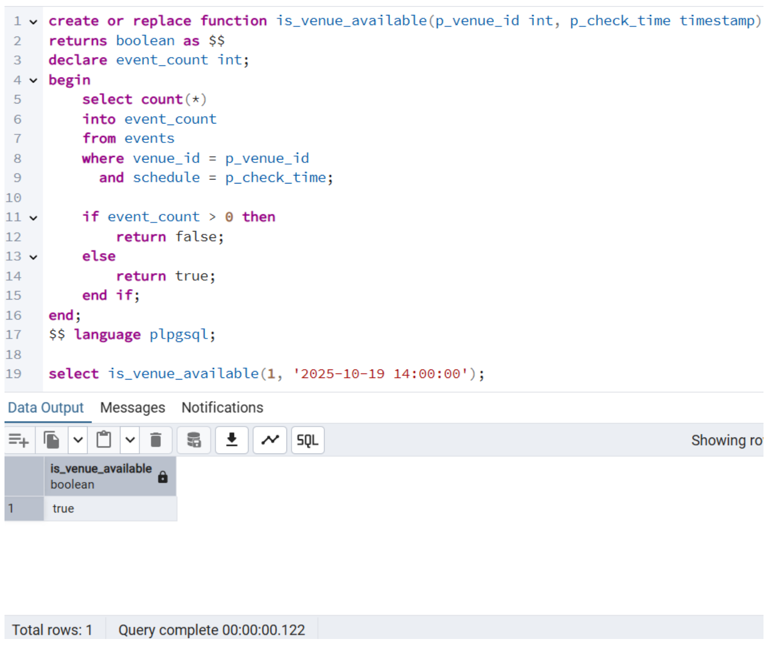
Data Output Messages Notifications

```
  get_merch_revenue
  numeric
1          18100.00

Total rows: 1    Query complete 00:00:00.099
```

Figure 5.39: Function 19: Total revenue for a given merchandise item

5.4.5 Function 20: Venue availability at a timestamp

Requirement: Create a function that checks if a specific venue is free at a given timestamp. It returns `true` if free, and `false` if it's already booked for an event.



```sql
create or replace function is_venue_available(p_venue_id int, p_check_time timestamp)
returns boolean as $$
declare event_count int;
begin
    select count(*)
    into event_count
    from events
    where venue_id = p_venue_id
      and schedule = p_check_time;

    if event_count > 0 then
        return false;
    else
        return true;
    end if;
end;
$$ language plpgsql;

select is_venue_available(1, '2025-10-19 14:00:00');
```

Data Output Messages Notifications

```
is_venue_available
boolean
1  true

Total rows: 1   Query complete 00:00:00.122
```

Figure 5.40: Function 20: Check if a venue is free at a given time

Chapter 6

Conclusion

Synapse Fest requires a centralized and secure database to replace scattered spreadsheets and informal coordination. This project delivers a normalized relational schema with integrity constraints and modular support for departments, ticket verification, accommodation, inventory, and financial transactions. The design improves communication, ensures data consistency, mitigates fraud (especially fake tickets and fund misuse), and provides a clean foundation for queries and future fest planning. The system prioritizes scalability, security, and transparency, making fest operations more reliable and efficient.