Hult International Business School

Carolina Galindo Mendoza

Business Intelligence & Insights

SQL and ERD Design.

12/01/2024.

GALAXY ENTERTAINMENT:

BUSINESS INTELLIGENCE & DATABASE DESIGN

Project Overview

A comprehensive database design and SQL analytics solution for a global entertainment company, featuring entity-relationship modeling, normalized database schema, and advanced business intelligence queries.

Key Features

- Complete ERD design with 15+ entities
- Normalized database schema (3NF)
- 20+ analytical SQL queries for business intelligence
- Sample dataset with 10+ records per table
- Performance optimization with indexes

Business Context

Galaxy Entertainment, inspired by real-world K-pop industry leaders, faces complex data management challenges in artist development, global market expansion, and fan engagement. This project delivers a scalable database solution and analytical framework to drive data-informed business decisions.

Technical Challenge

Design a normalized database schema that can handle:

- Multi-dimensional artist performance tracking
- Global market content distribution
- Complex fan engagement metrics
- Cross-channel revenue analysis

- Training program ROI measurement

NOTE: this case was inspired by the SM Entertainment business case from Stanford Business School, case reference no. IB108.

ENTITY IDENTIFICATION:

1. Artists:
    - Individuals that are managed by Galaxy Entertainment.
2. Trainings:
    - Training sessions or programs that artists take.
3. Contents:
    - Represents music, videos, etc.
4. Markets:
    - Regions where the company operates or aims to get in.
5. Events:
    - Live concerts, visual fan events, or promotional activities.
6. Fans:
    - Fan engagement with the company's artists and content
7. Merch:
    - Physical products for sale, such as clothes, posters, and physical albums.
8. Sales:
    - Sales made by fans for content and events.

ATTRIBUTE IDENTIFICATION:

1. Artists:
    - (PK) artist_id - VC(10)
    - artist_f_name - VC(20)
    - artist_l_name - VC(20)
    - artist_dob - Date
    - artist_nationality - VC(20)
    - artist_debut - Date
    - artist_status ENUM('active', 'inactive', 'retired')
2. Trainings:
    - (PK) training_id - VC(10)
    - (FK) artist_id - VC(10)
    - training_name - VC(20)
    - traning_professor - VC(20)

    - training_duration - int
    - training_area - VC(20)
    - training_cost - Decimal(8,2)
    - success_rate - Decimal(5,2)
3. Contents:
    - (PK) content_id - VC(10)
    - content_title - VC(20)
    - content_type - VC(20)
    - content_release - Date
    - production_cost - Decimal(10,2)
    - stream_count - int
    - revenue_generated - DECIMAL(12,2)
    - (FK) artist_id - VC(10)
4. Markets:
    - (PK) market_id - VC(10)
    - market_region - VC(20)
    - market_language - VC(20)
    - market_population - int
    - market_country - VC(20)
5. Events:
    - (PK) event_id - VC(10)
    - event_name - VC(20)
    - event_type - VC(20)
    - event_date - Date
    - event_location - VC(50)
    - event_status ENUM('scheduled', 'cancelled', 'completed')
6. Fans:
    - (PK) fan_id - VC(10)
    - (FK) artist_id - VC(10)
    - fan_email - VC(50)
    - fan_country - VC(20)
    - fan_region - VC(20)
    - fan_age - int
    - fan_engagement_score - int
    - fan_join_date - Date
7. Merch:
    - (PK) merch_id - VC(10)
    - (FK) artist_id - VC(10)
    - merch_type - VC(20)
    - merch_release - Date

    - merch_status ENUM('available', 'discontinued')
    - merch_available_units - int
8. Sales:
    - (PK) sale_id - VC(10)
    - (FK) artist_id - VC(10)
    - (FK) fan_id - VC(10)
    - sale_type - VC(20)
    - sale_date - Date
    - sale_quantity - int
    - sale_unit_price - DECIMAL(8,2)
    - sale_total_amount - DECIMAL(10,2)
    - sale_status ENUM('pending', 'completed', 'refunded')

RELATIONSHIP IDENTIFICATION:

Artists and Trainings (many-to-many):

- An artist can enroll to one or many training programs.
- A training classes can have one or many artists enrolled.

Artists and Fans (one-to-many):

- An artist can have many fans.
- A fan can have one and only one artist.

Artists and Events (many-to-many):

- An artist can participate in none or many events.
- A event need to host at least one artist and can have many artists

Artists and Contents (one-to-many):

- An artist can create many content.
- A content is created by one and only one artist.

Artist and Merch (one-to-many):

- An artist launch many merch
- A merch can be launched by one and only one artist.

Artists and Sales (one-to-many):

- An artist can make many sales
- A sale can be made by one and only one artist

Contents and Markets (many-to-many):

- A content can be release in one or many markets.
- A market can have one or many contents.

Merch and Sales (many-to-many):

- A merch can have many sales
- A sale can have many merch

Event and Sales (many-to-many):

- An event has many sales
- A sale can have many events

Fans and Events (many-to-many):

- A fan can attend many events
- An event can be attended by many fans

Event and Content (many-to-many):

- An event can feature many contents
- A content can be featured by many events

Fans and Sales (one-to-many)

- A fan can make many sales
- A sale can be made by one and only one fan

ASSOCIATIVE ENTITY:

Enrollments (many-to-many of artists and trainings):

- (PK) enrollment_id - VC(10)
- (FK) artist_id - VC(10)
- FK) training_id - VC(10)
- enrollment_start_date - Date

Participations (many to many of artists and events):

- (PK) participation_id - VC(10)
- (FK) artist_id - VC(10)
- (FK) event_id - VC(10)

Market Targeted Contents (many to many of contents and markets):

- (PK/FK) market_id - VC(10)
- (PK/FK) content_id - VC(10)

Merch Sales (many to many of merch and sales):

- (PK) merch_sale_id - VC(10)
- (FK) merch_id - VC(10)
- (FK) sale_id - VC(10)
- merch_sale_quantity - int
- merch_sale_price - DECIMAL(12,2)

Event Sales (many to many of events and sales):

- (PK) event_sale_id - VC(10)
- (FK) event_id - VC(10)
- (FK) sale_id - VC(10)
- event_sale_quantity - int
- event_sale_price - DECIMAL(12,2)

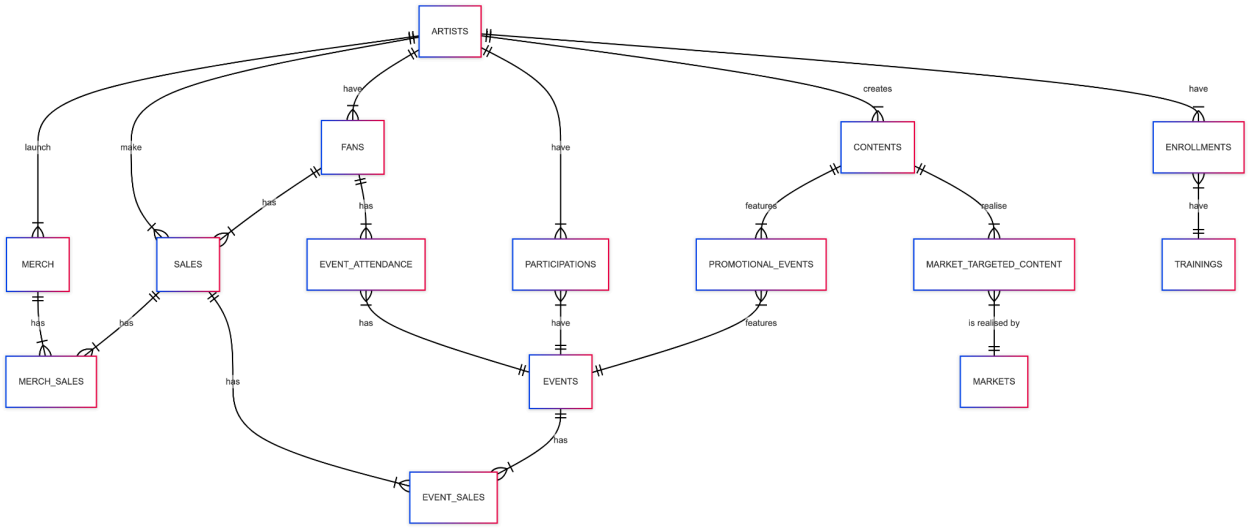
Event Attendance (many to many of fans and events):

- (PK) attendance_id - VC(10)
- (FK) fan_id - VC(10)
- (FK) event_id - VC(10)
- ticket_type - VC(20)
- attendance_status - VC(20)

Promotional Event (many to many of contents and events):

- (PK) promo_event_id - VC(10)
- (FK) event_id - VC(10)
- (FK) content_id - VC(10)
- promo_event_type - VC(50)

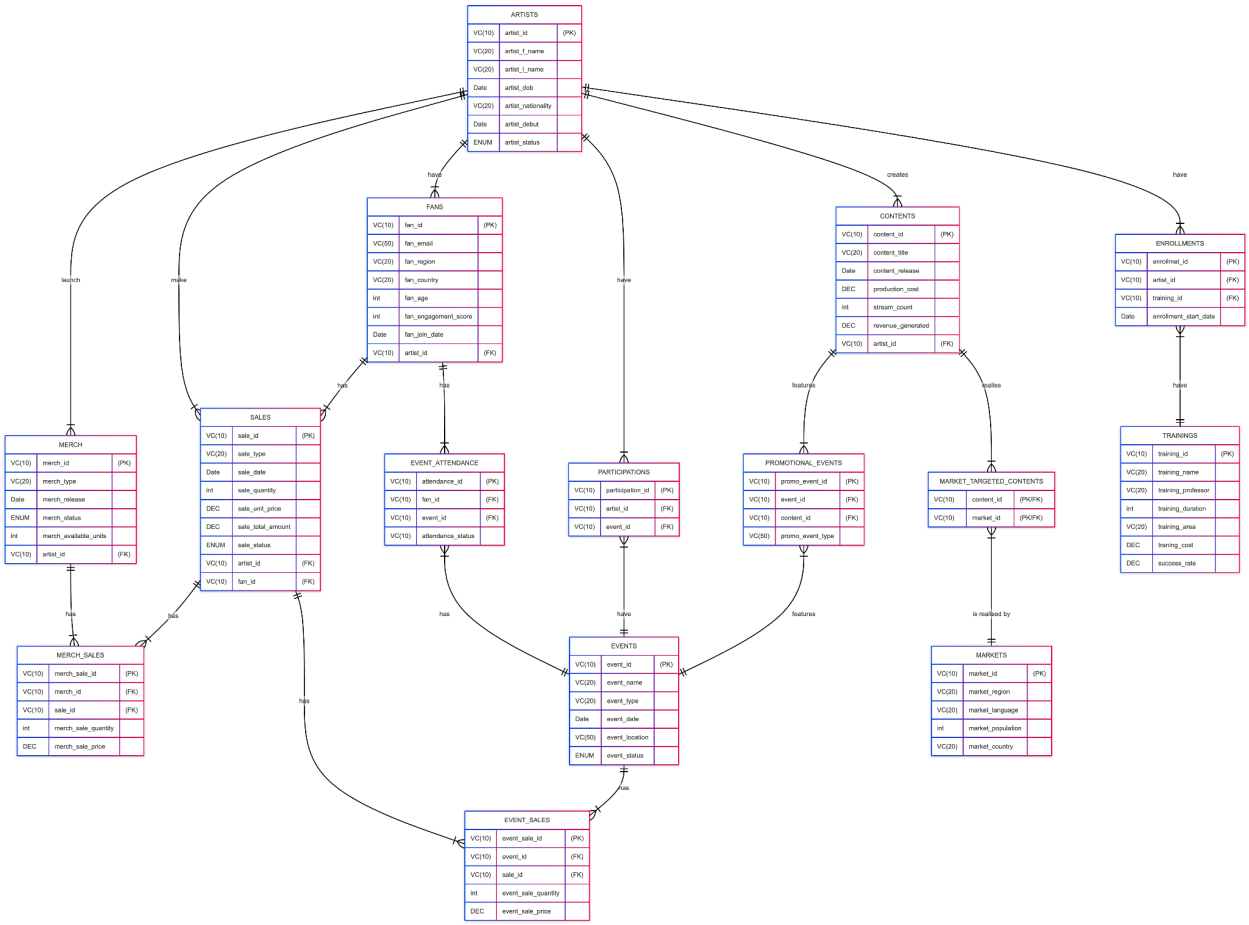
ERD CREATION:



MERMAID CODE:

```
erDiagram
ARTISTS|| -- |{ MERCH: "launch"
MERCH|| -- |{MERCH_SALES: "has"
ARTISTS|| -- |{ SALES: "make"
SALES|| -- |{MERCH_SALES: "has"
ARTISTS|| -- |{ FANS: "have"
FANS|| -- |{SALES: "has"
ARTISTS|| -- |{ PARTICIPATIONS: "have"
PARTICIPATIONS|{ -- || EVENTS: "have"
FANS|| -- |{EVENT_ATTENDANCE: "has"
EVENT_ATTENDANCE|{ -- ||EVENTS: "has"
SALES|| -- |{EVENT_SALES: "has"
EVENTS|| -- |{EVENT_SALES: "has"
ARTISTS|| -- |{ CONTENTS: "creates"
CONTENTS|| -- |{PROMOTIONAL_EVENTS: "features"
PROMOTIONAL_EVENTS|{ -- ||EVENTS: "features"
CONTENTS|| -- |{ MARKET_TARGETED_CONTENT: "realise"
MARKET_TARGETED_CONTENT|{ -- || MARKETS: "is realised by"
ARTISTS|| -- |{ ENROLLMENTS: "have"
ENROLLMENTS|{ -- || TRAININGS: "have"
```

LOGICAL MODEL:



MERMAID CODE:

```
erDiagram
ENROLLMENTS{
VC(10) enrollment_id "(PK)"
VC(10) artist_id "(FK)"
VC(10) training_id "(FK)"
Date enrollment_start_date
}
ARTISTS{
VC(10) artist_id "(PK)"
VC(20) artist_f_name
VC(20) artist_l_name
Date artist_dob
VC(20) artist_nationality
Date artist_debut
ENUM artist_status
}
TRAININGS{
VC(10) training_id "(PK)"
VC(20) training_name
VC(20) training_professor
int training_duration
VC(20) training_area
```

```
DEC training_cost
DEC success_rate
}
PARTICIPATIONS{
VC(10) participation_id "(PK)"
VC(10) artist_id "(FK)"
VC(10) event_id "(FK)"
}
EVENTS{
VC(10) event_id "(PK)"
VC(20) event_name
VC(20) event_type
Date event_date
VC(50) event_location
ENUM event_status
}
FANS{
VC(10) fan_id "(PK)"
VC(50) fan_email
VC(20) fan_region
VC(20) fan_country
int fan_age
int fan_engagement_score
Date fan_join_date
VC(10) artist_id "(FK)"
}
CONTENTS{
VC(10) content_id "(PK)"
VC(20) content_title
Date content_release
DEC production_cost
int stream_count
DEC revenue_generated
VC(10) artist_id "(FK)"
}
MARKET_TARGETED_CONTENTS{
VC(10) content_id "(PK/FK)"
VC(10) market_id "(PK/FK)"
}
MARKETS{
VC(10) market_id "(PK)"
VC(20) market_region
VC(20) market_language
int market_population
VC(20) market_country
}
MERCH{
 VC(10) merch_id "(PK)"
 VC(20) merch_type
 Date merch_release
 ENUM merch_status
 int merch_available_units
```

```
 VC(10) artist_id "(FK)"
}
SALES{
 VC(10) sale_id "(PK)"
 VC(20) sale_type
 Date sale_date
 int sale_quantity
 DEC sale_unit_price
 DEC sale_total_amount
 ENUM sale_status
 VC(10) artist_id "(FK)"
 VC(10) fan_id "(FK)"
}
MERCH_SALES{
 VC(10) merch_sale_id "(PK)"
 VC(10) merch_id "(FK)"
 VC(10) sale_id "(FK)"
 int merch_sale_quantity
 DEC merch_sale_price
}
EVENT_SALES{
 VC(10) event_sale_id "(PK)"
 VC(10) event_id "(FK)"
 VC(10) sale_id "(FK)"
 int event_sale_quantity
 DEC event_sale_price
}
EVENT_ATTENDANCE{
 VC(10) attendance_id "(PK)"
 VC(10) fan_id "(FK)"
 VC(10) event_id "(FK)"
 VC(10) attendance_status
}
PROMOTIONAL_EVENTS{
 VC(10) promo_event_id "(PK)"
 VC(10) event_id "(FK)"
 VC(10) content_id "(FK)"
 VC(50) promo_event_type
}
```

NORMALIZATION:

1NF:
Every table in the database includes primary keys like (artist_id) and (content_id) to ensure unique identification, and all attributes are kept atomic, meaning each field holds just one single value, such as (artist_f_name) instead of combining values like a full name.

2NF:
To meet 2NF, the database starts in 1NF, and all non-key attributes must fully depend on the entire primary key. In the diagram, tables like MARKET_TARGETED_CONTENTS,

ENROLLMENTS, and PARTICIPATIONS, which have composite primary keys, only include attributes that rely on the combined keys, avoiding any partial dependencies. For example, in CONTENT_MARKET, both content_id and market_id are needed to define the relationship, and in ENROLLMENTS, enrollment_start_date depends on both artist_id and training_id, ensuring all non-key attributes are fully tied to the whole primary key.

3NF:
In this database, no non-key attribute depends on another non-key attribute, as all attributes rely only on the primary key. For example, in the ARTISTS table, (artist_dob) and (artist_nationality) depend solely on (artist_id), and in the CONTENTS table, (content_title) and (content_release) depend only on (content_id). This structure keeps each table properly normalized.

ARTISTS:

artist_id	artist_f_name	artist_l_name	artist_dob	artist_nationality	artist_debut	artist_status

artist_id → (determining) → artist_f_name, artist_l_name, artist_dob, artist_nationality, artist_debut, artist_status.

ENROLLMENTS:

enrollment_id	artist_id	training_id	enrollment_start_date

enrollment_id → (determining) → artist_id, training_id, Enrollment_start_date.

TRAINING:

training_id	training_name	training_professor	training_duration	training_area	training_cost	success_rate

training_id → (determining) → training_name, training_professor, training_duration, training_area, training_cost, success_rate.

PARTICIPATIONS:

participation_id	artist_id	event_id

participation_id → (determining) → artist_id, event_id, event_id.

EVENTS:

event_id	event_name	event_type	event_date	event_location	event_status

event_id → (determining) → event_name, event_type, event_date, event_location, event_status.

FANS:

fan_id	fan_region	fan_country	fan_age	artist_id	fan_email	fan_engagement_score	fan_join_date

fan_id → (determining) → fan_region, fan_country, fan_age, artist_id, fan_email, fan_engagement_score, fan_join_date.

CONTENTS:

content_id	content_title	content_release	artist_id	content_type	production_cost	stream_count	revenue_generated

content_id → (determining) → content_title, content_release, artist_id, content_type, production_cost, stream_count, revenue_generated.

MARKET_TARGETED_CONTENTS:

market_targeted_content_id	market_id	content_id

market_targeted_content_id → (determining) → market_id, content_id.

MARKETS:

market_id	market_region	market_language	market_population	market_country

market_id → (determining) → market_region, market_language, market_population, market_country.

MERCH:

merch_id	merch_type	merch_release	merch_status	merch_available_units

merch_id → (determining) → merch_type, merch_release, merch_status, merch_available_units.

SALES:

sale_id	sale_type	sale_date	sale_quantity	sale_unit_price	sale_total_amount	sale_status

sale_id → (determining) → sale_type, sale_date, sale_quantity, sale_unit_price, sale_total_amount, sale_status.

MERCH SALES:

merch_sale_id	merch_id	sale_id	merch_sale_quantity	merch_sale_price

merch_sale_id → (determining) → merch_id, sale_id, merch_sale_quantity, merch_sale_price.

EVENT SALES:

event_sale_id	event_id	sale_id	event_sale_quantity	event_sale_price

event_sale_id → (determining) → event_id, sale_id, event_sale_quantity, event_sale_price.

EVENT ATTENDANCE:

attendance_id	fan_id	event_id	ticket_type	attendance_status

attendance_id → (determining) → fan_id, event_id, ticket_type, attendance_status.

PROMOTIONAL EVENTS:

promo_event_id	event_id	content_id	promo_event_type

promo_event_id → (determining) → event_id, content_id, promo_event_type.

SQL:

1. **Database Creation:**

```
-- Create Database
CREATE DATABASE IF NOT EXISTS galaxy_entertainment;
USE galaxy_entertainment;

-- Table: ARTISTS
CREATE TABLE ARTISTS (
    artist_id VARCHAR(10) PRIMARY KEY,
    artist_f_name VARCHAR(20) NOT NULL,
    artist_l_name VARCHAR(20) NOT NULL,
    artist_dob DATE,
    artist_nationality VARCHAR(20),
    artist_debut DATE,
    artist_status ENUM('active', 'inactive', 'retired')
);

-- Table: TRAININGS
CREATE TABLE TRAININGS (
    training_id VARCHAR(10) PRIMARY KEY,
    training_name VARCHAR(20) NOT NULL,
    training_professor VARCHAR(20),
    training_duration INT,
    training_area VARCHAR(20),
```

```
    training_cost DECIMAL(8,2),
    success_rate DECIMAL(5,2)
);

-- Table: CONTENTS
CREATE TABLE CONTENTS (
    content_id VARCHAR(10) PRIMARY KEY,
    content_title VARCHAR(20) NOT NULL,
    content_type VARCHAR(20),
    content_release DATE,
    production_cost DECIMAL(10,2),
    stream_count INT,
    revenue_generated DECIMAL(12,2),
    artist_id VARCHAR(10),
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id)
);

-- Table: MARKETS
CREATE TABLE MARKETS (
    market_id VARCHAR(10) PRIMARY KEY,
    market_region VARCHAR(20),
    market_language VARCHAR(20),
    market_population INT,
```

```
    market_country VARCHAR(20)
);

-- Table: EVENTS
CREATE TABLE EVENTS (
    event_id VARCHAR(10) PRIMARY KEY,
    event_name VARCHAR(20) NOT NULL,
    event_type VARCHAR(20),
    event_date DATE,
    event_location VARCHAR(50),
    event_status ENUM('scheduled', 'cancelled', 'completed')
);

-- Table: FANS
CREATE TABLE FANS (
    fan_id VARCHAR(10) PRIMARY KEY,
    artist_id VARCHAR(10),
    fan_email VARCHAR(50),
    fan_country VARCHAR(20),
    fan_region VARCHAR(20),
    fan_age INT,
    fan_engagement_score INT,
    fan_join_date DATE,
```

```
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id)
);

-- Table: MERCH
CREATE TABLE MERCH (
    merch_id VARCHAR(10) PRIMARY KEY,
    artist_id VARCHAR(10),
    merch_type VARCHAR(20),
    merch_release DATE,
    merch_status ENUM('available', 'discontinued'),
    merch_available_units INT,
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id)
);

-- Table: SALES
CREATE TABLE SALES (
    sale_id VARCHAR(10) PRIMARY KEY,
    artist_id VARCHAR(10),
    fan_id VARCHAR(10),
    sale_type VARCHAR(20),
    sale_date DATE,
    sale_quantity INT,
    sale_unit_price DECIMAL(8,2),
```

```
    sale_total_amount DECIMAL(10,2),
    sale_status ENUM('pending', 'completed', 'refunded'),
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id),
    FOREIGN KEY (fan_id) REFERENCES FANS(fan_id)
);

-- Associative Table: ENROLLMENTS
CREATE TABLE ENROLLMENTS (
    enrollment_id VARCHAR(10) PRIMARY KEY,
    artist_id VARCHAR(10),
    training_id VARCHAR(10),
    enrollment_start_date DATE,
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id),
    FOREIGN KEY (training_id) REFERENCES TRAININGS(training_id)
);

-- Associative Table: PARTICIPATIONS
CREATE TABLE PARTICIPATIONS (
    participation_id VARCHAR(10) PRIMARY KEY,
    artist_id VARCHAR(10),
    event_id VARCHAR(10),
    FOREIGN KEY (artist_id) REFERENCES ARTISTS(artist_id),
    FOREIGN KEY (event_id) REFERENCES EVENTS(event_id)
```

```
);

-- Associative Table: MARKET_TARGETED_CONTENTS
CREATE TABLE MARKET_TARGETED_CONTENTS (
    market_id VARCHAR(10),
    content_id VARCHAR(10),
    PRIMARY KEY (market_id, content_id),
    FOREIGN KEY (market_id) REFERENCES MARKETS(market_id),
    FOREIGN KEY (content_id) REFERENCES CONTENTS(content_id)
);

-- Associative Table: MERCH_SALES
CREATE TABLE MERCH_SALES (
    merch_sale_id VARCHAR(10) PRIMARY KEY,
    merch_id VARCHAR(10),
    sale_id VARCHAR(10),
    merch_sale_quantity INT,
    merch_sale_price DECIMAL(12,2),
    FOREIGN KEY (merch_id) REFERENCES MERCH(merch_id),
    FOREIGN KEY (sale_id) REFERENCES SALES(sale_id)
);

-- Associative Table: EVENT_SALES
```

```
CREATE TABLE EVENT_SALES (
    event_sale_id VARCHAR(10) PRIMARY KEY,
    event_id VARCHAR(10),
    sale_id VARCHAR(10),
    event_sale_quantity INT,
    event_sale_price DECIMAL(12,2),
    FOREIGN KEY (event_id) REFERENCES EVENTS(event_id),
    FOREIGN KEY (sale_id) REFERENCES SALES(sale_id)
);

-- Associative Table: EVENT_ATTENDANCE
CREATE TABLE EVENT_ATTENDANCE (
    attendance_id VARCHAR(10) PRIMARY KEY,
    fan_id VARCHAR(10),
    event_id VARCHAR(10),
    ticket_type VARCHAR(20),
    attendance_status VARCHAR(20),
    FOREIGN KEY (fan_id) REFERENCES FANS(fan_id),
    FOREIGN KEY (event_id) REFERENCES EVENTS(event_id)
);

-- Associative Table: PROMOTIONAL_EVENT
CREATE TABLE PROMOTIONAL_EVENT (
```

```
    promo_event_id VARCHAR(10) PRIMARY KEY,
    event_id VARCHAR(10),
    content_id VARCHAR(10),
    promo_event_type VARCHAR(50),
    FOREIGN KEY (event_id) REFERENCES EVENTS(event_id),
    FOREIGN KEY (content_id) REFERENCES CONTENTS(content_id)
);

-- Create indexes for better performance
CREATE INDEX idx_artists_status ON ARTISTS(artist_status);
CREATE INDEX idx_contents_release ON CONTENTS(content_release);
CREATE INDEX idx_events_date ON EVENTS(event_date);
CREATE INDEX idx_fans_engagement ON FANS(fan_engagement_score);
CREATE INDEX idx_sales_date ON SALES(sale_date);
CREATE INDEX idx_merch_status ON MERCH(merch_status);
CREATE INDEX idx_enrollments_start ON ENROLLMENTS(enrollment_start_date);
```

2. **Data Insertion:**

```
-- Insert sample data into ARTISTS
INSERT INTO ARTISTS (artist_id, artist_f_name, artist_l_name, artist_dob,
artist_nationality, artist_debut, artist_status) VALUES
('A001', 'John', 'Doe', '1990-05-15', 'American', '2010-06-01', 'active'),
('A002', 'Jane', 'Smith', '1985-12-23', 'British', '2008-03-15', 'active'),
('A003', 'Carlos', 'Gomez', '1993-07-09', 'Spanish', '2012-09-20', 'active'),
```

('A004', 'Akira', 'Tanaka', '1992-01-18', 'Japanese', '2011-11-25', 'inactive'),

('A005', 'Lina', 'Khan', '1995-04-28', 'Indian', '2015-08-17', 'active'),

('A006', 'Emily', 'Johnson', '1988-10-10', 'Canadian', '2009-05-12', 'active'),

('A007', 'Marcus', 'Lee', '1991-03-22', 'Australian', '2010-10-30', 'active'),

('A008', 'Sophia', 'Brown', '1994-08-19', 'French', '2013-07-11', 'inactive'),

('A009', 'Noah', 'Miller', '1990-02-25', 'German', '2009-12-05', 'retired'),

('A010', 'Zara', 'Ali', '1996-06-15', 'Pakistani', '2016-04-09', 'active');

-- Insert sample data into TRAININGS

INSERT INTO TRAININGS (training_id, training_name, training_professor, training_duration, training_area, training_cost, success_rate) VALUES

('T001', 'Vocal Training', 'Dr. Hill', 12, 'Music', 5000.00, 85.50),

('T002', 'Stage Presence', 'Prof. Collins', 8, 'Performance', 3500.00, 78.25),

('T003', 'Dance Choreography', 'Ms. Rivera', 16, 'Dance', 7200.00, 92.00),

('T004', 'Media Relations', 'Dr. Nguyen', 6, 'Communication', 2800.00, 65.75),

('T005', 'Song Writing', 'Mr. Davis', 10, 'Music', 4200.00, 88.90),

('T006', 'Acting Workshop', 'Ms. Taylor', 8, 'Acting', 3800.00, 72.40),

('T007', 'Instrument Mastery', 'Dr. Lewis', 20, 'Music', 8500.00, 91.25),

('T008', 'Fitness Training', 'Prof. Kim', 12, 'Health', 4200.00, 68.90),

('T009', 'Social Media', 'Mr. Patel', 6, 'Marketing', 2200.00, 79.60),

('T010', 'Language Course', 'Ms. Clark', 16, 'Language', 5800.00, 84.75);

-- Insert sample data into CONTENTS

```
INSERT INTO CONTENTS (content_id, content_title, content_type, content_release, production_cost, stream_count, revenue_generated, artist_id) VALUES

('C001', 'Summer Nights', 'Music Video', '2023-06-15', 25000.00, 2500000, 75000.00, 'A001'),

('C002', 'Winter Symphony', 'Album', '2023-11-20', 18000.00, 1800000, 65000.00, 'A002'),

('C003', 'Dance Revolution', 'Single', '2023-08-10', 12000.00, 3200000, 98000.00, 'A003'),

('C004', 'Tokyo Lights', 'EP', '2023-03-05', 15000.00, 1200000, 42000.00, 'A004'),

('C005', 'Bollywood Beat', 'Music Video', '2023-09-12', 28000.00, 4500000, 125000.00, 'A005'),

('C006', 'Urban Legends', 'Album', '2023-04-18', 22000.00, 2800000, 82000.00, 'A006'),

('C007', 'Ocean Waves', 'Single', '2023-07-22', 14000.00, 1900000, 58000.00, 'A007'),

('C008', 'Paris Nights', 'EP', '2023-02-14', 16000.00, 1500000, 48000.00, 'A008'),

('C009', 'Berlin Beats', 'Music Video', '2023-10-05', 32000.00, 3800000, 112000.00, 'A009'),

('C010', 'Desert Rose', 'Single', '2023-12-01', 18000.00, 2600000, 76000.00, 'A010');


-- Insert sample data into MARKETS

INSERT INTO MARKETS (market_id, market_region, market_language, market_population, market_country) VALUES

('M001', 'North America', 'English', 350000000, 'USA'),

('M002', 'Europe', 'English', 67000000, 'UK'),

('M003', 'Asia', 'Hindi', 1380000000, 'India'),

('M004', 'East Asia', 'Japanese', 125000000, 'Japan'),

('M005', 'Europe', 'Spanish', 47000000, 'Spain'),
```

('C006', 'Urban Legends', 'Album', '2023-04-18', 22000.00, 2800000, 82000.00, 'A006'),

('C007', 'Ocean Waves', 'Single', '2023-07-22', 14000.00, 1900000, 58000.00, 'A007'),

('C008', 'Paris Nights', 'EP', '2023-02-14', 16000.00, 1500000, 48000.00, 'A008'),

('C009', 'Berlin Beats', 'Music Video', '2023-10-05', 32000.00, 3800000, 112000.00, 'A009'),

('C010', 'Desert Rose', 'Single', '2023-12-01', 18000.00, 2600000, 76000.00, 'A010');

-- Insert sample data into EVENTS

INSERT INTO EVENTS (event_id, event_name, event_type, event_date, event_location, event_status) VALUES

('EV001', 'Summer Fest 2023', 'Concert', '2023-07-15', 'Los Angeles', 'completed'),

('EV002', 'Music Awards', 'Award Show', '2023-09-20', 'London', 'completed'),

('EV003', 'Asian Tour', 'Concert', '2023-11-05', 'Tokyo', 'scheduled'),

('EV004', 'Fan Meeting', 'Meet & Greet', '2023-12-10', 'Mumbai', 'scheduled'),

('EV005', 'Charity Gala', 'Fundraiser', '2023-10-15', 'Madrid', 'completed'),

('EV006', 'Winter Concert', 'Concert', '2023-12-20', 'Toronto', 'scheduled'),

('EV007', 'Fan Appreciation', 'Meet & Greet', '2023-11-30', 'Sydney', 'scheduled'),

('EV008', 'Music Festival', 'Festival', '2023-08-25', 'Rio de Janeiro', 'completed'),

('EV009', 'Album Launch', 'Launch Event', '2023-10-08', 'Seoul', 'completed'),

('EV010', 'Charity Concert', 'Fundraiser', '2023-09-18', 'Dubai', 'completed');

-- Insert sample data into FANS

INSERT INTO FANS (fan_id, artist_id, fan_email, fan_country, fan_region, fan_age, fan_engagement_score, fan_join_date) VALUES

('F001', 'A001', 'fan1@email.com', 'USA', 'North America', 25, 85, '2022-01-15'),

('F002', 'A002', 'fan2@email.com', 'UK', 'Europe', 30, 92, '2021-03-20'),

('F003', 'A003', 'fan3@email.com', 'Spain', 'Europe', 22, 78, '2022-08-10'),

('F004', 'A004', 'fan4@email.com', 'Japan', 'Asia', 28, 65, '2021-11-05'),

('F005', 'A005', 'fan5@email.com', 'India', 'Asia', 24, 88, '2022-05-30'),

('F006', 'A006', 'fan6@email.com', 'Canada', 'North America', 27, 81, '2021-07-12'),

('F007', 'A007', 'fan7@email.com', 'Australia', 'Oceania', 31, 76, '2022-03-25'),

('F008', 'A008', 'fan8@email.com', 'France', 'Europe', 23, 69, '2023-01-08'),

('F009', 'A009', 'fan9@email.com', 'Germany', 'Europe', 29, 94, '2020-11-15'),

('F010', 'A010', 'fan10@email.com', 'Pakistan', 'Asia', 26, 87, '2022-09-20');

-- Insert sample data into MERCH

INSERT INTO MERCH (merch_id, artist_id, merch_type, merch_release, merch_status, merch_available_units) VALUES

('ME001', 'A001', 'T-Shirt', '2023-06-01', 'available', 500),

('ME002', 'A002', 'Poster', '2023-07-15', 'available', 1000),

('ME003', 'A003', 'Album', '2023-08-20', 'available', 800),

('ME004', 'A004', 'Photobook', '2023-05-10', 'discontinued', 0),

('ME005', 'A005', 'Hoodie', '2023-09-05', 'available', 300),

('ME006', 'A006', 'Cap', '2023-07-08', 'available', 400),

('ME007', 'A007', 'Vinyl', '2023-08-25', 'available', 250),

('ME008', 'A008', 'Keychain', '2023-04-12', 'discontinued', 0),

('ME009', 'A009', 'Limited Edition', '2023-11-30', 'available', 150),

('ME010', 'A010', 'Tote Bag', '2023-09-18', 'available', 600);

-- Insert sample data into ENROLLMENTS

INSERT INTO ENROLLMENTS (enrollment_id, artist_id, training_id, enrollment_start_date) VALUES

('E001', 'A001', 'T001', '2023-01-10'),

('E002', 'A002', 'T002', '2023-02-15'),

('E003', 'A003', 'T003', '2023-01-25'),

('E004', 'A004', 'T004', '2023-03-05'),

('E005', 'A005', 'T005', '2023-02-20'),

('E006', 'A006', 'T006', '2023-03-12'),

('E007', 'A007', 'T007', '2023-01-18'),

('E008', 'A008', 'T008', '2023-04-02'),

('E009', 'A009', 'T009', '2023-02-28'),

('E010', 'A010', 'T010', '2023-03-22');

-- Insert sample data into MARKET_TARGETED_CONTENTS

INSERT INTO MARKET_TARGETED_CONTENTS (market_id, content_id) VALUES

('M001', 'C001'),

('M002', 'C002'),

('M003', 'C005'),

('M004', 'C004'),

('M005', 'C003'),

('M006', 'C008'),

('M007', 'C006'),

('M008', 'C007'),

('M009', 'C009'),

('M010', 'C010');

-- Insert sample data into SALES (10 rows)

INSERT INTO SALES (sale_id, artist_id, fan_id, sale_type, sale_date, sale_quantity, sale_unit_price, sale_total_amount, sale_status) VALUES

('S001', 'A001', 'F001', 'merch', '2023-07-20', 2, 25.00, 50.00, 'completed'),

('S002', 'A002', 'F002', 'ticket', '2023-09-25', 1, 75.00, 75.00, 'completed'),

('S003', 'A003', 'F003', 'merch', '2023-08-15', 3, 30.00, 90.00, 'completed'),

('S004', 'A004', 'F004', 'album', '2023-06-10', 1, 15.00, 15.00, 'completed'),

('S005', 'A005', 'F005', 'merch', '2023-10-05', 2, 40.00, 80.00, 'completed'),

('S006', 'A006', 'F006', 'ticket', '2023-11-28', 2, 60.00, 120.00, 'pending'),

('S007', 'A007', 'F007', 'merch', '2023-09-12', 1, 35.00, 35.00, 'completed'),

('S008', 'A008', 'F008', 'album', '2023-05-22', 1, 12.00, 12.00, 'refunded'),

('S009', 'A009', 'F009', 'merch', '2023-12-01', 1, 50.00, 50.00, 'completed'),

('S010', 'A010', 'F010', 'ticket', '2023-10-30', 1, 45.00, 45.00, 'completed');

-- Insert sample data into PARTICIPATIONS (10 rows)

INSERT INTO PARTICIPATIONS (participation_id, artist_id, event_id) VALUES

('P001', 'A001', 'EV001'),

('P002', 'A002', 'EV002'),

('P003', 'A003', 'EV003'),

('P004', 'A004', 'EV004'),

('P005', 'A005', 'EV005'),

('P006', 'A006', 'EV006'),

('P007', 'A007', 'EV007'),

('P008', 'A008', 'EV008'),

('P009', 'A009', 'EV009'),

('P010', 'A010', 'EV010');

-- Insert sample data into MERCH_SALES (10 rows)

INSERT INTO MERCH_SALES (merch_sale_id, merch_id, sale_id,
merch_sale_quantity, merch_sale_price) VALUES

('MS001', 'ME001', 'S001', 2, 50.00),

('MS002', 'ME002', 'S003', 3, 90.00),

('MS003', 'ME003', 'S004', 1, 15.00),

('MS004', 'ME005', 'S005', 2, 80.00),

('MS005', 'ME006', 'S007', 1, 35.00),

('MS006', 'ME007', 'S008', 1, 12.00),

('MS007', 'ME009', 'S009', 1, 50.00),

('MS008', 'ME010', 'S003', 1, 30.00),

('MS009', 'ME001', 'S007', 1, 25.00),

('MS010', 'ME005', 'S001', 1, 40.00);

-- Insert sample data into EVENT_SALES (10 rows)

INSERT INTO EVENT_SALES (event_sale_id, event_id, sale_id, event_sale_quantity,
event_sale_price) VALUES

('ES001', 'EV001', 'S002', 1, 75.00),

('ES002', 'EV002', 'S006', 2, 120.00),

('ES003', 'EV003', 'S010', 1, 45.00),

('ES004', 'EV004', 'S002', 1, 75.00),

('ES005', 'EV005', 'S006', 1, 60.00),

('ES006', 'EV006', 'S010', 1, 45.00),

('ES007', 'EV007', 'S002', 1, 75.00),

('ES008', 'EV008', 'S006', 2, 120.00),

('ES009', 'EV009', 'S010', 1, 45.00),

('ES010', 'EV010', 'S002', 1, 75.00);

-- Insert sample data into EVENT_ATTENDANCE (10 rows)

INSERT INTO EVENT_ATTENDANCE (attendance_id, fan_id, event_id, ticket_type, attendance_status) VALUES

('AT001', 'F001', 'EV001', 'VIP', 'confirmed'),

('AT002', 'F002', 'EV002', 'Standard', 'confirmed'),

('AT003', 'F003', 'EV003', 'Premium', 'pending'),

('AT004', 'F004', 'EV004', 'Standard', 'confirmed'),

('AT005', 'F005', 'EV005', 'VIP', 'confirmed'),

('AT006', 'F006', 'EV006', 'Standard', 'pending'),

('AT007', 'F007', 'EV007', 'Premium', 'confirmed'),

('AT008', 'F008', 'EV008', 'Standard', 'cancelled'),

('AT009', 'F009', 'EV009', 'VIP', 'confirmed'),

('AT010', 'F010', 'EV010', 'Standard', 'confirmed');

-- Insert sample data into PROMOTIONAL_EVENT (10 rows)

INSERT INTO PROMOTIONAL_EVENT (promo_event_id, event_id, content_id, promo_event_type) VALUES

('PE001', 'EV001', 'C001', 'Album Launch Promotion'),

('PE002', 'EV002', 'C002', 'Award Show Performance'),

('PE003', 'EV003', 'C003', 'Tour Promotion'),

('PE004', 'EV004', 'C004', 'Fan Meeting Content'),

('PE005', 'EV005', 'C005', 'Charity Event Feature'),

('PE006', 'EV006', 'C006', 'Winter Concert Promotion'),

('PE007', 'EV007', 'C007', 'Fan Appreciation Content'),

('PE008', 'EV008', 'C008', 'Festival Performance'),

('PE009', 'EV009', 'C009', 'Album Launch Event'),

('PE010', 'EV010', 'C010', 'Charity Concert Feature');

3. SQL Queries and Reports:

Key Business Questions:

- Which artists and training programs deliver the best ROI?
- What content types perform best in different markets?
- How can we optimize fan engagement and conversion?
- Which markets show the most growth potential?

- **Artist Performance Dashboard**

```
-- Artist ROI Analysis: Calculate return on investment for each artist
SELECT
   a.artist_id,
   CONCAT(a.artist_f_name, ' ', a.artist_l_name) as artist_name,
   a.artist_status,
   COUNT(DISTINCT c.content_id) as content_count,
   SUM(c.stream_count) as total_streams,
   SUM(c.revenue_generated) as total_revenue,
```

```
    SUM(c.production_cost) as total_production_cost,
    (SUM(c.revenue_generated) - SUM(c.production_cost)) as net_profit,
    ROUND(((SUM(c.revenue_generated) - SUM(c.production_cost)) /
NULLIF(SUM(c.production_cost), 0)) * 100, 2) as roi_percentage
FROM ARTISTS a
LEFT JOIN CONTENTS c ON a.artist_id = c.artist_id
GROUP BY a.artist_id, artist_name, a.artist_status
ORDER BY roi_percentage DESC;
```

- **Training Program Effectiveness**

```
-- Training ROI Analysis: Which training programs deliver best results?
SELECT
    t.training_name,
    t.training_area,
    t.training_cost,
    t.success_rate,
    COUNT(DISTINCT e.artist_id) as artists_trained,
    AVG(c.stream_count) as avg_streams_post_training,
    AVG(c.revenue_generated) as avg_revenue_post_training,
    ROUND((AVG(c.revenue_generated) - t.training_cost) / NULLIF(t.training_cost, 0) * 100, 2)
as training_roi
FROM TRAININGS t
LEFT JOIN ENROLLMENTS e ON t.training_id = e.training_id
LEFT JOIN ARTISTS a ON e.artist_id = a.artist_id
LEFT JOIN CONTENTS c ON a.artist_id = c.artist_id AND c.content_release >
e.enrollment_start_date
GROUP BY t.training_id, t.training_name, t.training_area, t.training_cost, t.success_rate
ORDER BY training_roi DESC;
```

- **Market Performance Analysis**

```
-- Market Revenue Performance with Rankings
SELECT
    m.market_region,
    m.market_country,
    COUNT(DISTINCT mtc.content_id) as content_count,
    SUM(c.stream_count) as total_streams,
    SUM(c.revenue_generated) as total_revenue,
    ROUND(SUM(c.revenue_generated) / NULLIF(COUNT(DISTINCT mtc.content_id), 0), 2) as
revenue_per_content,
    RANK() OVER (ORDER BY SUM(c.revenue_generated) DESC) as revenue_rank,
    ROUND((SUM(c.stream_count) / NULLIF(m.market_population, 0)) * 100, 4) as
market_penetration_rate
FROM MARKETS m
JOIN MARKET_TARGETED_CONTENTS mtc ON m.market_id = mtc.market_id
```

```
JOIN CONTENTS c ON mtc.content_id = c.content_id
GROUP BY m.market_region, m.market_country, m.market_population
ORDER BY total_revenue DESC;
```

- **Fan Engagement Analysis**

```
-- Fan Segmentation by Engagement and Value
SELECT
   f.fan_region,
   f.fan_country,
   COUNT(f.fan_id) as total_fans,
   AVG(f.fan_engagement_score) as avg_engagement,
   COUNT(s.sale_id) as total_purchases,
   COALESCE(SUM(s.sale_total_amount), 0) as total_revenue,
   CASE
      WHEN AVG(f.fan_engagement_score) >= 90 THEN 'Super Fan'
      WHEN AVG(f.fan_engagement_score) >= 75 THEN 'Engaged Fan'
      WHEN AVG(f.fan_engagement_score) >= 50 THEN 'Casual Fan'
      ELSE 'Low Engagement'
   END as fan_segment,
   ROUND(COALESCE(SUM(s.sale_total_amount), 0) / NULLIF(COUNT(f.fan_id), 0), 2) as
revenue_per_fan
FROM FANS f
LEFT JOIN SALES s ON f.fan_id = s.fan_id AND s.sale_status = 'completed'
GROUP BY f.fan_region, f.fan_country
ORDER BY revenue_per_fan DESC;
```

- **Content Performance Trends**

```
-- Monthly Content Performance with Growth Rates
SELECT
   DATE_FORMAT(c.content_release, '%Y-%m') as release_month,
   c.content_type,
   COUNT(c.content_id) as content_count,
   AVG(c.stream_count) as avg_streams,
   AVG(c.revenue_generated) as avg_revenue,
   SUM(c.revenue_generated) as monthly_revenue,
   LAG(SUM(c.revenue_generated)) OVER (ORDER BY DATE_FORMAT(c.content_release,
'%Y-%m')) as prev_month_revenue,
   ROUND(
      ((SUM(c.revenue_generated) - LAG(SUM(c.revenue_generated)) OVER (ORDER BY
DATE_FORMAT(c.content_release, '%Y-%m')))
      / NULLIF(LAG(SUM(c.revenue_generated)) OVER (ORDER BY
DATE_FORMAT(c.content_release, '%Y-%m')), 0)) * 100, 2
   ) as monthly_growth_rate
FROM CONTENTS c
```

```
WHERE c.content_release IS NOT NULL
GROUP BY DATE_FORMAT(c.content_release, '%Y-%m'), c.content_type
ORDER BY release_month, monthly_revenue DESC;
```

- **Merchendise Performance Analysis**

```
-- Merch Performance by Type and Artist
SELECT
   a.artist_id,
   CONCAT(a.artist_f_name, ' ', a.artist_l_name) as artist_name,
   m.merch_type,
   m.merch_status,
   SUM(ms.merch_sale_quantity) as total_units_sold,
   SUM(ms.merch_sale_quantity * ms.merch_sale_price) as total_revenue,
   m.merch_available_units as current_inventory,
   ROUND(SUM(ms.merch_sale_quantity) / NULLIF((SUM(ms.merch_sale_quantity) +
m.merch_available_units), 0) * 100, 2) as sell_through_rate
FROM MERCH m
JOIN ARTISTS a ON m.artist_id = a.artist_id
LEFT JOIN MERCH_SALES ms ON m.merch_id = ms.merch_id
GROUP BY a.artist_id, artist_name, m.merch_type, m.merch_status, m.merch_available_units
ORDER BY total_revenue DESC;
```

- **Event Performance and Attendance**

```
-- Event Success Metrics
SELECT
   e.event_name,
   e.event_type,
   e.event_date,
   e.event_location,
   e.event_status,
   COUNT(DISTINCT p.artist_id) as artist_count,
   COUNT(DISTINCT ea.fan_id) as attendance_count,
   COUNT(DISTINCT es.sale_id) as ticket_sales,
   SUM(es.event_sale_price) as ticket_revenue,
   ROUND(COUNT(DISTINCT ea.fan_id) / NULLIF(COUNT(DISTINCT es.sale_id), 0) * 100, 2)
as attendance_rate
FROM EVENTS e
LEFT JOIN PARTICIPATIONS p ON e.event_id = p.event_id
LEFT JOIN EVENT_ATTENDANCE ea ON e.event_id = ea.event_id AND ea.attendance_status
= 'confirmed'
LEFT JOIN EVENT_SALES es ON e.event_id = es.event_id
GROUP BY e.event_id, e.event_name, e.event_type, e.event_date, e.event_location,
e.event_status
ORDER BY e.event_date DESC;
```

- **Artist Development Timeline**

```
-- Artist Career Progression Analysis
SELECT
    a.artist_id,
    CONCAT(a.artist_f_name, ' ', a.artist_l_name) as artist_name,
    a.artist_debut,
    DATEDIFF(CURDATE(), a.artist_debut) as days_since_debut,
    COUNT(DISTINCT e.training_id) as trainings_completed,
    COUNT(DISTINCT c.content_id) as content_produced,
    COUNT(DISTINCT p.event_id) as events_participated,
    COUNT(DISTINCT f.fan_id) as fan_count,
    SUM(c.stream_count) as career_streams,
    SUM(c.revenue_generated) as career_revenue
FROM ARTISTS a
LEFT JOIN ENROLLMENTS e ON a.artist_id = e.artist_id
LEFT JOIN CONTENTS c ON a.artist_id = c.artist_id
LEFT JOIN PARTICIPATIONS p ON a.artist_id = p.artist_id
LEFT JOIN FANS f ON a.artist_id = f.artist_id
GROUP BY a.artist_id, artist_name, a.artist_debut
ORDER BY career_revenue DESC;
```

- **Sales Funnel Analysis**

```
-- Sales Performance by Type and Status
SELECT
    s.sale_type,
    s.sale_status,
    COUNT(s.sale_id) as transaction_count,
    SUM(s.sale_quantity) as total_units,
    SUM(s.sale_total_amount) as total_revenue,
    AVG(s.sale_total_amount) as avg_transaction_value,
    ROUND(SUM(CASE WHEN s.sale_status = 'completed' THEN s.sale_total_amount ELSE 0
END) / NULLIF(SUM(s.sale_total_amount), 0) * 100, 2) as completion_rate
FROM SALES s
GROUP BY s.sale_type, s.sale_status
ORDER BY s.sale_type, total_revenue DESC;
```

- **Content-Market Fit Analysis**

```
-- Best Performing Content Types by Market
SELECT
    m.market_region,
    c.content_type,
    COUNT(c.content_id) as content_count,
    AVG(c.stream_count) as avg_streams,
```

```
    AVG(c.revenue_generated) as avg_revenue,
    SUM(c.revenue_generated) as total_revenue,
    ROUND(AVG(c.revenue_generated) / NULLIF(AVG(c.production_cost), 0), 2) as roi_ratio
FROM CONTENTS c
JOIN MARKET_TARGETED_CONTENTS mtc ON c.content_id = mtc.content_id
JOIN MARKETS m ON mtc.market_id = m.market_id
GROUP BY m.market_region, c.content_type
HAVING content_count >= 1
ORDER BY m.market_region, roi_ratio DESC;
```

- **Training Completion and Impact**

```
-- Training Program Completion and Subsequent Performance
SELECT
    a.artist_id,
    CONCAT(a.artist_f_name, ' ', a.artist_l_name) as artist_name,
    t.training_name,
    e.enrollment_start_date,
    COUNT(DISTINCT c.content_id) as content_post_training,
    AVG(c.stream_count) as avg_streams_post_training,
    AVG(c.revenue_generated) as avg_revenue_post_training
FROM ARTISTS a
JOIN ENROLLMENTS e ON a.artist_id = e.artist_id
JOIN TRAININGS t ON e.training_id = t.training_id
LEFT JOIN CONTENTS c ON a.artist_id = c.artist_id AND c.content_release >
e.enrollment_start_date
GROUP BY a.artist_id, artist_name, t.training_name, e.enrollment_start_date
ORDER BY avg_revenue_post_training DESC;
```

- **Fan Loyalty and Purchase Behavior**

```
-- Fan Lifetime Value Analysis
SELECT
    f.fan_id,
    f.fan_region,
    f.fan_country,
    f.fan_engagement_score,
    DATEDIFF(CURDATE(), f.fan_join_date) as days_as_fan,
    COUNT(s.sale_id) as total_purchases,
    SUM(s.sale_total_amount) as total_spent,
    AVG(s.sale_total_amount) as avg_purchase_value,
    CASE
        WHEN COUNT(s.sale_id) >= 5 THEN 'VIP Customer'
        WHEN COUNT(s.sale_id) >= 2 THEN 'Regular Customer'
        ELSE 'One-time Customer'
    END as customer_segment
```

```
FROM FANS f
LEFT JOIN SALES s ON f.fan_id = s.fan_id AND s.sale_status = 'completed'
GROUP BY f.fan_id, f.fan_region, f.fan_country, f.fan_engagement_score, f.fan_join_date
ORDER BY total_spent DESC;
```

- **Event Type Performance**

```
-- Event Type Effectiveness
SELECT
   e.event_type,
   COUNT(DISTINCT e.event_id) as total_events,
   COUNT(DISTINCT p.artist_id) as total_artists,
   COUNT(DISTINCT ea.fan_id) as total_attendees,
   SUM(es.event_sale_price) as total_revenue,
   AVG(es.event_sale_price) as avg_ticket_price,
   ROUND(COUNT(DISTINCT ea.fan_id) / NULLIF(COUNT(DISTINCT e.event_id), 0), 0) as
avg_attendance_per_event
FROM EVENTS e
LEFT JOIN PARTICIPATIONS p ON e.event_id = p.event_id
LEFT JOIN EVENT_ATTENDANCE ea ON e.event_id = ea.event_id AND ea.attendance_status
= 'confirmed'
LEFT JOIN EVENT_SALES es ON e.event_id = es.event_id
WHERE e.event_status = 'completed'
GROUP BY e.event_type
ORDER BY total_revenue DESC;
```

- **Content Release Strategy**

```
-- Optimal Content Release Timing Analysis
SELECT
   MONTH(c.content_release) as release_month,
   DAYNAME(c.content_release) as release_day,
   c.content_type,
   COUNT(c.content_id) as content_count,
   AVG(c.stream_count) as avg_streams,
   AVG(c.revenue_generated) as avg_revenue,
   ROUND(AVG(c.revenue_generated) / NULLIF(AVG(c.production_cost), 0), 2) as avg_roi
FROM CONTENTS c
WHERE c.content_release IS NOT NULL
GROUP BY MONTH(c.content_release), DAYNAME(c.content_release), c.content_type
ORDER BY avg_roi DESC;
```

- **Analysis Network Value**

```
-- Artist Cross-Promotion Opportunities
SELECT
   a1.artist_id as artist1_id,
```

```
    CONCAT(a1.artist_f_name, ' ', a1.artist_l_name) as artist1_name,
    a2.artist_id as artist2_id,
    CONCAT(a2.artist_f_name, ' ', a2.artist_l_name) as artist2_name,
    COUNT(DISTINCT e1.event_id) as shared_events
FROM ARTISTS a1
JOIN PARTICIPATIONS p1 ON a1.artist_id = p1.artist_id
JOIN PARTICIPATIONS p2 ON p1.event_id = p2.event_id
JOIN ARTISTS a2 ON p2.artist_id = a2.artist_id AND a1.artist_id != a2.artist_id
JOIN EVENTS e1 ON p1.event_id = e1.event_id
GROUP BY a1.artist_id, artist1_name, a2.artist_id, artist2_name
HAVING shared_events >= 1
ORDER BY shared_events DESC;
```

- **Market Expansion Opportunities**

```
-- Untapped Market Potential
SELECT
    m.market_region,
    m.market_country,
    m.market_population,
    COUNT(DISTINCT mtc.content_id) as current_content,
    COUNT(DISTINCT f.fan_id) as current_fans,
    ROUND(COUNT(DISTINCT f.fan_id) / NULLIF(m.market_population, 0) * 1000000, 2) as
fans_per_million,
    CASE
        WHEN COUNT(DISTINCT mtc.content_id) = 0 THEN 'Untapped'
        WHEN COUNT(DISTINCT mtc.content_id) <= 3 THEN 'Underdeveloped'
        ELSE 'Developed'
    END as market_status
FROM MARKETS m
LEFT JOIN MARKET_TARGETED_CONTENTS mtc ON m.market_id = mtc.market_id
LEFT JOIN FANS f ON m.market_country = f.fan_country
GROUP BY m.market_region, m.market_country, m.market_population
ORDER BY market_population DESC;
```

- **Revenue Stream Analysis**

```
-- Revenue Breakdown by Source
SELECT
    'Content' as revenue_source,
    SUM(revenue_generated) as total_revenue,
    COUNT(content_id) as item_count,
    ROUND(SUM(revenue_generated) / NULLIF(COUNT(content_id), 0), 2) as
avg_revenue_per_item
FROM CONTENTS
WHERE revenue_generated > 0
```

```
UNION ALL

SELECT
    'Merchandise' as revenue_source,
    SUM(ms.merch_sale_quantity * ms.merch_sale_price) as total_revenue,
    COUNT(DISTINCT ms.merch_id) as item_count,
    ROUND(SUM(ms.merch_sale_quantity * ms.merch_sale_price) / NULLIF(COUNT(DISTINCT
ms.merch_id), 0), 2) as avg_revenue_per_item
FROM MERCH_SALES ms

UNION ALL

SELECT
    'Events' as revenue_source,
    SUM(es.event_sale_price) as total_revenue,
    COUNT(DISTINCT es.event_id) as item_count,
    ROUND(SUM(es.event_sale_price) / NULLIF(COUNT(DISTINCT es.event_id), 0), 2) as
avg_revenue_per_item
FROM EVENT_SALES es

ORDER BY total_revenue DESC;
```

- **Cost Efficiency Analysis**

```
-- Cost vs Revenue Efficiency
SELECT
    a.artist_id,
    CONCAT(a.artist_f_name, ' ', a.artist_l_name) as artist_name,
    SUM(c.production_cost) as total_production_cost,
    SUM(t.training_cost) as total_training_cost,
    SUM(c.production_cost + COALESCE(t.training_cost, 0)) as total_investment,
    SUM(c.revenue_generated) as total_revenue,
    (SUM(c.revenue_generated) - SUM(c.production_cost + COALESCE(t.training_cost, 0))) as
net_profit,
    ROUND(SUM(c.revenue_generated) / NULLIF(SUM(c.production_cost +
COALESCE(t.training_cost, 0)), 0), 2) as efficiency_ratio
FROM ARTISTS a
LEFT JOIN CONTENTS c ON a.artist_id = c.artist_id
LEFT JOIN ENROLLMENTS e ON a.artist_id = e.artist_id
LEFT JOIN TRAININGS t ON e.training_id = t.training_id
GROUP BY a.artist_id, artist_name
HAVING total_investment > 0
ORDER BY efficiency_ratio DESC;
```

- **Seasonal Performance Trends**

```
-- Quarterly Performance Analysis
SELECT
    CONCAT(YEAR(c.content_release), '-Q', QUARTER(c.content_release)) as quarter,
    c.content_type,
    COUNT(c.content_id) as content_count,
    SUM(c.stream_count) as total_streams,
    SUM(c.revenue_generated) as total_revenue,
    SUM(c.production_cost) as total_cost,
    (SUM(c.revenue_generated) - SUM(c.production_cost)) as net_profit,
    ROUND((SUM(c.revenue_generated) - SUM(c.production_cost)) /
NULLIF(SUM(c.production_cost), 0) * 100, 2) as profit_margin
FROM CONTENTS c
WHERE c.content_release IS NOT NULL
GROUP BY YEAR(c.content_release), QUARTER(c.content_release), c.content_type
ORDER BY quarter, total_revenue DESC;
```

- **Comprehensive Business Health Dashboard**

```
-- Executive Summary Dashboard
SELECT
    (SELECT COUNT(*) FROM ARTISTS WHERE artist_status = 'active') as active_artists,
    (SELECT COUNT(*) FROM CONTENTS) as total_content,
    (SELECT COUNT(*) FROM FANS) as total_fans,
    (SELECT SUM(revenue_generated) FROM CONTENTS) as content_revenue,
    (SELECT SUM(ms.merch_sale_quantity * ms.merch_sale_price) FROM MERCH_SALES
ms) as merch_revenue,
    (SELECT SUM(es.event_sale_price) FROM EVENT_SALES es) as event_revenue,
    (SELECT SUM(revenue_generated) FROM CONTENTS) +
    (SELECT SUM(ms.merch_sale_quantity * ms.merch_sale_price) FROM MERCH_SALES
ms) +
    (SELECT SUM(es.event_sale_price) FROM EVENT_SALES es) as total_revenue,
    (SELECT AVG(fan_engagement_score) FROM FANS) as avg_fan_engagement,
    (SELECT COUNT(*) FROM EVENTS WHERE event_status = 'completed') as
completed_events,
    (SELECT COUNT(*) FROM MARKETS) as active_markets;
```

These queries can provide comprehensive business intelligence across all aspects of Galaxy Entertainment. They demonstrate:

- Financial Analysis (ROI, revenue streams, cost efficiency)
- Operational Metrics (training effectiveness, event performance)
- Market Intelligence (regional performance, expansion opportunities)
- Fan Analytics (engagement, loyalty, segmentation)
- Strategic Insights (content performance, seasonal trends)

-