54.All-Inclusive Resource and Logistics Management System

Abstract:

The All-Inclusive Resource and Logistics Management System for Event Planning is a Java-based application designed to streamline the coordination of event resources, such as rooms, equipment, and staff. The system provides real-time tracking and status updates of resources, automates allocation processes, and resolves conflicts efficiently. Key functionalities of the system include resource management, feedback collection and analysis, and a real-time alert system for critical feedback.

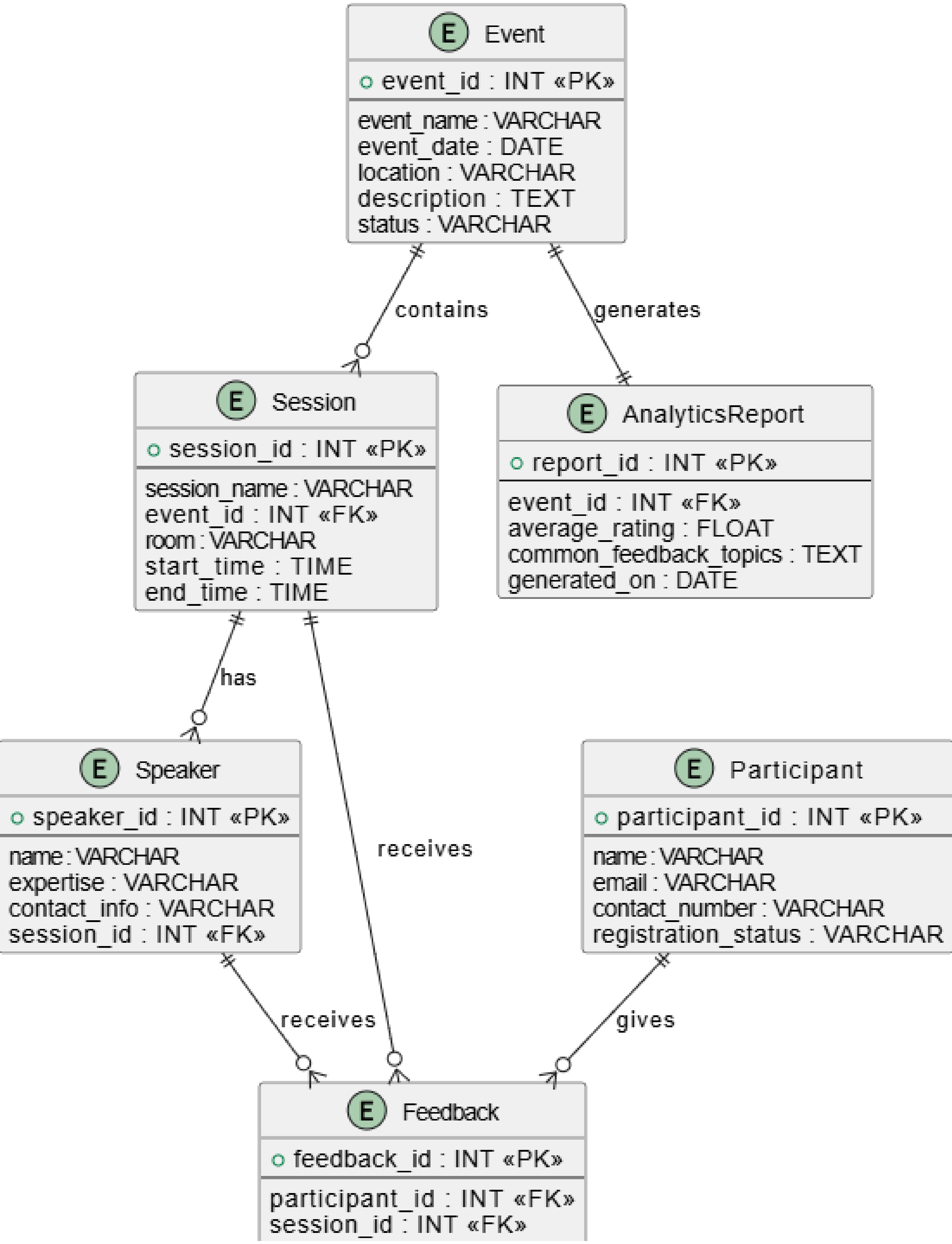
The core of the application is built using Object-Oriented Programming (OOP) principles, with essential classes like Event, Feedback, Participant, Session, Speaker, Analytics, and Alert. Interfaces are used for methods like submitting feedback, aggregating feedback, and generating reports, allowing for flexibility and scalability. Subclasses such as SessionFeedback and SpeakerFeedback extend the Feedback class, providing specialized functionality.

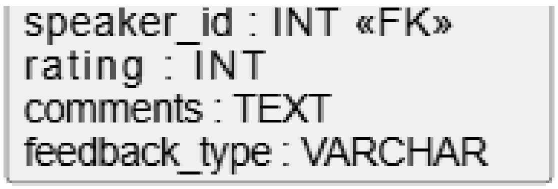
Collections such as ArrayList and HashMap are used for in-memory data processing, and file handling is implemented to export feedback analytics. JDBC is utilized for managing feedback data, and multi-threading is employed to handle concurrent feedback submissions while maintaining system performance. Thread safety is ensured through synchronized methods and blocks, ensuring that feedback processing and real-time report generation are seamless.

On the database management side, the system features normalized tables for Event, Feedback, Participant, Session, Speaker, and AnalyticsReport, with properly defined primary and foreign keys to maintain data integrity. The system leverages SQL JOIN queries to combine feedback data with event details, and views are created to provide real-time metrics such as average ratings and feedback trends. Triggers are implemented to log changes to feedback submissions, ensuring an audit trail for data modifications. The system guarantees ACID compliance and uses indexing to enhance query performance, particularly for frequently queried fields like event IDs and user IDs.

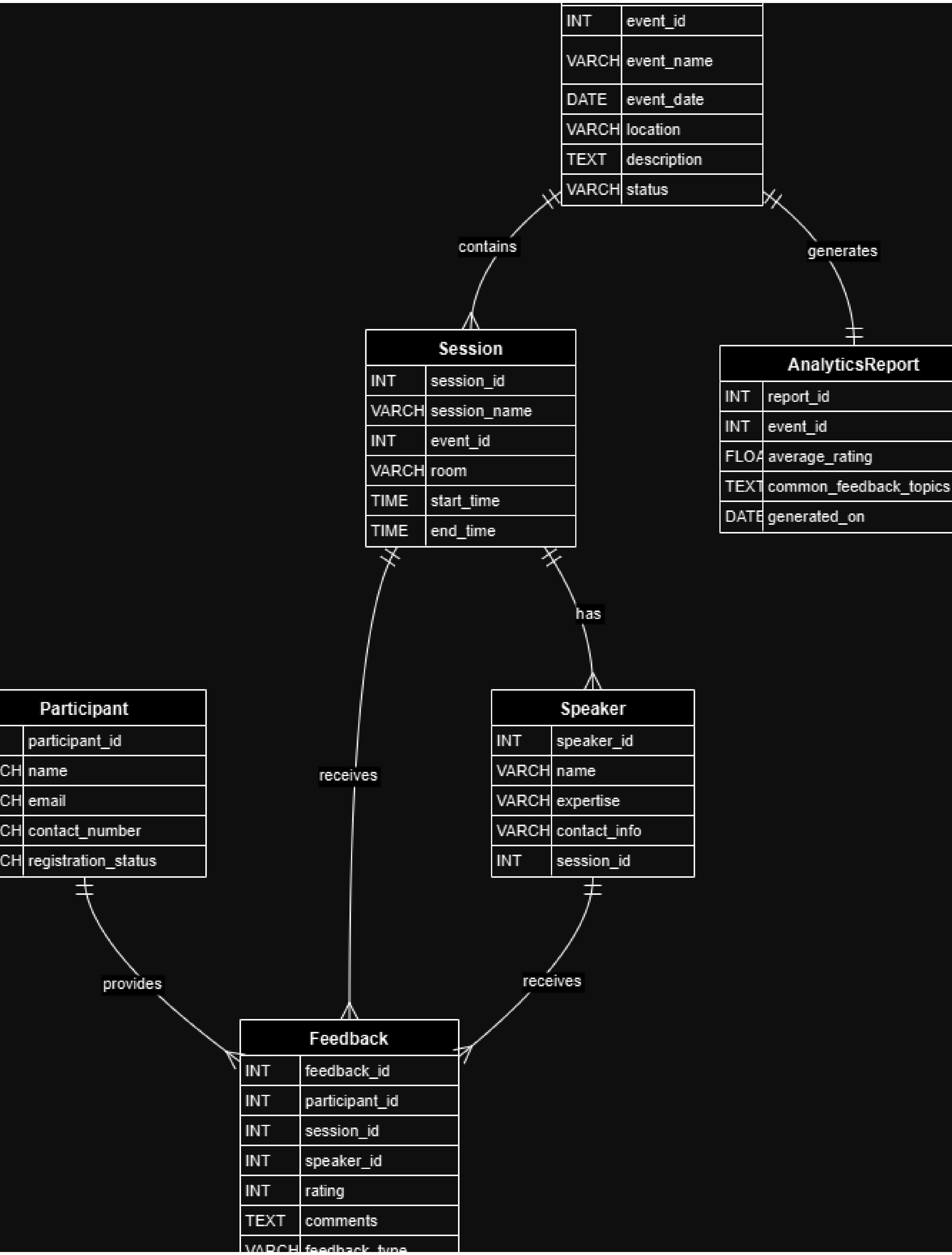
This comprehensive system is an ideal solution for managing and analyzing event-related resources and feedback, providing event planners with powerful tools for decision-making, performance monitoring, and participant satisfaction tracking.

Schema Diagram:


E
Event
event_id : INT «PK»
event_name : VARCHAR
event_date : DATE
location : VARCHAR
description : TEXT
status : VARCHAR
contains
generates
E
Session
session_id : INT «PK»
session_name : VARCHAR
event_id : INT «FK»
room : VARCHAR
start_time : TIME
end_time : TIME
E
AnalyticsReport
report_id : INT «PK»
event_id : INT «FK»
average_rating : FLOAT
common_feedback_topics : TEXT
generated_on : DATE
has
receives
E
Speaker
speaker_id : INT «PK»
name : VARCHAR
expertise : VARCHAR
contact_info : VARCHAR
session_id : INT «FK»
E
Participant
participant_id : INT «PK»
name : VARCHAR
email : VARCHAR
contact_number : VARCHAR
registration_status : VARCHAR
receives
gives
E
Feedback
feedback_id : INT «PK»
participant_id : INT «FK»
session_id : INT «FK»


speaker_id : INT «FK»
rating : INT
comments : TEXT
feedback_type : VARCHAR

ER Diagram:


INT event_id
VARCH event_name
DATE event_date
VARCH location
TEXT description
VARCH status
contains
generates
Session
INT session_id
VARCH session_name
INT event_id
VARCH room
TIME start_time
TIME end_time
AnalyticsReport
INT report_id
INT event_id
FLOA average_rating
TEXT common_feedback_topics
DATE generated_on
has
receives
Participant
participant_id
CH name
CH email
CH contact_number
CH registration_status
Speaker
INT speaker_id
VARCH name
VARCH expertise
VARCH contact_info
INT session_id
provides
receives
Feedback
INT feedback_id
INT participant_id
INT session_id
INT speaker_id
INT rating
TEXT comments
VARCH feedback_type

	VARCH	feedback_type	

MYSql Code:

```
create database logistics_management;

use logistics_management;

-- Create the 'Event' table

CREATE TABLE Event (

   event_id INT PRIMARY KEY AUTO_INCREMENT,

   event_name VARCHAR(255) NOT NULL,

   event_date DATE NOT NULL,

   event_location VARCHAR(255) NOT NULL

);

-- Create the 'Participant' table

CREATE TABLE Participant (

   participant_id INT PRIMARY KEY AUTO_INCREMENT,

   first_name VARCHAR(100) NOT NULL,

   last_name VARCHAR(100) NOT NULL,

   email VARCHAR(100) UNIQUE NOT NULL

);

-- Create the 'Session' table

CREATE TABLE Session (

   session_id INT PRIMARY KEY AUTO_INCREMENT,

   session_name VARCHAR(255) NOT NULL,

   session_time TIME NOT NULL,

   event_id INT,

   FOREIGN KEY (event_id) REFERENCES Event(event_id) ON DELETE CASCADE
);

-- Create the 'Speaker' table

CREATE TABLE Speaker (

   speaker_id INT PRIMARY KEY AUTO_INCREMENT,

   first_name VARCHAR(100) NOT NULL,

   last_name VARCHAR(100) NOT NULL,

   bio TEXT,

   session_id INT,

   FOREIGN KEY (session_id) REFERENCES Session(session_id) ON DELETE CASCADE

);

-- Create the 'Feedback' table

CREATE TABLE Feedback (

   feedback_id INT PRIMARY KEY AUTO_INCREMENT,

   rating INT CHECK (rating BETWEEN 1 AND 5),

   comments TEXT,

   participant_id INT,

   event_id INT,

   session_id INT,

   speaker_id INT,

   submission_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (participant_id) REFERENCES Participant(participant_id),

   FOREIGN KEY (event_id) REFERENCES Event(event_id),

   FOREIGN KEY (session_id) REFERENCES Session(session_id),

   FOREIGN KEY (speaker_id) REFERENCES Speaker(speaker_id)

);
```

```sql
-- Create the 'AnalyticsReport' table to store analytics data

CREATE TABLE AnalyticsReport (

   report_id INT PRIMARY KEY AUTO_INCREMENT,

   event_id INT,

   average_rating FLOAT,

   total_feedback INT,

   report_generated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (event_id) REFERENCES Event(event_id)

);

-- Create views for quick access to real-time metrics

-- View to show average ratings per event

CREATE VIEW avg_event_rating AS

SELECT e.event_name, AVG(f.rating) AS avg_rating

FROM Event e

JOIN Feedback f ON e.event_id = f.event_id

GROUP BY e.event_id;

-- View to show the count of feedback per session

CREATE VIEW session_feedback_count AS

SELECT s.session_name, COUNT(f.feedback_id) AS feedback_count

FROM Session s

LEFT JOIN Feedback f ON s.session_id = f.session_id

GROUP BY s.session_id;

-- View to show the most common feedback topics (basic example)

CREATE VIEW common_feedback_topics AS

SELECT f.comments, COUNT(f.feedback_id) AS topic_count
```

```
FROM Feedback f

GROUP BY f.comments

ORDER BY topic_count DESC;

DESCRIBE Feedback;

INSERT INTO Participant (first_name, last_name, email) VALUES ('John', 'Doe',
'john.doe@example.com');

select * from participant;

-- Insert sample events

INSERT INTO Event (event_name, event_date, event_location)

VALUES

   ('Tech Conference 2024', '2024-12-01', 'New York City'),

   ('AI Expo 2024', '2024-12-05', 'San Francisco'),

   ('Cloud Computing Summit 2024', '2024-12-10', 'Chicago');

 -- Insert sample participants

INSERT INTO Participant (first_name, last_name, email)

VALUES

   ('Alice', 'Smith', 'alice.smith@example.com'),

   ('Bob', 'Brown', 'bob.brown@example.com'),

   ('Charlie', 'Johnson', 'charlie.johnson@example.com');

-- Insert sample sessions for the 'Tech Conference 2024'

INSERT INTO Session (session_name, session_time, event_id)

VALUES

   ('AI for Beginners', '09:00:00', 1),

   ('Advanced AI Topics', '11:00:00', 1),

   ('Cloud Security Basics', '14:00:00', 2);

-- Insert sample speakers
```

```
INSERT INTO Speaker (first_name, last_name, bio, session_id)
VALUES
    ('David', 'Clark', 'Expert in AI and Machine Learning.', 1),
    ('Eva', 'Green', 'Cloud Computing Specialist.', 3);
  -- Insert sample feedback
INSERT INTO Feedback (rating, comments, participant_id, event_id, session_id, speaker_id)
VALUES
    (5, 'Great session on AI!', 1, 1, 1, 1),
    (4, 'Very informative talk on Cloud Computing.', 2, 2, 3, 2);
-- Insert sample analytics report
INSERT INTO AnalyticsReport (event_id, average_rating, total_feedback)
VALUES
    (1, 4.5, 2),
    (2, 4.0, 1);
select * from event;
select * from Participant;
select * from Feedback;
```