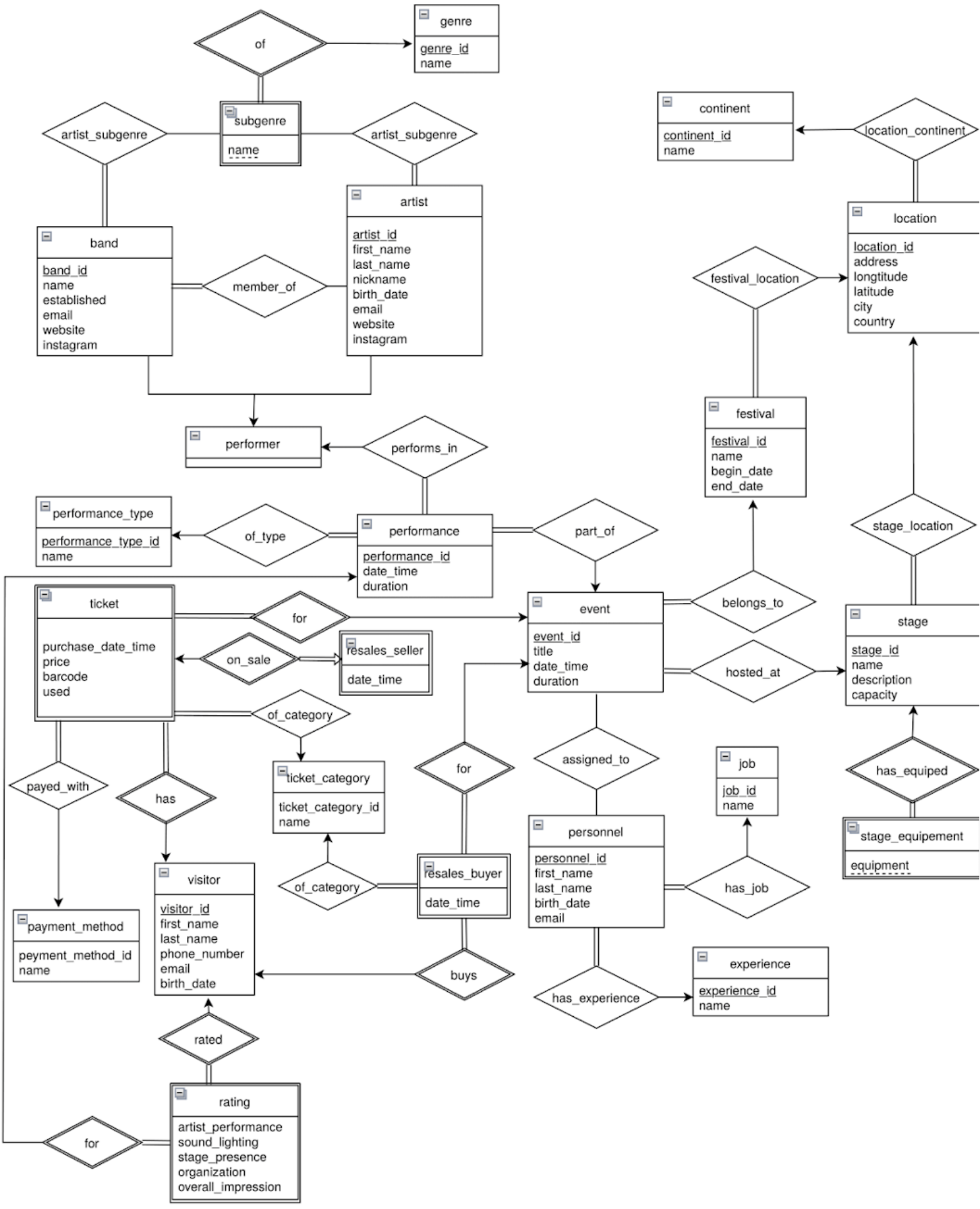
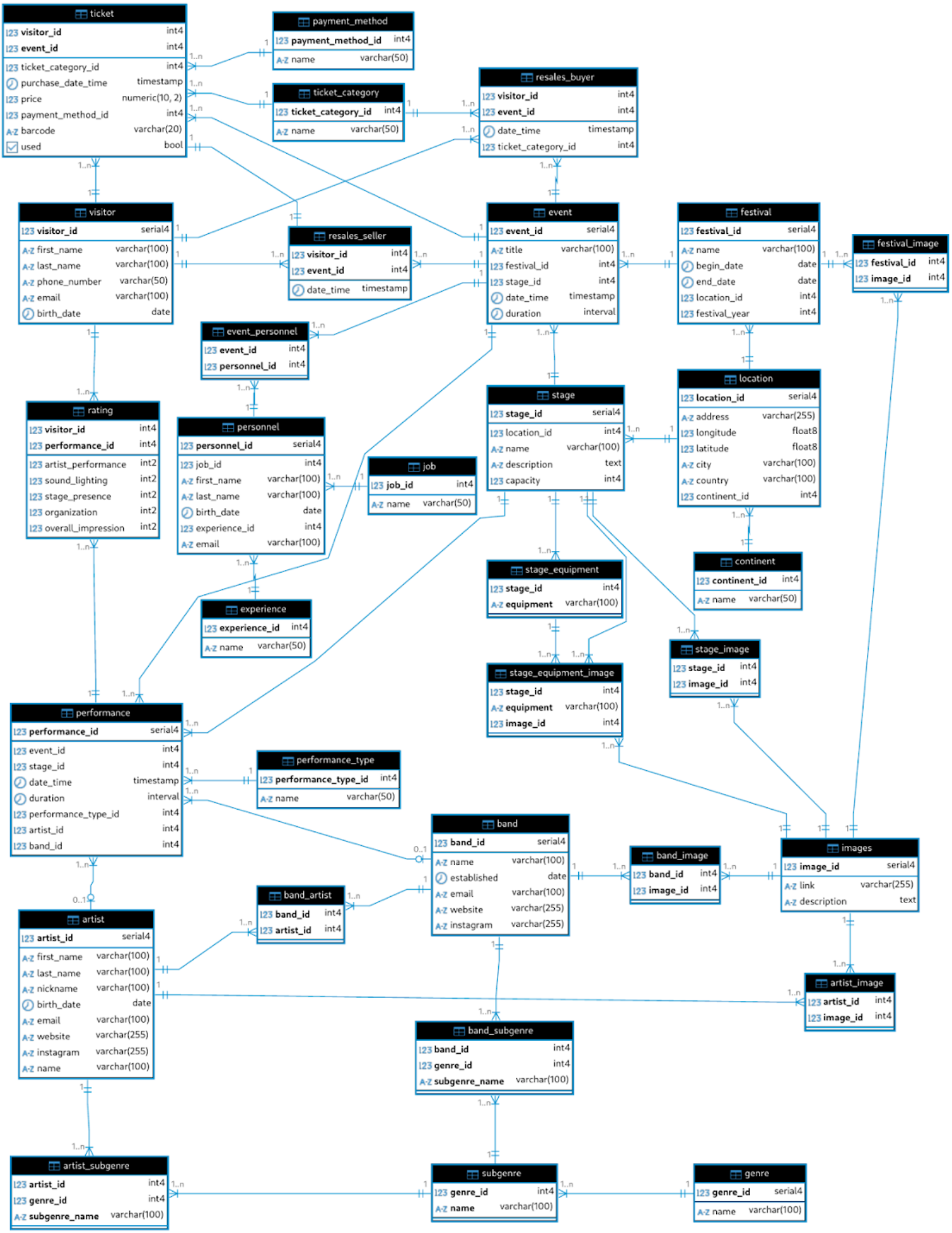

genre
genre_id
name
of
subgenre
name
artist_subgenre
artist_subgenre
band
band_id
name
established
email
website
instagram
member_of
artist
artist_id
first_name
last_name
nickname
birth_date
email
website
instagram
performer
performs_in
performance_type
performance_type_id
name
of_type
performance
performance_id
date_time
duration
part_of
ticket
purchase_date_time
price
barcode
used
for
on_sale
resales_seller
date_time
of_category
event
event_id
title
date_time
duration
belongs_to
hosted_at
continent
continent_id
name
location_continent
location
location_id
address
longtitude
latitude
city
country
festival_location
festival
festival_id
name
begin_date
end_date
stage_location
stage
stage_id
name
description
capacity
has_equiped
stage_equipement
equipment
payed_with
has
ticket_category
ticket_category_id
name
for
assigned_to
job
job_id
name
personnel
personnel_id
first_name
last_name
birth_date
email
has_job
visitor
visitor_id
first_name
last_name
phone_number
email
birth_date
of_category
resales_buyer
date_time
buys
payment_method
peyment_method_id
name
has_experience
experience
experience_id
name
rated
rating
artist_performance
sound_lighting
stage_presence
organization
overall_impression
for


ticket
visitor_id int4
event_id int4
ticket_category_id int4
purchase_date_time timestamp
price numeric(10, 2)
payment_method_id int4
barcode varchar(20)
used bool
payment_method
payment_method_id int4
name varchar(50)
ticket_category
ticket_category_id int4
name varchar(50)
resales_buyer
visitor_id int4
event_id int4
date_time timestamp
ticket_category_id int4
visitor
visitor_id serial4
first_name varchar(100)
last_name varchar(100)
phone_number varchar(50)
email varchar(100)
birth_date date
resales_seller
visitor_id int4
event_id int4
date_time timestamp
event
event_id serial4
title varchar(100)
festival_id int4
stage_id int4
date_time timestamp
duration interval
festival
festival_id serial4
name varchar(100)
begin_date date
end_date date
location_id int4
festival_year int4
festival_image
festival_id int4
image_id int4
event_personnel
event_id int4
personnel_id int4
rating
visitor_id int4
performance_id int4
artist_performance int2
sound_lighting int2
stage_presence int2
organization int2
overall_impression int2
personnel
personnel_id serial4
job_id int4
first_name varchar(100)
last_name varchar(100)
birth_date date
experience_id int4
email varchar(100)
job
job_id int4
name varchar(50)
stage
stage_id serial4
location_id int4
name varchar(100)
description text
capacity int4
location
location_id serial4
address varchar(255)
longitude float8
latitude float8
city varchar(100)
country varchar(100)
continent_id int4
continent
continent_id int4
name varchar(50)
stage_equipment
stage_id int4
equipment varchar(100)
stage_image
stage_id int4
image_id int4
stage_equipment_image
stage_id int4
equipment varchar(100)
image_id int4
experience
experience_id int4
name varchar(50)
performance
performance_id serial4
event_id int4
stage_id int4
date_time timestamp
duration interval
performance_type_id int4
artist_id int4
band_id int4
performance_type
performance_type_id int4
name varchar(50)
band
band_id serial4
name varchar(100)
established date
email varchar(100)
website varchar(255)
instagram varchar(255)
band_image
band_id int4
image_id int4
images
image_id serial4
link varchar(255)
description text
band_artist
band_id int4
artist_id int4
artist
artist_id serial4
first_name varchar(100)
last_name varchar(100)
nickname varchar(100)
birth_date date
email varchar(100)
website varchar(255)
instagram varchar(255)
name varchar(100)
artist_image
artist_id int4
image_id int4
band_subgenre
band_id int4
genre_id int4
subgenre_name varchar(100)
artist_subgenre
artist_id int4
genre_id int4
subgenre_name varchar(100)
subgenre
genre_id int4
name varchar(100)
genre
genre_id serial4
name varchar(100)

- Schema

```
9DROP TABLE IF EXISTS job CASCADE;
CREATE TABLE job (
      job_id INT PRIMARY KEY,
      name VARCHAR(50) NOT NULL
);

DROP TABLE IF EXISTS payment_method CASCADE;
CREATE TABLE payment_method (
      payment_method_id INT PRIMARY KEY,
      name VARCHAR(50) NOT NULL
);

DROP TABLE IF EXISTS ticket_category CASCADE;
CREATE TABLE ticket_category (
      ticket_category_id INT PRIMARY KEY,
      name VARCHAR(50) NOT NULL
);

DROP TABLE IF EXISTS experience CASCADE;
CREATE TABLE experience (
      experience_id INT PRIMARY KEY CHECK (experience_id BETWEEN 1 AND 5),
      name VARCHAR(50) NOT NULL
);

DROP TABLE IF EXISTS performance_type CASCADE;
CREATE TABLE performance_type (
      performance_type_id INT PRIMARY KEY,
      name VARCHAR(50) NOT NULL
);

DROP TABLE IF EXISTS continent CASCADE;
CREATE TABLE continent (
      continent_id INT PRIMARY KEY,
      name VARCHAR(50) NOT NULL
);

-- Data tables

DROP TABLE IF EXISTS location CASCADE;
CREATE TABLE location (
      location_id SERIAL PRIMARY KEY,
      address VARCHAR(255) NOT NULL,
      longitude DOUBLE PRECISION NOT NULL,
      latitude DOUBLE PRECISION NOT NULL,
      city VARCHAR(100) NOT NULL,
      country VARCHAR(100) NOT NULL,
      continent_id INT NOT NULL REFERENCES continent(continent_id) ON DELETE RESTRICT
);

DROP TABLE IF EXISTS stage CASCADE;
CREATE TABLE stage (
      stage_id SERIAL PRIMARY KEY,
      location_id INT NOT NULL REFERENCES location(location_id) ON DELETE CASCADE,
      name VARCHAR(100) NOT NULL,
      description TEXT,
      capacity INT NOT NULL CHECK (capacity >= 0)
);

DROP TABLE IF EXISTS festival CASCADE;
CREATE TABLE festival (
      festival_id SERIAL PRIMARY KEY,
      name VARCHAR(100) NOT NULL,
      begin_date DATE NOT NULL,
      end_date DATE NOT NULL,
      location_id INT NOT NULL REFERENCES location(location_id) ON DELETE CASCADE,

      festival_year INT GENERATED ALWAYS AS (EXTRACT(YEAR FROM begin_date)) STORED,
      CONSTRAINT same_begin_end_year CHECK (festival_year = EXTRACT(YEAR FROM end_date)),
      CONSTRAINT unique_festival_per_year UNIQUE (festival_year)
);

DROP TABLE IF EXISTS event CASCADE;
CREATE TABLE event (
      event_id SERIAL PRIMARY KEY,
      title VARCHAR(100) NOT NULL,
      festival_id INT NOT NULL REFERENCES festival(festival_id) ON DELETE RESTRICT,
      stage_id INT NOT NULL REFERENCES stage(stage_id) ON DELETE CASCADE,
      date_time TIMESTAMP NOT NULL,
      duration INTERVAL NOT NULL,


      CONSTRAINT stage_single_event_per_time EXCLUDE USING gist (
      stage_id WITH =,
      tsrange(date_time, date_time + duration, '[]') WITH &&
```

```
      )
);

DROP TABLE IF EXISTS stage_equipment CASCADE;
CREATE TABLE stage_equipment (
      stage_id INT NOT NULL REFERENCES stage(stage_id) ON DELETE CASCADE,
      equipment VARCHAR(100) NOT NULL,
      PRIMARY KEY (stage_id, equipment)
);

DROP TABLE IF EXISTS personnel CASCADE;
CREATE TABLE personnel (
      personnel_id SERIAL PRIMARY KEY,
      job_id INT NOT NULL REFERENCES job(job_id) ON DELETE RESTRICT,
      first_name VARCHAR(100) NOT NULL,
      last_name VARCHAR(100) NOT NULL,
      birth_date DATE NOT NULL,
      experience_id INT NOT NULL REFERENCES experience(experience_id) ON DELETE RESTRICT,
      email VARCHAR(100) NOT NULL UNIQUE CHECK (email ~'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,}$')
);

DROP TABLE IF EXISTS event_personnel CASCADE;
CREATE TABLE event_personnel (
      event_id INT NOT NULL REFERENCES event(event_id) ON DELETE RESTRICT,
      personnel_id INT NOT NULL REFERENCES personnel(personnel_id) ON DELETE CASCADE,
      PRIMARY KEY (event_id, personnel_id)
);

DROP TABLE IF EXISTS artist CASCADE;
CREATE TABLE artist (
      artist_id SERIAL PRIMARY KEY,
      first_name VARCHAR(100) NOT NULL,
      last_name VARCHAR(100) NOT NULL,
      nickname VARCHAR(100),
      birth_date DATE NOT NULL,
      email VARCHAR(100) NOT NULL UNIQUE CHECK (email ~'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,}$'),
      website VARCHAR(255),
      instagram VARCHAR(255),

      name VARCHAR(100) GENERATED ALWAYS AS (COALESCE(nickname, first_name || ' ' || last_name)) STORED
);

DROP TABLE IF EXISTS genre CASCADE;
CREATE TABLE genre (
      genre_id SERIAL PRIMARY KEY,
      name VARCHAR(100) NOT NULL
);

DROP TABLE IF EXISTS subgenre CASCADE;
CREATE TABLE subgenre (
      genre_id INT NOT NULL REFERENCES genre(genre_id) ON DELETE CASCADE,
      name VARCHAR(100) NOT NULL,
      PRIMARY KEY (genre_id, name)
);

DROP TABLE IF EXISTS artist_subgenre CASCADE;
CREATE TABLE artist_subgenre (
      artist_id INT NOT NULL REFERENCES artist(artist_id) ON DELETE CASCADE,
      genre_id INT NOT NULL,
      subgenre_name VARCHAR(100) NOT NULL,
      PRIMARY KEY (artist_id, genre_id, subgenre_name),
      FOREIGN KEY (genre_id, subgenre_name) REFERENCES subgenre(genre_id, name) ON DELETE CASCADE
);

DROP TABLE IF EXISTS band CASCADE;
CREATE TABLE band (
      band_id SERIAL PRIMARY KEY,
      name VARCHAR(100) NOT NULL,
      established DATE NOT NULL,
      email VARCHAR(100) NOT NULL UNIQUE CHECK (email ~'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,}$'),
      website VARCHAR(255),
      instagram VARCHAR(255)
);

DROP TABLE IF EXISTS band_artist CASCADE;
CREATE TABLE band_artist (
      band_id INT NOT NULL REFERENCES band(band_id) ON DELETE CASCADE,
      artist_id INT NOT NULL REFERENCES artist(artist_id) ON DELETE CASCADE,
      PRIMARY KEY (band_id, artist_id)
);

DROP TABLE IF EXISTS band_subgenre CASCADE;
CREATE TABLE band_subgenre (
      band_id INT NOT NULL REFERENCES band(band_id) ON DELETE CASCADE,
      genre_id INT NOT NULL,
      subgenre_name VARCHAR(100) NOT NULL,
      PRIMARY KEY (band_id, genre_id, subgenre_name),
```

```
      FOREIGN KEY (genre_id, subgenre_name) REFERENCES subgenre(genre_id, name) ON DELETE CASCADE
);

DROP TABLE IF EXISTS performance CASCADE;
CREATE TABLE performance (
      performance_id SERIAL PRIMARY KEY,
      event_id INT NOT NULL REFERENCES event(event_id) ON DELETE RESTRICT,
      stage_id INT NOT NULL REFERENCES stage(stage_id) ON DELETE CASCADE,
      date_time TIMESTAMP NOT NULL,
      duration INTERVAL NOT NULL CHECK (duration < '3 hours'),
      performance_type_id INT NOT NULL REFERENCES performance_type(performance_type_id) ON DELETE RESTRICT,
      artist_id INT REFERENCES artist(artist_id) ON DELETE CASCADE,
      band_id INT REFERENCES band(band_id) ON DELETE CASCADE CHECK (
      (artist_id IS NOT NULL AND band_id IS NULL) OR
      (artist_id IS NULL AND band_id IS NOT NULL)
      )
);

DROP TABLE IF EXISTS visitor CASCADE;
CREATE TABLE visitor (
      visitor_id SERIAL PRIMARY KEY,
      first_name VARCHAR(100) NOT NULL,
      last_name VARCHAR(100) NOT NULL,
      phone_number VARCHAR(50),
      email VARCHAR(100) NOT NULL UNIQUE CHECK (email ~'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,}$'),
      birth_date DATE NOT NULL
);

DROP TABLE IF EXISTS ticket CASCADE;
CREATE TABLE ticket (
      visitor_id INT NOT NULL REFERENCES visitor(visitor_id) ON DELETE CASCADE,
      event_id INT NOT NULL REFERENCES event(event_id) ON DELETE RESTRICT,
      ticket_category_id INT NOT NULL REFERENCES ticket_category(ticket_category_id) ON DELETE RESTRICT,
      purchase_date_time TIMESTAMP NOT NULL,
      price DECIMAL(10,2) NOT NULL,
      payment_method_id INT NOT NULL REFERENCES payment_method(payment_method_id) ON DELETE RESTRICT,
      barcode VARCHAR(20) NOT NULL,
      used BOOLEAN NOT NULL DEFAULT FALSE,
      PRIMARY KEY (visitor_id, event_id)
);

DROP TABLE IF EXISTS resales_buyer CASCADE;
CREATE TABLE resales_buyer (
      visitor_id INT NOT NULL REFERENCES visitor(visitor_id) ON DELETE CASCADE,
      event_id INT NOT NULL REFERENCES event(event_id) ON DELETE RESTRICT,
      date_time TIMESTAMP NOT NULL,
      ticket_category_id INT NOT NULL REFERENCES ticket_category(ticket_category_id) ON DELETE RESTRICT,
      PRIMARY KEY (visitor_id, event_id)
);

DROP TABLE IF EXISTS resales_seller CASCADE;
CREATE TABLE resales_seller (
      visitor_id INT NOT NULL REFERENCES visitor(visitor_id) ON DELETE CASCADE,
      event_id INT NOT NULL REFERENCES event(event_id) ON DELETE RESTRICT,
      date_time TIMESTAMP NOT NULL,
      PRIMARY KEY (visitor_id, event_id),
      FOREIGN KEY (visitor_id, event_id) REFERENCES ticket(visitor_id, event_id) ON DELETE CASCADE
);

DROP TABLE IF EXISTS rating CASCADE;
CREATE TABLE rating (
      visitor_id INT NOT NULL REFERENCES visitor(visitor_id) ON DELETE CASCADE,
      performance_id INT NOT NULL REFERENCES performance(performance_id) ON DELETE CASCADE,
      artist_performance SMALLINT NOT NULL CHECK (artist_performance BETWEEN 1 AND 5),
      sound_lighting SMALLINT NOT NULL CHECK (sound_lighting BETWEEN 1 AND 5),
      stage_presence SMALLINT NOT NULL CHECK (stage_presence BETWEEN 1 AND 5),
      organization SMALLINT NOT NULL CHECK (organization BETWEEN 1 AND 5),
      overall_impression SMALLINT NOT NULL CHECK (overall_impression BETWEEN 1 AND 5),
      PRIMARY KEY (visitor_id, performance_id)
);

DROP TABLE IF EXISTS images CASCADE;
CREATE TABLE images (
      image_id SERIAL PRIMARY KEY,
      link VARCHAR(255) NOT NULL,
      description TEXT
);

DROP TABLE IF EXISTS festival_image CASCADE;
CREATE TABLE festival_image (
      festival_id INT NOT NULL REFERENCES festival(festival_id) ON DELETE RESTRICT,
      image_id INT NOT NULL REFERENCES images(image_id) ON DELETE CASCADE,
      PRIMARY KEY (festival_id, image_id)
);

DROP TABLE IF EXISTS artist_image CASCADE;
CREATE TABLE artist_image (
```

```
      artist_id INT NOT NULL REFERENCES artist(artist_id) ON DELETE CASCADE,
      image_id INT NOT NULL REFERENCES images(image_id) ON DELETE CASCADE,
      PRIMARY KEY (artist_id, image_id)
);

DROP TABLE IF EXISTS band_image CASCADE;
CREATE TABLE band_image (
      band_id INT NOT NULL REFERENCES band(band_id) ON DELETE CASCADE,
      image_id INT NOT NULL REFERENCES images(image_id) ON DELETE CASCADE,
      PRIMARY KEY (band_id, image_id)
);

DROP TABLE IF EXISTS stage_image CASCADE;
CREATE TABLE stage_image (
      stage_id INT NOT NULL REFERENCES stage(stage_id) ON DELETE CASCADE,
      image_id INT NOT NULL REFERENCES images(image_id) ON DELETE CASCADE,
      PRIMARY KEY (stage_id, image_id)
);

DROP TABLE IF EXISTS stage_equipment_image CASCADE;
CREATE TABLE stage_equipment_image (
      stage_id INT NOT NULL REFERENCES stage(stage_id) ON DELETE CASCADE,
      equipment VARCHAR(100) NOT NULL,
      image_id INT NOT NULL REFERENCES images(image_id) ON DELETE CASCADE,
      PRIMARY KEY (stage_id, equipment, image_id),
      FOREIGN KEY (stage_id, equipment) REFERENCES stage_equipment(stage_id, equipment) ON DELETE CASCADE
);
```

- Q01

```
SELECT
      f.festival_year AS "Festival Year",
      SUM(t.price) AS "Total Income",
      ROUND((100.0 * SUM(t.price) FILTER (WHERE pm.name = 'Credit')) / SUM(t.price), 2)
      AS "Credit (%)",
      ROUND((100.0 * SUM(t.price) FILTER (WHERE pm.name = 'Debit')) / SUM(t.price), 2)
      AS "Debit (%)",
      ROUND((100.0 * SUM(t.price) FILTER (WHERE pm.name = 'Bank Transfer')) / SUM(t.price), 2)
      AS "Bank Transfer (%)"
FROM ticket t
JOIN event e USING (event_id)
JOIN festival f USING (festival_id)
LEFT JOIN payment_method pm USING (payment_method_id)
GROUP BY f.festival_year
ORDER BY f.festival_year;
```

- Q02

```
WITH vars AS (SELECT
      'Rock' AS target_genre,
      2023 AS target_year)
SELECT DISTINCT
      CASE WHEN p.artist_id IS NOT NULL THEN 'Artist' ELSE 'Band' END AS performer_type,
      COALESCE(p.artist_id, p.band_id) AS performer_id,
      COALESCE(ANY_VALUE(a.name), ANY_VALUE(b.name)) AS performer_name,
      CASE WHEN BOOL_OR(EXTRACT(YEAR FROM p.date_time) = vars.target_year) THEN 'Yes' ELSE 'No' END AS performed_on_target_year
FROM performance p
LEFT JOIN artist a USING (artist_id)
LEFT JOIN band b USING (band_id)
LEFT JOIN artist_subgenre asg USING (artist_id)
LEFT JOIN band_subgenre bsg USING (band_id)
JOIN genre g ON g.genre_id = COALESCE(asg.genre_id, bsg.genre_id)
JOIN vars ON vars.target_genre = g.name
GROUP BY p.artist_id, p.band_id
```

- Q03

```
WITH warm_up_performances AS (
      SELECT
      p.artist_id,
      p.band_id,
      f.festival_year AS festival_year,
      COUNT(*) AS warm_up_count
      FROM performance p
      JOIN event e USING (event_id)
      JOIN festival f USING (festival_id)
      JOIN performance_type USING (performance_type_id)
      WHERE performance_type.name = 'Warm Up'
      GROUP BY p.artist_id, p.band_id, f.festival_id
      HAVING COUNT(*) > 2
)

SELECT
      'Artist' AS performer_type,
      a.artist_id AS performer_id,
      a.name AS name,
      w.festival_year,
```

```
      w.warm_up_count
FROM warm_up_performances w
JOIN artist a ON w.artist_id = a.artist_id

UNION ALL

SELECT
      'Band' AS performer_type,
      b.band_id AS performer_id,
      b.name AS name,
      w.festival_year,
      w.warm_up_count
FROM warm_up_performances w
JOIN band b ON w.band_id = b.band_id

ORDER BY warm_up_count DESC;
```

- Q04

```
WITH vars AS (SELECT
      42 AS target_artist_id)
SELECT
      ANY_VALUE(artist.name) AS name,
      ROUND(AVG(artist_performance)::numeric, 2) AS average_artist_performance,
      ROUND(AVG(overall_impression)::numeric, 2) AS average_overal_impression
FROM performance
JOIN artist USING (artist_id)
JOIN rating USING (performance_id)
JOIN vars ON artist_id = target_artist_id
GROUP BY artist_id;
```

- Q05

```
WITH vars AS (SELECT
      42 AS target_artist_id)
SELECT
      ANY_VALUE(artist.name) AS name,
      ROUND(AVG(artist_performance)::numeric, 2) AS average_artist_performance,
      ROUND(AVG(overall_impression)::numeric, 2) AS average_overal_impression
FROM performance
JOIN artist USING (artist_id)
JOIN rating USING (performance_id)
JOIN vars ON artist_id = target_artist_id
GROUP BY artist_id;
```

- Q06

```
WITH vars AS (SELECT
      42 AS target_artist_id)
SELECT
      ANY_VALUE(artist.name) AS name,
      ROUND(AVG(artist_performance)::numeric, 2) AS average_artist_performance,
      ROUND(AVG(overall_impression)::numeric, 2) AS average_overal_impression
FROM performance
JOIN artist USING (artist_id)
JOIN rating USING (performance_id)
JOIN vars ON artist_id = target_artist_id
GROUP BY artist_id;
```

- Q07

```
WITH vars AS (SELECT
      42 AS target_artist_id)
SELECT
      ANY_VALUE(artist.name) AS name,
      ROUND(AVG(artist_performance)::numeric, 2) AS average_artist_performance,
      ROUND(AVG(overall_impression)::numeric, 2) AS average_overal_impression
FROM performance
JOIN artist USING (artist_id)
JOIN rating USING (performance_id)
JOIN vars ON artist_id = target_artist_id
GROUP BY artist_id;
```

- Q08

```
WITH vars AS (SELECT
      '2023-07-23'::DATE AS target_date)
SELECT personnel_id, first_name, last_name
FROM personnel
JOIN job USING (job_id)
WHERE
      personnel_id NOT IN (
      SELECT personnel_id
      FROM personnel
      JOIN event_personnel USING (personnel_id)
      JOIN event USING (event_id)
      JOIN vars ON event.date_time::DATE = target_date)
      AND job.name = 'Assistant';
```

- Q09

```
WITH vars AS (SELECT
      '2023-07-23'::DATE AS target_date)
SELECT personnel_id, first_name, last_name
FROM personnel
JOIN job USING (job_id)
WHERE
      personnel_id NOT IN (
      SELECT personnel_id
      FROM personnel
      JOIN event_personnel USING (personnel_id)
      JOIN event USING (event_id)
      JOIN vars ON event.date_time::DATE = target_date)
      AND job.name = 'Assistant';
```

- Q10

```
SELECT
      ANY_VALUE(g1.name) AS first_genre,
      ANY_VALUE(g2.name) AS second_genre,
      COUNT(*) AS total_occurences
FROM (
      SELECT
      COALESCE(bg1.genre_id, ag1.genre_id) AS genre_1_id,
      COALESCE(bg2.genre_id, ag2.genre_id) AS genre_2_id
      FROM performance p
      LEFT JOIN (SELECT DISTINCT band_id, genre_id FROM band_subgenre) AS bg1 USING (band_id)
      LEFT JOIN (SELECT DISTINCT band_id, genre_id FROM band_subgenre) AS bg2 USING (band_id)
      LEFT JOIN (SELECT DISTINCT artist_id, genre_id FROM artist_subgenre) AS ag1 USING (artist_id)
      LEFT JOIN (SELECT DISTINCT artist_id, genre_id FROM artist_subgenre) AS ag2 USING (artist_id))
JOIN genre g1 ON genre_1_id = g1.genre_id
JOIN genre g2 ON genre_2_id = g2.genre_id
WHERE genre_1_id < genre_2_id
GROUP BY (g1.genre_id, g2.genre_id)
ORDER BY total_occurences DESC
LIMIT 3;
```

- Q11

```
SELECT
      ANY_VALUE(g1.name) AS first_genre,
      ANY_VALUE(g2.name) AS second_genre,
      COUNT(*) AS total_occurences
FROM (
      SELECT
      COALESCE(bg1.genre_id, ag1.genre_id) AS genre_1_id,
      COALESCE(bg2.genre_id, ag2.genre_id) AS genre_2_id
      FROM performance p
      LEFT JOIN (SELECT DISTINCT band_id, genre_id FROM band_subgenre) AS bg1 USING (band_id)
      LEFT JOIN (SELECT DISTINCT band_id, genre_id FROM band_subgenre) AS bg2 USING (band_id)
      LEFT JOIN (SELECT DISTINCT artist_id, genre_id FROM artist_subgenre) AS ag1 USING (artist_id)
      LEFT JOIN (SELECT DISTINCT artist_id, genre_id FROM artist_subgenre) AS ag2 USING (artist_id))
JOIN genre g1 ON genre_1_id = g1.genre_id
JOIN genre g2 ON genre_2_id = g2.genre_id
WHERE genre_1_id < genre_2_id
GROUP BY (g1.genre_id, g2.genre_id)
ORDER BY total_occurences DESC
LIMIT 3;
```

- Q12

```
SELECT
      event.date_time::DATE as day,
      CEIL(SUM(capacity) * 0.02) AS assistant,
      CEIL(SUM(capacity) * 0.05) AS security,
      CEIL(SUM(capacity) * 0.02) + CEIL(SUM(capacity) * 0.05) AS total_personnel
FROM event
JOIN stage USING (stage_id)
GROUP BY event.date_time::DATE
ORDER BY day DESC
```

- Q13

```
SELECT
      event.date_time::DATE as day,
      CEIL(SUM(capacity) * 0.02) AS assistant,
      CEIL(SUM(capacity) * 0.05) AS security,
      CEIL(SUM(capacity) * 0.02) + CEIL(SUM(capacity) * 0.05) AS total_personnel
FROM event
JOIN stage USING (stage_id)
GROUP BY event.date_time::DATE
ORDER BY day DESC
```

- Q14

```
SELECT
```

```
      event.date_time::DATE as day,
      CEIL(SUM(capacity) * 0.02) AS assistant,
      CEIL(SUM(capacity) * 0.05) AS security,
      CEIL(SUM(capacity) * 0.02) + CEIL(SUM(capacity) * 0.05) AS total_personnel
FROM event
JOIN stage USING (stage_id)
GROUP BY event.date_time::DATE
ORDER BY day DESC
```

- Q15

```
SELECT
      event.date_time::DATE as day,
      CEIL(SUM(capacity) * 0.02) AS assistant,
      CEIL(SUM(capacity) * 0.05) AS security,
      CEIL(SUM(capacity) * 0.02) + CEIL(SUM(capacity) * 0.05) AS total_personnel
FROM event
JOIN stage USING (stage_id)
GROUP BY event.date_time::DATE
ORDER BY day DESC
```