ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

«Υλοποίηση βάσης δεδομένων για το διεθνές φεστιβάλ μουσικής, Pulse University»

Εξαμηνιαία Εργασία

στο μάθημα **«Βάσεις δεδομένων»**

των φοιτητών

Νικόλαου Σμυρνάκη, Α.Μ.: 03122428
Μαρίνας Φραγκούλη, Α.Μ.: 03122429

Διδάσκοντες: Δ. Τσουμάκος, Μ. Κόνιαρης

Αθήνα, Μάιος 2025

Πίνακας περιεχομένων

1. Σχεδιασμός και υλοποίηση

a. E-R διάγραμμα

Με υπογράμμιση φαίνονται τα Primary Keys και με (FK) τα Foreign Keys (σύμφωνα με το βιβλίο τα FK παραλείπονται λόγω της ύπαρξης των ρόμβων όμως με αυτόν τον τρόπο ήταν πιο εύκολη η μετάβαση από το ER στην SQL). Σύμφωνα με την εκφώνηση κάθε μια από τις υπογραμμισμένες λέξεις είναι ένας πίνακας. Οι πίνακες της εκφώνησης είναι οι μπλε πίνακες ενώ πορτοκαλί είναι οι πίνακες που προσθέσαμε εμείς κατά παραδοχή. Το ER μας έχει ως weak entities τα role_of_personel_on _event, τα reviews, τα group_members, τους buyers και τους sellers.

Table of contents

1 artist

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
artist_ID	int(11)		No		auto_increment			
artist_name	varchar(255)		No					
stage_name	varchar(255)		Yes	NULL				
artist_date_of_birth	date		No					
artist_debute	date		No					
artist_website	varchar(255)		Yes	NULL				
artist_instagram	varchar(255)		Yes	NULL				
num_of_consecutive_years_participating	int(11)		Yes	0				

2 building

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
building_ID	int(11)		No		auto_increment			
building_name	varchar(255)		No					
building_description	text		No					
max_capacity	int(11)		No					

3 buyer

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
buyer_ID	int(11)		No			-> visitor.visitor_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

4 events

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
event_ID	int(11)		No		auto_increment			
festival_ID	int(11)		Yes	NULL		-> festival.festival_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
event_name	varchar(255)		No					
festival_day	int(11)		No					
event_start_time	datetime		No					
event_end_time	datetime		No					
building_ID	int(11)		Yes	NULL		-> building.building_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
event_duration	int(11)		Yes	NULL	STORED GENERATED			
VIP_total	int(11)		Yes	NULL				
backstage_total	int(11)		Yes	NULL				
general_total	int(11)		Yes	NULL				

5 festival

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
festival_ID	int(11)		No		auto_increment			
starting_date	date		No					
duration	int(11)		No					

6 festival_location

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
festival_ID	int(11)		Yes	NULL		-> festival.festival_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
festival_location_ID	int(11)		No		auto_increment			
address	varchar(255)		No					
town	varchar(100)		No					
country	varchar(100)		No					
continent	varchar(100)		No					
geo_coordinates	varchar(100)		No					

7 genre

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
genre_ID	int(11)		No		auto_increment			
genre_name	varchar(100)		No					
subgenre_name	varchar(100)		Yes	NULL				
artist_ID	int(11)		Yes	NULL		-> artist.artist_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
group_ID	int(11)		Yes	NULL		-> group.group_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

8 group

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
group_ID	int(11)		No		auto_increment			
group_name	varchar(255)		No					
group_date_of_birth	date		No					
group_debute	date		No					
group_website	varchar(255)		Yes	NULL				
group_instagram	varchar(255)		Yes	NULL				
member_names	text		Yes	''				
num_of_consecutive_years_participating	int(11)		Yes	0				

9 group_members

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
group_ID	int(11)		No			-> group.group_ID ON UPDATE RESTRICT ON DELETE CASCADE		
artist_ID	int(11)		No			-> artist.artist_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

10 performances

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
performance_ID	int(11)		No		auto_increment			
event_ID	int(11)		Yes	NULL		-> events.event_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
performance_type	enum('warm up', 'headline', 'special_guest', 'finale')		No					
performance_start_time	datetime		No					
performance_end_time	datetime		No					
performance_duration	int(11)		Yes	NULL	STORED GENERATED			
building_ID	int(11)		No			-> building.building_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
building_name	varchar(255)		No					
artist_ID	int(11)		Yes	NULL		-> artist.artist_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
group_ID	int(11)		Yes	NULL		-> group.group_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

11 personel

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
personel_ID	int(11)		No		auto_increment			
first_name	varchar(100)		No					
last_name	varchar(100)		No					
age	int(11)		No					
email	varchar(255)		No					
phone_number	varchar(20)		No					
expertise_status	enum('intern', 'beginer', 'intermidiate', 'experienced', 'very_experienced')		Yes	NULL				

12 photo

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
photo_ID	int(11)		No		auto_increment			
photo_name	varchar(255)		No					
photo_description	text		No					
artist_ID	int(11)		Yes	NULL		-> artist.artist_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
group_ID	int(11)		Yes	NULL		-> group.group_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
performance_ID	int(11)		Yes	NULL		-> performances.performance_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
event_ID	int(11)		Yes	NULL		-> events.event_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
festival_ID	int(11)		Yes	NULL		-> festival.festival_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
technical_equipment_ID	int(11)		Yes	NULL		-> technical_equipment.technical_equipment_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

13 resale_queue

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
resale_ID	int(11)		No		auto_increment			
buyer_ID	int(11)		Yes	NULL		-> visitor.visitor_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
seller_ID	int(11)		Yes	NULL		-> visitor.visitor_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
event_name	varchar(255)		Yes	NULL				
ticket_type	enum('general_admission', 'VIP', 'backstage')		Yes	NULL				
ticket_ID	int(11)		Yes	NULL		-> ticket.ticket_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
listed_at	timestamp		No	current_timestamp()	on update current_timestamp()			

14 review

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
ticket_ID	int(11)		No			-> ticket.ticket_ID ON UPDATE RESTRICT ON DELETE CASCADE		
performance_ID	int(11)		No			-> performances.performance_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
artist_performance	enum('1', '2', '3', '4', '5')		Yes	NULL				
sound_and_lighting	enum('1', '2', '3', '4', '5')		Yes	NULL				
stage_presence	enum('1', '2', '3', '4', '5')		Yes	NULL				
event_organization	enum('1', '2', '3', '4', '5')		Yes	NULL				
overall_impression	enum('1', '2', '3', '4', '5')		Yes	NULL				

15 role_of_personel_on_event

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
personel_ID	int(11)		No			-> personel.personel_ID ON UPDATE RESTRICT ON DELETE CASCADE		
event_ID	int(11)		No			-> events.event_ID ON UPDATE RESTRICT ON DELETE CASCADE		
role	enum('technical', 'security', 'support')		No					

16 seller

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
seller_ID	int(11)		No			-> visitor.visitor_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

17 technical_equipment

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
technical_equipment_ID	int(11)		No		auto_increment			
building_ID	int(11)		Yes	NULL		-> building.building_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
equipment_name	varchar(255)		No					
equipment_description	text		No					

18 temp_resale_matches

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
match_ID	int(11)		No		auto_increment			
buyer_ID	int(11)		Yes	NULL		-> buyer.buyer_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
seller_ID	int(11)		Yes	NULL		-> seller.seller_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
ticket_ID	int(11)		Yes	NULL		-> ticket.ticket_ID ON UPDATE RESTRICT ON DELETE RESTRICT		

19 ticket

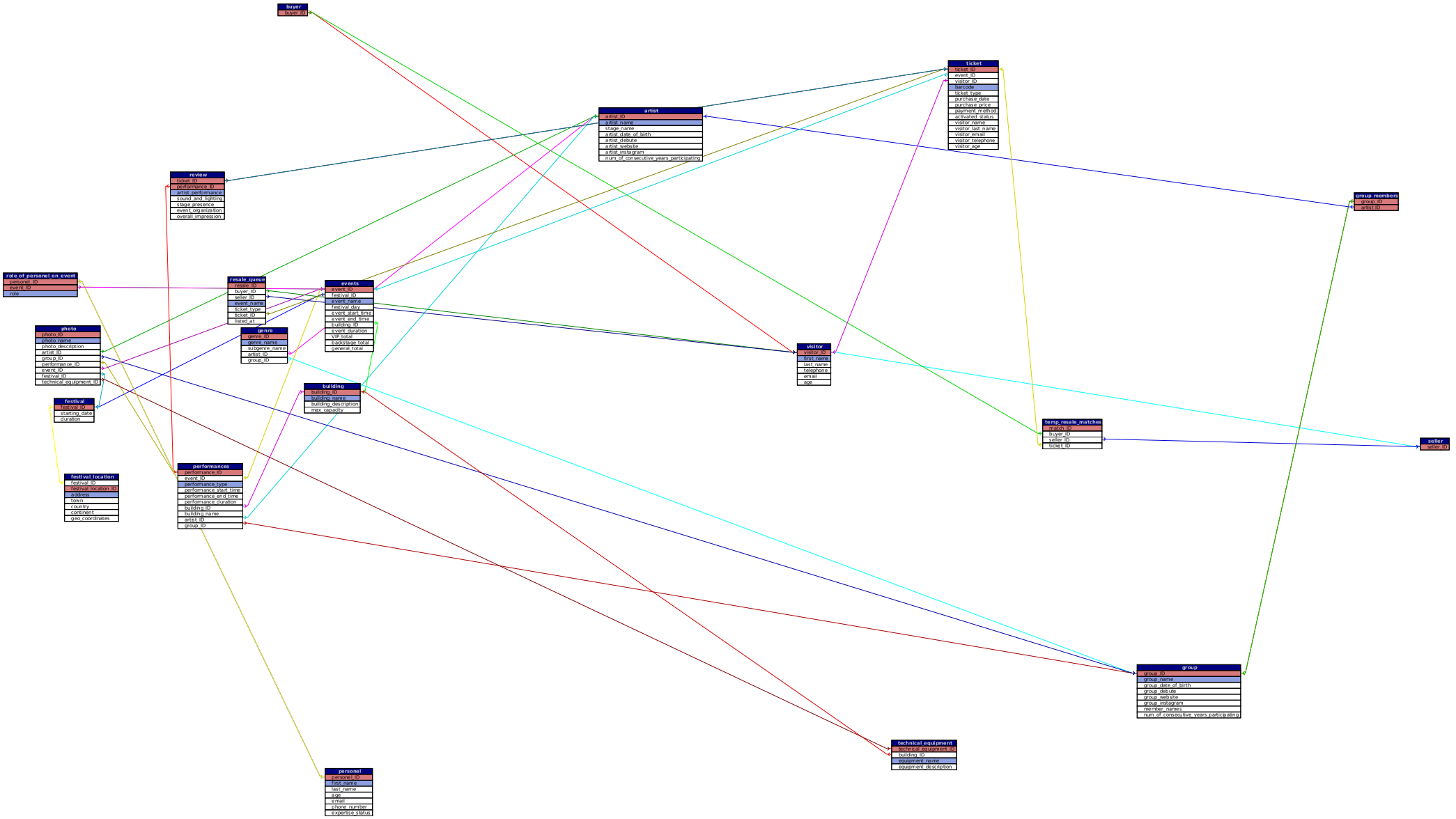
Creation: May 11, 2025 at 02:40 PM

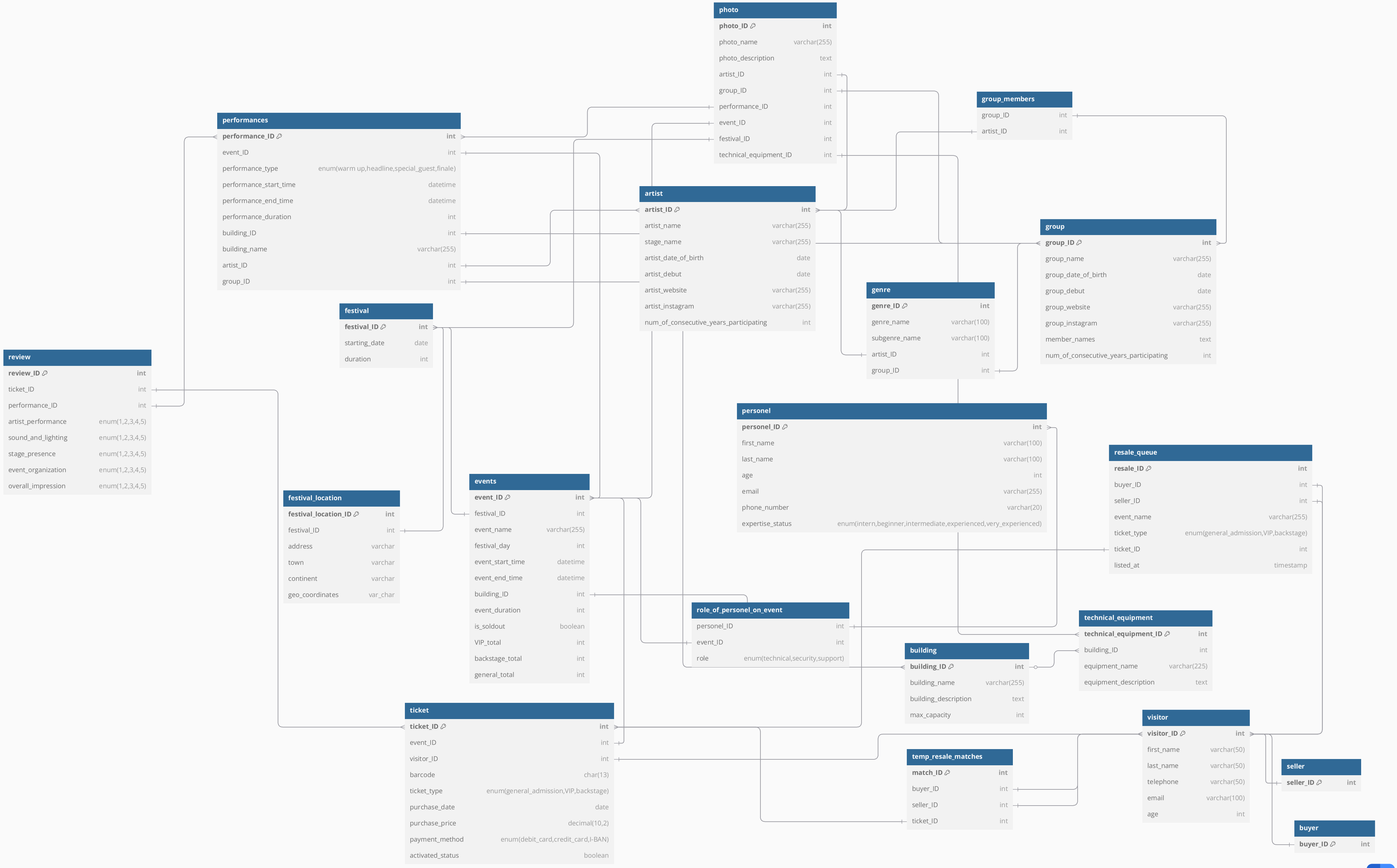
Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
ticket_ID	int(11)		No		auto_increment			
event_ID	int(11)		Yes	NULL		-> events.event_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
visitor_ID	int(11)		No			-> visitor.visitor_ID ON UPDATE RESTRICT ON DELETE RESTRICT		
barcode	char(13)		Yes	NULL				
ticket_type	enum('general_admission', 'VIP', 'backstage')		No					
purchase_date	date		Yes	NULL				
purchase_price	decimal(10, 2)		Yes	NULL				
payment_method	enum('debit_card', 'credit_card', 'I-BAN')		Yes	NULL				
activated_status	tinyint(1)		Yes	0				
visitor_name	varchar(100)		Yes	NULL				
visitor_last_name	varchar(100)		Yes	NULL				
visitor_email	varchar(100)		Yes	NULL				
visitor_telephone	varchar(20)		Yes	NULL				
visitor_age	int(11)		Yes	NULL				

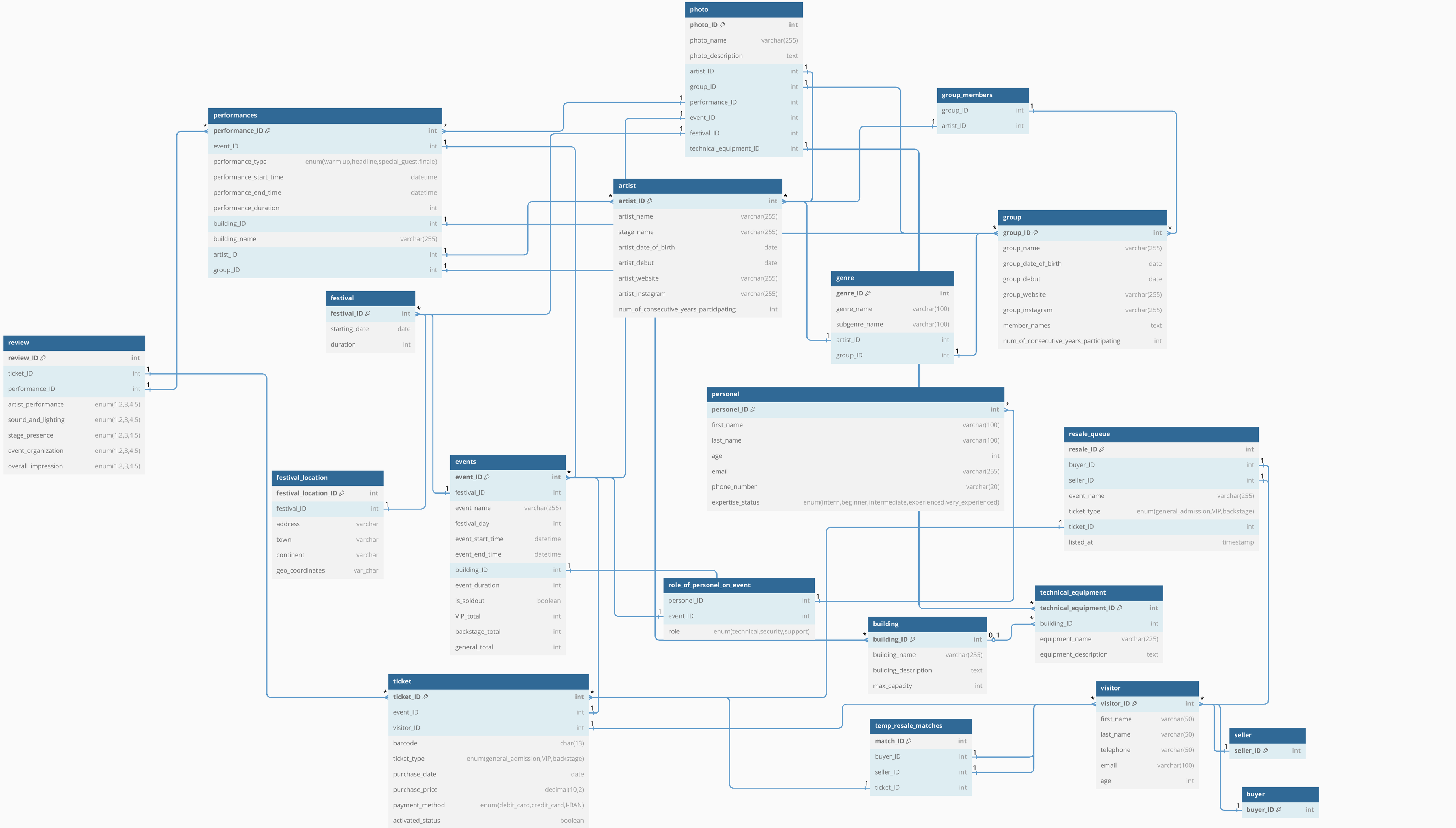
20 visitor

Creation: May 11, 2025 at 02:40 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
visitor_ID	int(11)		No		auto_increment			
first_name	varchar(50)		No					
last_name	varchar(50)		No					
telephone	varchar(50)		No					
email	varchar(100)		No					
age	int(11)		No					


buyer
buyer_ID
review
ticket_ID
performance_ID
artist_performance
sound_and_lighting
stage_presence
event_organization
overall_impression
artist
artist_ID
stage_name
artist_date_of_birth
artist_debute
artist_website
artist_instagram
num_of_consecutive_years_participating
ticket
ticket_ID
event_ID
visitor_ID
barcode
ticket_type
purchase_date
purchase_price
payment_method
activated_status
visitor_name
visitor_last_name
visitor_email
visitor_telephone
visitor_age
group_members
group_ID
artist_ID
role_of_personel_on_event
personel_ID
event_ID
role
resale_queue
resale_ID
buyer_ID
seller_ID
event_name
ticket_type
ticket_ID
listed_at
events
event_ID
festival_ID
event_name
festival_day
event_start_time
event_end_time
building_ID
event_duration
VIP_total
backstage_total
general_total
photo
photo_ID
photo_name
photo_description
artist_ID
group_ID
performance_ID
event_ID
festival_ID
technical_equipment_ID
genre
genre_ID
genre_name
subgenre_name
artist_ID
group_ID
visitor
visitor_ID
first_name
last_name
telephone
email
age
festival
festival_ID
starting_date
duration
building
building_ID
building_name
building_description
max_capacity
temp_resale_matches
match_ID
buyer_ID
seller_ID
ticket_ID
seller
seller_ID
festival_location
festival_ID
festival_location_ID
address
town
country
continent
geo_coordinates
performances
performance_ID
event_ID
performance_type
performance_start_time
performance_end_time
performance_duration
building_ID
building_name
artist_ID
group_ID
group
group_ID
group_name
group_date_of_birth
group_debute
group_website
group_instagram
member_names
num_of_consecutive_years_participating
technical_equipment
technical_equipment_ID
building_ID
equipment_name
equipment_description
personel
personel_ID
first_name
last_name
age
email
phone_number
expertise_status





b. Σχεσιακό διάγραμμα

Εδώ υπάρχει κείμενο

i. Ορισμός απαραίτητων περιορισμών

Εκφώνηση

Να ορίσετε όλους τους απαραίτητους περιορισμούς που θα εξασφαλίζουν την ορθότητα της ΒΔ. Αυτοί είναι περιορισμοί ακεραιότητας, κλειδιά, αναφορική ακεραιότητα, ακεραιότητα πεδίου τιμών και περιορισμοί οριζόμενοι από τον χρήστη.

Απάντηση

Γενικότερα για να εξασφαλιστεί η ορθότητα της βάσης δεδομένων (ΒΔ), πρέπει να οριστούν οι παρακάτω περιορισμοί (constraints):

2. Περιορισμοί ακεραιότητας (PK, FK, UNIQUE, NOT NULL)
3. Αναφορική ακεραιότητα(Referential Integrity) (FK, ON DELETE CASCADE / ON UPDATE CASCADE)
4. Ακεραιότητα Πεδίου Τιμών (Domain Integrity) (CHECK, ENUM)
5. Περιορισμοί από τον Χρήστη.

Εμείς χρησιμοποιήσαμε Triggers, Constrains και Cascades.

TRIGGERS

Deletion Triggers

- prevent_festival_deletion (2)
- prevent_performance_deletion (2)

Ticket Triggers

- check_ticket_availability --- Check if the ticket can be sold based on the event's capacity and ticket type limits (3)
- fill_ticket_visitor_data --- When a new ticket is created, fill in visitor data from the visitor table

Resale Triggers

- check_ticket_activation_before_resale (3)
- match_resale_after_insert --Matching Seller and Buyer and Updating Ticket
- create_buyer_or_seller_after_visitor (4)
- trg_check_soldout_before_resale (4)

Event Triggers

- check_festival_day -- Ensure that the festival_day is within the festival duration (4)

Genre Triggers

- check_genre_entity_exclusivity -- Ensure that each genre is linked to either one artist or one group, but not both or neither (3)

Performance Triggers

- check_performance_overlap -- Ensure a minimum 5-minute break between performances of the same event in the same building
- trg_check_consecutive_years_artists (4)
- trg_check_consecutive_years_groups (4)
- prevent_artist_group_overlap (on Insert) (4)
- prevent_artist_group_overlap_update (on Update) (4)

Ticket Triggers

- check_vip_limit (4)
- prevent_duplicate_ticket-- Prevent duplicate tickets for the same visitor and event

Group Triggers

- group_member_names

Review Triggers

- check_ticket_activation -- Ensure that a review can only be created if the ticket is activated
- check_review_validity -- Ensure that the performance belongs to the same event as the ticket and ticket is activated

Role of personnel on event Triggers

- check_personel_availability -- Ensure that the same personel cannot have multiple roles in the same event

CONSTRAINS

Resale Constrains

chk_seller_or_buyer

chk_one_side_only

CASCADES

after deletion on personel delete role_of_personel_on event as well

after deletion on ticket delete review as well

after deletion on event delete role_of_perdonel_on_event as well

after deletion on group delete group_members as well

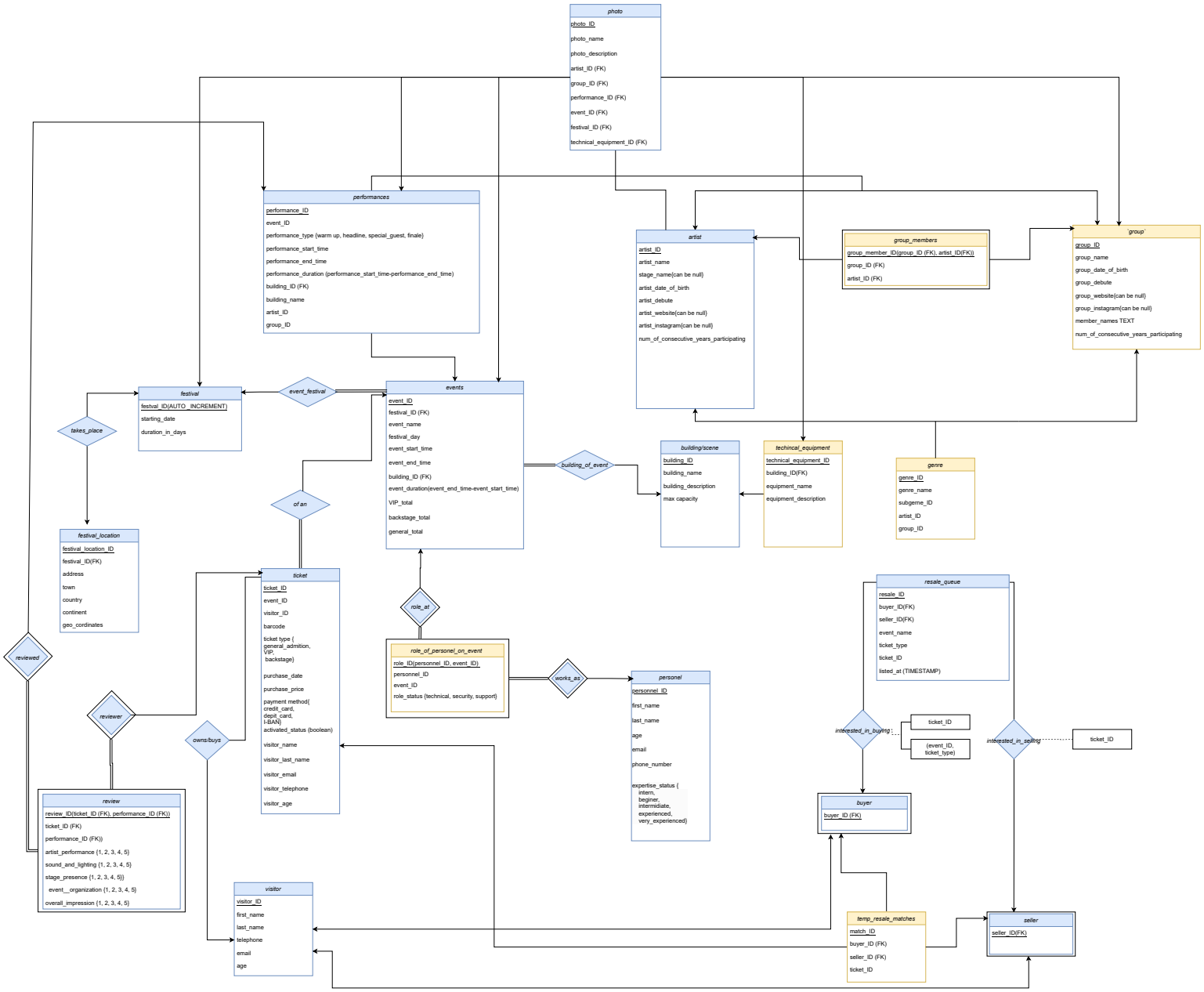


photo
photo_ID
photo_name
photo_description
artist_ID (FK)
group_ID (FK)
performance_ID (FK)
event_ID (FK)
festival_ID (FK)
technical_equipment_ID (FK)
performances
performance_ID
event_ID
performance_type (warm up, headline, special_guest, finale)
performance_start_time
performance_end_time
performance_duration (performance_start_time-performance_end_time)
building_ID (FK)
building_name
artist_ID
group_ID
artist
artist_ID
artist_name
stage_name(can be null)
artist_date_of_birth
artist_debute
artist_website(can be null)
artist_instagram(can be null)
num_of_consecutive_years_participating
group_members
group_member_ID(group_ID (FK), artist_ID(FK))
group_ID (FK)
artist_ID (FK)
'group'
group_ID
group_name
group_date_of_birth
group_debute
group_website(can be null)
group_instagram(can be null)
member_names TEXT
num_of_consecutive_years_participating
festival
festival_ID(AUTO_INCREMENT)
starting_date
duration_in_days
event_festival
events
event_ID
festival_ID (FK)
event_name
festival_day
event_start_time
event_end_time
building_ID (FK)
event_duration(event_end_time-event_start_time)
VIP_total
backstage_total
general_total
takes_place
building_of_event
building/scene
building_ID
building_name
building_description
max capacity
techincal_equipment
technical_equipment_ID
building_ID(FK)
equipment_name
equipment_description
genre
genre_ID
genre_name
subgerne_ID
artist_ID
group_ID
of an
festival_location
festival_location_ID
festival_ID(FK)
address
town
country
continent
geo_cordinates
ticket
ticket_ID
event_ID
visitor_ID
barcode
ticket type (general_admition, VIP, backstage)
purchase_date
purchase_price
payment method(credit_card, depit_card, I-BAN)
activated_status (boolean)
visitor_name
visitor_last_name
visitor_email
visitor_telephone
visitor_age
role_at
role_of_personel_on_event
role_ID(personnel_ID, event_ID),
personnel_ID
event_ID
role_status (technical, security, support)
works_as
personel
personnel_ID
first_name
last_name
age
email
phone_number
expertise_status (intern, beginer, intermidiate, experienced, very_experienced)
resale_queue
resale_ID
buyer_ID(FK)
seller_ID(FK)
event_name
ticket_type
ticket_ID
listed_at (TIMESTAMP)
interested_in_buying
ticket_ID
(event_ID, ticket_type)
interested_in_selling
ticket_ID
buyer
buyer_ID (FK)
seller
seller_ID(FK)
temp_resale_matches
match_ID
buyer_ID (FK)
seller_ID (FK)
ticket_ID
reviewed
reviewer
owns/buys
review
review_ID(ticket_ID (FK), performance_ID (FK))
ticket_ID (FK)
performance_ID (FK))
artist_performance {1, 2, 3, 4, 5)
sound_and_lighting {1, 2, 3, 4, 5)
stage_presence {1, 2, 3, 4, 5))
event__organization (1, 2, 3, 4, 5)
overall_impression {1, 2, 3, 4, 5)
visitor
visitor_ID
first_name
last_name
telephone
email
age

i. Ευρετήρια

Εκφώνηση

Να ορίσετε κατάλληλα ευρετήρια (indexes) για τους πίνακες της ΒΔ και να δικαιολογήσετε την επιλογή σας με βάση την χρησιμότητα τους για τα ερωτήματα στα οποία χρησιμοποιούνται.

Απάντηση

```
-- == INDEXES == --

CREATE INDEX idx_perf_event_artist ON performances(event_ID,
artist_ID); Q04, Q05, Q09, Q10, Q11, Q13, Q14, Q15
CREATE INDEX idx_artist_name ON artist(artist_name); Q04
CREATE INDEX idx_perf_artist_event ON performances(artist_ID,
event_ID); Q02, Q04, Q05, Q10,Q11, Q13, Q15
CREATE INDEX idx_perf_group_event ON performances(group_ID,
event_ID); Q02, Q04, Q05, Q10,Q11, Q13, Q15

CREATE INDEX idx_events_festival ON events(festival_ID); Q05, Q10,
Q11, Q13
CREATE INDEX idx_ticket_visitor_event ON ticket(visitor_ID,
event_ID); Q06, Q09, Q15
CREATE INDEX idx_ticket_event ON ticket(event_ID); Q09,Q15
CREATE INDEX idx_review_ticket ON review(ticket_ID); Q04, Q06, Q15
CREATE INDEX idx_genre_artist ON genre(artist_ID); Q02, Q10, Q14
CREATE INDEX idx_genre_group ON genre(group_ID); Q02, Q10, Q14
CREATE INDEX idx_role_event_role ON
role_of_personel_on_event(event_ID, role); Q12
CREATE INDEX idx_visitor_full_name ON visitor(last_name,
first_name); Q06
CREATE INDEX idx_ticket_purchase_year_price ON
ticket(purchase_date, purchase_price); Q01
CREATE INDEX idx_perf_type_artist_event ON
performances(performance_type, artist_ID, event_ID); Q03
CREATE INDEX idx_role_event_role_personel ON
role_of_personel_on_event(role, personel_ID, event_ID); Q07, Q08
```

```
CREATE INDEX idx_personel_expertise ON personel(expertise_status);
Q08
CREATE INDEX idx_festival_location_continent ON
festival_location(festival_ID, continent); Q13
```

6.

Queries

Εκφώνηση

Να σχεδιάσετε και εκτελέσετε τα ακόλουθα ερωτήματα:

(τα ερωτήματα είναι ισόβαθμα). Κάθε ερώτημα θα πρέπει να υλοποιείται με ένα query και να επιστρέφει ένα σύνολο αποτελεσμάτων. Όλα τα ερωτήματα πρέπει να επιστρέφουν έγκυρα αποτελέσματα, διαφορετικά δεν θα βαθμολογούνται.

** Για τα ερωτήματα 4 και 6, η απάντηση σας θα πρέπει να περιέχει εκτός από το query, εναλλακτικό Query Plan (πχ με force index), τα αντίστοιχα traces και τα συμπεράσματα σας από την μελέτη αυτών. 4 Να δοκιμάσετε διαφορετικές στρατηγικές join (π.χ. Nested Loop Join, Hash Join, Merge Join) για να αναλύσετε την επίδραση στη συνολική απόδοση.*

Στα ερωτήματα 4 και 6 φτιάξαμε δύο υλοποιήσεις και με την χρήση της εντολής EXPLAIN συγκρίναμε τα αποτελέσματα.

Για το ερώτημα 4:

```
-- With the indexes we saw it used Block Nested Loop Join
EXPLAIN FORMAT = JSON
SELECT
    a.artist_name,
    ROUND(AVG(r.artist_performance), 2) AS avg_artist_performance,
    ROUND(AVG(r.overall_impression), 2) AS avg_overall_impression
FROM
    review r FORCE INDEX (idx_review_ticket)
    JOIN ticket t FORCE INDEX (idx_ticket_event) ON r.ticket_ID =
t.ticket_ID
    JOIN events e ON t.event_ID = e.event_ID
    JOIN performances p FORCE INDEX (idx_perf_event_artist) ON e.event_ID
= p.event_ID
```

```
    AND t.event_ID = p.event_ID
    JOIN artist a FORCE INDEX (idx_artist_name) ON p.artist_ID =
a.artist_ID
WHERE
    a.artist_ID = (
        SELECT
            artist_ID
        FROM
            artist FORCE INDEX (idx_artist_name)
        WHERE
            artist_name = ' Albert Carr'
    )
    AND r.artist_performance IS NOT NULL
    AND r.overall_impression IS NOT NULL
GROUP BY
    a.artist_name;

-- Without indexes Block Nested Loop Join
SET
    join_cache_level = 8;

EXPLAIN FORMAT = JSO
SELECT
    a.artist_name,
    ROUND(AVG(r.artist_performance), 2) AS avg_artist_performance,
    ROUND(AVG(r.overall_impression), 2) AS avg_overall_impression
FROM
    review r IGNORE INDEX (idx_review_ticket)
    JOIN ticket t IGNORE INDEX (idx_ticket_event) ON r.ticket_ID =
t.ticket_ID
    JOIN events e ON t.event_ID = e.event_ID
    JOIN performances p IGNORE INDEX (idx_perf_event_artist) ON
e.event_ID = p.event_ID
    AND t.event_ID = p.event_ID
    JOIN artist a IGNORE INDEX (idx_artist_name) ON p.artist_ID =
a.artist_ID
WHERE
    a.artist_ID = (
        SELECT
            artist_ID
        FROM
            artist IGNORE INDEX (idx_artist_name)
        WHERE
            artist_name = 'Albert Carr'
    )
GROUP BY
    a.artist_name;
```

Στην πρώτη υλοποίηση κάνουμε εξαναγκασμένη χρήση των index και παρατηρήσαμε από το EXPLAIN ότι έγινε χρήση του Block-Nested-Loop Join όπως φαίνεται και παρακάτω:

Στην δεύτερη υλοποίηση κάνουμε εξαναγκασμένη χρήση χωρίς index και παρατηρήσαμε από το EXPLAIN ότι έγινε χρήση του Nested-Loop Join.

Για το ερώτημα 6:

```
-- using forced index it used Nested Loop Join
EXPLAIN FORMAT = JSON
SELECT
    CONCAT (v.first_name, ' ', v.last_name) AS visitor_name,
    e.event_name,
    ROUND(AVG(r.overall_impression), 2) AS avg_overall_impression
```

```
FROM
    visitor v FORCE INDEX (idx_visitor_full_name)
    JOIN ticket t ON v.visitor_ID = t.visitor_ID
    JOIN events e ON t.event_ID = e.event_ID
    JOIN review r ON t.ticket_ID = r.ticket_ID
WHERE
    v.first_name = 'Jason'
    AND v.last_name = 'Perez'
GROUP BY
    e.event_name;

-- Wihtout indexes still used Nested Loop Join
EXPLAIN FORMAT = JSON
SELECT
    CONCAT (v.first_name, ' ', v.last_name) AS visitor_name,
    e.event_name,
    ROUND(AVG(r.overall_impression), 2) AS avg_overall_impression
FROM
    visitor v IGNORE INDEX (idx_visitor_full_name)
    JOIN ticket t ON v.visitor_ID = t.visitor_ID
    JOIN events e ON t.event_ID = e.event_ID
    JOIN review r ON t.ticket_ID = r.ticket_ID
WHERE
    v.first_name = 'Jason'
    AND v.last_name = 'Perez'
GROUP BY
    e.event_name;
```

Στο ερώτημα 6 και στις δύο υλοποιήσεις που κάνουμε παρατηρήσαμε απο το EXPLAIN ότι έγινε χρήση του Nested-Loop Join.

Επειδή η βάση δεδομένων μας είναι μικρή δεν παρατηρήσουμε κάποια βελτίωση στην ταχύτητα εκτέλεσης στα ερωτήματα 4 και 6, ενώ θεωρητικά τα indexes επιταχύνουν την διαδικασία.

7.

DDL script (install.sql)

a. Υποενότητα 3.1

Σε αυτή την εργασία η SQL χρησιμοποιήθηκε ως DDL(Data Definition Language).

```sql
-- == TABLES CREATION == --

-- Festival
-- Stores basic information about each festival
CREATE TABLE festival (
    festival_ID INT PRIMARY KEY AUTO_INCREMENT,
    starting_date DATE NOT NULL,
    duration INT NOT NULL -- in days
);

-- Festival Location
-- Each festival can have one or more associated locations
CREATE TABLE festival_location (
    festival_ID INT,
    festival_location_ID INT PRIMARY KEY AUTO_INCREMENT,
    address VARCHAR(255) NOT NULL,
    town VARCHAR(100) NOT NULL,
    country VARCHAR(100) NOT NULL,
    continent VARCHAR(100) NOT NULL,
    geo_coordinates VARCHAR(100) NOT NULL,
    FOREIGN KEY (festival_ID) REFERENCES festival(festival_ID)
);

-- Personel
-- Stores personal and professional information about event staff
CREATE TABLE personel (
    personel_ID INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    expertise_status ENUM('intern', 'beginer', 'intermidiate', 'experienced', 'very_experienced')
);
```

```
-- Building
-- Details about buildings where events may take place
CREATE TABLE building (
    building_ID INT PRIMARY KEY AUTO_INCREMENT,
    building_name VARCHAR(255) NOT NULL,
    building_description TEXT NOT NULL,
    max_capacity INT NOT NULL
);

-- Technical Equipment
-- Stores information about technical equipment wanted from buildings
CREATE TABLE technical_equipment (
    technical_equipment_ID INT PRIMARY KEY AUTO_INCREMENT,
    building_ID INT,
    equipment_name VARCHAR(255) NOT NULL,
    equipment_description TEXT NOT NULL,
    FOREIGN KEY (building_ID) REFERENCES building(building_ID)
);
-- Artist
-- Stores artist profiles and metadata
CREATE TABLE artist (
    artist_ID INT PRIMARY KEY AUTO_INCREMENT,
    artist_name VARCHAR(255) NOT NULL,
    stage_name VARCHAR(255),  -- can be NULL
    artist_date_of_birth DATE NOT NULL,
    artist_debute DATE NOT NULL,
    artist_website VARCHAR(255),     -- can be NULL
    artist_instagram VARCHAR(255),   -- can be NULL
    num_of_consecutive_years_participating INT DEFAULT 0,
    CHECK (0 <= num_of_consecutive_years_participating AND
num_of_consecutive_years_participating <= 3) -- number of years the artist has participated in the
festival
);

-- Group
-- Stores information about groups, including their members
```

```
CREATE TABLE `group` ( -- renamed from group to avoid SQL keyword conflict
    group_ID INT PRIMARY KEY AUTO_INCREMENT,
    group_name VARCHAR(255) NOT NULL,
    group_date_of_birth DATE NOT NULL,
    group_debute DATE NOT NULL,
    group_website VARCHAR(255),     -- can be NULL
    group_instagram VARCHAR(255),   -- can be NULL
    member_names TEXT DEFAULT '',
    num_of_consecutive_years_participating INT DEFAULT 0,
    CHECK (0 <= num_of_consecutive_years_participating AND
num_of_consecutive_years_participating <= 3) -- number of years the artist has participated in the
festival


    );

-- Genre
-- Represents the genre of an artist or group
CREATE TABLE genre (
    genre_ID INT PRIMARY KEY AUTO_INCREMENT,
    genre_name VARCHAR(100) NOT NULL,
    subgenre_name VARCHAR(100),
    artist_ID INT,
    group_ID INT,
    FOREIGN KEY (artist_ID) REFERENCES artist(artist_ID),
    FOREIGN KEY (group_ID) REFERENCES `group`(group_ID)
);



-- Group Members
-- Stores the relationship between groups and their members
CREATE TABLE group_members (
    group_ID INT,
    artist_ID INT,
    PRIMARY KEY (group_ID, artist_ID),
    FOREIGN KEY (group_ID) REFERENCES `group`(group_ID),
    FOREIGN KEY (artist_ID) REFERENCES artist(artist_ID)
```

```
);

-- Events
-- Represents a specific event held as part of a festival
CREATE TABLE events (
    event_ID INT PRIMARY KEY AUTO_INCREMENT,
    festival_ID INT,
    event_name VARCHAR(255) NOT NULL,
    festival_day INT NOT NULL,
    event_start_time DATETIME NOT NULL,
    event_end_time DATETIME NOT NULL,
    building_ID INT,
    event_duration INT GENERATED ALWAYS AS (
        CASE
            WHEN event_end_time >= event_start_time THEN TIMESTAMPDIFF(MINUTE, event_start_time, event_end_time)
            ELSE TIMESTAMPDIFF(MINUTE, event_start_time, event_end_time) + 1440
        END
    ) STORED,
    FOREIGN KEY (building_ID) REFERENCES building(building_ID),
    FOREIGN KEY (festival_ID) REFERENCES festival(festival_ID), -- ,
    VIP_total INT,
    backstage_total INT,
    general_total INT
    );

-- Performances
-- Each performance belongs to an event and has a type and duration
CREATE TABLE performances (
    performance_ID INT PRIMARY KEY AUTO_INCREMENT,
    event_ID INT,
    performance_type ENUM('warm up', 'headline', 'special_guest', 'finale') NOT NULL,
    performance_start_time DATETIME NOT NULL,
    performance_end_time DATETIME NOT NULL, -- Time plain didnt work because an event can begin at 11:00 and end at next day 01:00
    performance_duration INT GENERATED ALWAYS AS (
```

```
        CASE
            WHEN performance_end_time >= performance_start_time THEN
TIMESTAMPDIFF(MINUTE, performance_start_time, performance_end_time)
            ELSE TIMESTAMPDIFF(MINUTE, performance_start_time, performance_end_time) +
1440
        END
    ) STORED,
    building_ID INT NOT NULL,
    building_name VARCHAR(255) NOT NULL,
    artist_ID INT DEFAULT NULL,
    group_ID INT DEFAULT NULL,
    FOREIGN KEY (event_ID) REFERENCES events(event_ID),
    FOREIGN KEY (building_ID) REFERENCES building(building_ID),
    FOREIGN KEY (artist_ID) REFERENCES artist(artist_ID),
    FOREIGN KEY (group_ID) REFERENCES `group`(group_ID),
    CHECK (performance_start_time < performance_end_time),
    CHECK (performance_duration <= 180) -- max duration of a performance is 3 hours
);

-- Personel-Event Relationship (many-to-many)
-- This table defines the role of each personel in each event
CREATE TABLE role_of_personel_on_event (
    personel_ID INT,
    event_ID INT,
    role ENUM('technical','security','support') NOT NULL,
    PRIMARY KEY (personel_ID, event_ID),
    FOREIGN KEY (personel_ID) REFERENCES personel(personel_ID),
    FOREIGN KEY (event_ID) REFERENCES events(event_ID)
);

-- Visitor
-- Stores personal data for individuals attending events
CREATE TABLE visitor (
    visitor_ID INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
```

```
    telephone VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    age INT NOT NULL
);

-- Ticket (one-to-many with visitor & event)
-- A ticket belongs to one visitor and one event
CREATE TABLE ticket (
    ticket_ID INT AUTO_INCREMENT PRIMARY KEY,
    event_ID INT,
    visitor_ID INT NOT NULL,
    barcode CHAR(13),
    FOREIGN KEY (visitor_ID) REFERENCES visitor(visitor_ID),
    FOREIGN KEY (event_ID) REFERENCES events(event_ID),

    ticket_type ENUM('general_admission', 'VIP', 'backstage') NOT NULL,
    purchase_date DATE,
    purchase_price DECIMAL(10, 2),
    payment_method ENUM('debit_card', 'credit_card', 'I-BAN'),
    activated_status BOOLEAN DEFAULT FALSE,
    visitor_name VARCHAR(100),
    visitor_last_name VARCHAR(100),
    visitor_email VARCHAR(100),
    visitor_telephone VARCHAR(20),
    visitor_age INT
);



-- Buyer
-- Represents users interested in buying tickets
CREATE TABLE buyer (
    buyer_ID INT PRIMARY KEY,
    FOREIGN KEY (buyer_ID) REFERENCES visitor(visitor_ID)
);
```

```sql
-- Seller
-- Represents users who are selling or listing tickets for resale
CREATE TABLE seller (
    seller_ID INT PRIMARY KEY ,
    FOREIGN KEY (seller_ID) REFERENCES visitor(visitor_ID)
);

-- Resale Queue (FIFO)
-- A queue for tickets listed for resale, based on timestamp
CREATE TABLE resale_queue (
    resale_ID INT AUTO_INCREMENT PRIMARY KEY,
    buyer_ID INT,
    seller_ID INT,
    event_name VARCHAR(255) NULL,
    ticket_type ENUM('general_admission', 'VIP', 'backstage') NULL,
    ticket_ID INT NULL,
    listed_at TIMESTAMP ,
    FOREIGN KEY (buyer_ID) REFERENCES visitor(visitor_ID),
    FOREIGN KEY (seller_ID) REFERENCES visitor(visitor_ID),
    FOREIGN KEY (ticket_ID) REFERENCES ticket(ticket_ID)
);



-- Review
-- Feedback for events by visitors who have activated tickets
-- (Use a trigger to ensure review is only allowed if ticket is activated)
CREATE TABLE review (
   ticket_ID INT NOT NULL,
   performance_ID INT NOT NULL,

   artist_performance ENUM('1', '2', '3', '4', '5'),
   sound_and_lighting ENUM('1', '2', '3', '4', '5'),
   stage_presence ENUM('1', '2', '3', '4', '5'),
   event_organization ENUM('1', '2', '3', '4', '5'),
   overall_impression ENUM('1', '2', '3', '4', '5'),
   PRIMARY KEY (ticket_ID, performance_ID),
```

```
  FOREIGN KEY (ticket_ID) REFERENCES ticket(ticket_ID),
  FOREIGN KEY (performance_ID) REFERENCES performances(performance_ID)
);



-- Temporary Table for Resale Matches
CREATE TABLE temp_resale_matches (
   match_ID INT AUTO_INCREMENT PRIMARY KEY,
   buyer_ID INT,
   seller_ID INT,
   ticket_ID INT,
   FOREIGN KEY (buyer_ID) REFERENCES buyer(buyer_ID),
   FOREIGN KEY (seller_ID) REFERENCES seller(seller_ID),
   FOREIGN KEY (ticket_ID) REFERENCES ticket(ticket_ID)
);

CREATE TABLE photo(
   photo_ID INT AUTO_INCREMENT PRIMARY KEY,
   photo_name VARCHAR(255) NOT NULL,
   photo_description TEXT NOT NULL,
   artist_ID INT,
   group_ID INT,
   performance_ID INT,
   event_ID INT,
   festival_ID INT,
   technical_equipment_ID INT,
   FOREIGN KEY (artist_ID) REFERENCES artist(artist_ID),
   FOREIGN KEY (group_ID) REFERENCES `group`(group_ID),
   FOREIGN KEY (performance_ID) REFERENCES performances(performance_ID),
   FOREIGN KEY (event_ID) REFERENCES events(event_ID),
   FOREIGN KEY (festival_ID) REFERENCES festival(festival_ID),
   FOREIGN KEY (technical_equipment_ID) REFERENCES
technical_equipment(technical_equipment_ID)
);
```

```
-- == INDEXES == --

CREATE INDEX idx_perf_event_artist ON performances(event_ID, artist_ID);
CREATE INDEX idx_artist_name ON artist(artist_name);
CREATE INDEX idx_perf_artist_event ON performances(artist_ID, event_ID);
CREATE INDEX idx_perf_group_event ON performances(group_ID, event_ID);
CREATE INDEX idx_events_festival ON events(festival_ID);
CREATE INDEX idx_ticket_visitor_event ON ticket(visitor_ID, event_ID);
CREATE INDEX idx_ticket_event ON ticket(event_ID);
CREATE INDEX idx_review_ticket ON review(ticket_ID);
CREATE INDEX idx_genre_artist ON genre(artist_ID);
CREATE INDEX idx_genre_group ON genre(group_ID);
CREATE INDEX idx_role_event_role ON role_of_personel_on_event(event_ID, role);
CREATE INDEX idx_visitor_full_name ON visitor(last_name, first_name);
CREATE INDEX idx_ticket_purchase_year_price ON ticket(purchase_date, purchase_price);
CREATE INDEX idx_perf_type_artist_event ON performances(performance_type, artist_ID, event_ID);
CREATE INDEX idx_role_event_role_personel ON role_of_personel_on_event(role, personel_ID, event_ID);
CREATE INDEX idx_personel_expertise ON personel(expertise_status);
CREATE INDEX idx_festival_location_continent ON festival_location(festival_ID, continent);


-- == TRIGGERS == --



-- Deletion Triggers
-- Prevent Festival Deletion Trigger
DELIMITER $$

CREATE TRIGGER prevent_festival_deletion
BEFORE DELETE ON festival
FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Festival cannot be deleted.';
END$$

DELIMITER ;

-- Prevent Performance Deletion Trigger
DELIMITER $$

CREATE TRIGGER prevent_performance_deletion
BEFORE DELETE ON performances
FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Performance cannot be deleted.';
END$$

DELIMITER ;

-- Ticket Triggers --
-- Ticket Trigger 1 --
-- Check if the ticket can be sold based on the event's capacity and ticket type limits

DELIMITER $$

CREATE TRIGGER check_ticket_availability
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
    DECLARE sold_total INT DEFAULT 0;
    DECLARE sold_vip INT DEFAULT 0;
    DECLARE sold_backstage INT DEFAULT 0;
    DECLARE sold_general INT DEFAULT 0;
    DECLARE vip_limit INT DEFAULT 0;
    DECLARE backstage_limit INT DEFAULT 0;
    DECLARE general_limit INT DEFAULT 0;
    DECLARE max_capacity INT DEFAULT 0;
```

```
-- Συνολικά εισιτήρια για το event
SELECT COUNT(*) INTO sold_total
FROM ticket
WHERE event_ID = NEW.event_ID;

-- Πωλημένα ανά τύπο
SELECT
    SUM(ticket_type = 'VIP'),
    SUM(ticket_type = 'backstage'),
    SUM(ticket_type = 'general_admission')
INTO sold_vip, sold_backstage, sold_general
FROM ticket
WHERE event_ID = NEW.event_ID;

-- Όρια εισιτηρίων για το event
SELECT VIP_total, backstage_total, general_total
INTO vip_limit, backstage_limit, general_limit
FROM events
WHERE event_ID = NEW.event_ID;

-- Χωρητικότητα του κτιρίου για το event
SELECT b.max_capacity
INTO max_capacity
FROM events e
JOIN building b ON e.building_ID = b.building_ID
WHERE e.event_ID = NEW.event_ID;

-- Έλεγχος χωρητικότητας
IF sold_total >= max_capacity THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Cannot sell ticket: event has reached maximum building capacity.';
END IF;

-- Έλεγχος διαθεσιμότητας ανά τύπο
IF NEW.ticket_type = 'VIP' AND sold_vip >= vip_limit THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot sell VIP ticket: limit reached.';
    ELSEIF NEW.ticket_type = 'backstage' AND sold_backstage >= backstage_limit THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot sell Backstage ticket: limit reached.';
    ELSEIF NEW.ticket_type = 'general_admission' AND sold_general >= general_limit THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot sell General Admission ticket: limit reached.';
    END IF;
END $$

DELIMITER ;

-- Ticket Trigger 2 --
-- When a new ticket is created, fill in visitor data from the visitor table
DELIMITER $$

CREATE TRIGGER fill_ticket_visitor_data
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
    DECLARE v_name VARCHAR(100);
    DECLARE v_last_name VARCHAR(100);
    DECLARE v_email VARCHAR(100);
    DECLARE v_phone VARCHAR(20);
    DECLARE v_age INT;

    -- Παίρνουμε τα στοιχεία του επισκέπτη από τον πίνακα visitor
    SELECT first_name, last_name, email, telephone, age
    INTO v_name, v_last_name, v_email, v_phone, v_age
    FROM visitor
    WHERE visitor_ID = NEW.visitor_ID;

    -- Αναθέτουμε τις τιμές στα αντίστοιχα πεδία του εισιτηρίου
    SET NEW.visitor_name = v_name;
    SET NEW.visitor_last_name = v_last_name;
```

```
    SET NEW.visitor_email = v_email;
    SET NEW.visitor_telephone = v_phone;
    SET NEW.visitor_age = v_age;
END$$

DELIMITER ;



-- Resale Triggers --
-- Resale Trigger 1 --
-- Trigger to check that the ticket is not activated before resale
DELIMITER $$

-- Trigger to check that the ticket is not activated before resale
CREATE TRIGGER check_ticket_activation_before_resale
BEFORE INSERT ON resale_queue
FOR EACH ROW
BEGIN
    -- Δηλώνουμε τη μεταβλητή ticket_activated στο αρχή του trigger
    DECLARE ticket_activated BOOLEAN;

    -- Ελέγχουμε αν ο πωλητής είναι ορισμένος και αν το ticket_ID δεν είναι NULL
    IF NEW.seller_ID IS NOT NULL AND NEW.ticket_ID IS NOT NULL THEN
        -- Ελέγχουμε την κατάσταση του εισιτηρίου στον πίνακα ticket
        SELECT activated_status INTO ticket_activated
        FROM ticket
        WHERE ticket_ID = NEW.ticket_ID;

        -- Αν το εισιτήριο είναι ενεργοποιημένο (activated_status = TRUE), επιστρέφουμε σφάλμα
        IF ticket_activated = TRUE THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Cannot resell an activated ticket.';
        END IF;
    END IF;
END$$
```

```
DELIMITER ;

-- Resale Trigger 2 --
--  Matching Seller and Buyer and Updating Ticket




-- Resale Trigger 4 --
-- Check if the ticket is sold out before allowing resale
DELIMITER $$

CREATE TRIGGER trg_check_soldout_before_resale
BEFORE INSERT ON resale_queue
FOR EACH ROW
BEGIN
   DECLARE event_id_val INT;
   DECLARE ticket_type_val ENUM('general_admission', 'VIP', 'backstage');
   DECLARE total_available INT;
   DECLARE sold_count INT;
   DECLARE msg_text VARCHAR(255);

   -- Περίπτωση 1: υπάρχει ticket_ID → πάρε event_ID και ticket_type από τον πίνακα ticket
   IF NEW.ticket_ID IS NOT NULL THEN
      SELECT event_ID, ticket_type
      INTO event_id_val, ticket_type_val
      FROM ticket
      WHERE ticket_ID = NEW.ticket_ID;

   -- Περίπτωση 2: δεν υπάρχει ticket_ID αλλά υπάρχει event_name και ticket_type
   ELSEIF NEW.event_name IS NOT NULL AND NEW.ticket_type IS NOT NULL THEN
      SELECT event_ID
      INTO event_id_val
      FROM events
      WHERE event_name = NEW.event_name
      LIMIT 1;  -- για ασφάλεια σε διπλότυπα ονόματα
```

```
        SET ticket_type_val = NEW.ticket_type;

    -- Περίπτωση 3: δεν υπάρχουν αρκετά στοιχεία για έλεγχο
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Not enough information to check resale availability.';
    END IF;

    -- Ανάλογα με τον τύπο εισιτηρίου, βρες το σύνολο διαθέσιμων
    IF ticket_type_val = 'VIP' THEN
        SELECT VIP_total INTO total_available FROM events WHERE event_ID = event_id_val;
    ELSEIF ticket_type_val = 'backstage' THEN
        SELECT backstage_total INTO total_available FROM events WHERE event_ID = event_id_val;
    ELSE
        SELECT general_total INTO total_available FROM events WHERE event_ID = event_id_val;
    END IF;

    -- Πόσα έχουν πουληθεί για το event και τον τύπο
    SELECT COUNT(*) INTO sold_count
    FROM ticket
    WHERE event_ID = event_id_val AND ticket_type = ticket_type_val;

    -- Έλεγχος αν επιτρέπεται το resale
    IF sold_count < total_available THEN
        SET msg_text = CONCAT('Resale not allowed: ', ticket_type_val, ' tickets are not sold out yet.');
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = msg_text;
    END IF;

END$$

DELIMITER ;
```

```
-- Resale Trigger 3 --
-- When a new resale entry of a buyer is created, add the buyer to the buyer table
-- When a new resale entry of a seller is created, add the seller to the seller table
DELIMITER $$

CREATE TRIGGER create_buyer_or_seller_after_visitor
BEFORE INSERT ON resale_queue
FOR EACH ROW
BEGIN
   -- Declare variables to check if the buyer or seller already exists
   DECLARE existing_buyer INT;
   DECLARE existing_seller INT;

   -- If buyer_ID is not NULL and it does not exist in the buyer table, insert it into the buyer table
   IF NEW.buyer_ID IS NOT NULL THEN
      -- Check if the buyer_ID already exists in the buyer table
      SELECT COUNT(*)
      INTO existing_buyer
      FROM buyer
      WHERE buyer_ID = NEW.buyer_ID;

      -- If the buyer doesn't exist, insert it into the buyer table
      IF existing_buyer = 0 THEN
         INSERT INTO buyer (buyer_ID)
         VALUES (NEW.buyer_ID);  -- Use NEW.buyer_ID as buyer_ID
      END IF;
   END IF;

   -- If seller_ID is not NULL and it does not exist in the seller table, insert it into the seller table
   IF NEW.seller_ID IS NOT NULL THEN
      -- Check if the seller_ID already exists in the seller table
      SELECT COUNT(*)
      INTO existing_seller
      FROM seller
      WHERE seller_ID = NEW.seller_ID;
```

```
        -- If the seller doesn't exist, insert it into the seller table
        IF existing_seller = 0 THEN
            INSERT INTO seller (seller_ID)
            VALUES (NEW.seller_ID);  -- Use NEW.seller_ID as seller_ID
        END IF;
    END IF;
END$$

DELIMITER ;

-- Resale Trigger 2 --
--  Matching Seller and Buyer and Updating Ticket

DELIMITER $$

CREATE TRIGGER match_resale_after_insert
BEFORE INSERT ON resale_queue
FOR EACH ROW
BEGIN
    DECLARE matched_seller INT;
    DECLARE matched_buyer INT;

    -- Εάν ο αγοραστής (buyer) είναι ορισμένος, κάνε την αντιστοίχιση με τον πωλητή
    IF NEW.buyer_ID IS NOT NULL THEN
        -- Βρες διαθέσιμο seller που δεν έχει ήδη γίνει match
        SELECT seller_ID INTO matched_seller
        FROM resale_queue
        WHERE ticket_ID = NEW.ticket_ID
          AND seller_ID IS NOT NULL
          AND buyer_ID IS NULL
          AND seller_ID NOT IN (
              SELECT seller_ID FROM temp_resale_matches WHERE ticket_ID = NEW.ticket_ID
          )
        ORDER BY listed_at ASC
        LIMIT 1;
```

```
    IF matched_seller IS NOT NULL THEN
        -- Εισαγωγή στο temp_resale_matches
        UPDATE ticket
        SET visitor_ID = NEW.buyer_ID
        WHERE ticket_ID = NEW.ticket_ID;

        INSERT INTO temp_resale_matches (buyer_ID, seller_ID, ticket_ID)
        VALUES (NEW.buyer_ID, matched_seller, NEW.ticket_ID);
    END IF;
END IF;

-- Εάν ο πωλητής (seller) είναι ορισμένος, κάνε την αντιστοίχιση με τον αγοραστή
IF NEW.seller_ID IS NOT NULL THEN
    -- Βρες διαθέσιμο buyer που δεν έχει ήδη γίνει match
    SELECT buyer_ID INTO matched_buyer
    FROM resale_queue
    WHERE ticket_ID = NEW.ticket_ID
      AND buyer_ID IS NOT NULL
      AND seller_ID IS NULL
      AND buyer_ID NOT IN (
          SELECT buyer_ID FROM temp_resale_matches WHERE ticket_ID = NEW.ticket_ID
      )
    ORDER BY listed_at ASC
    LIMIT 1;

    IF matched_buyer IS NOT NULL THEN
        -- Εισαγωγή στο temp_resale_matches
        UPDATE ticket
        SET visitor_ID = matched_buyer
        WHERE ticket_ID = NEW.ticket_ID;

        INSERT INTO temp_resale_matches (buyer_ID, seller_ID, ticket_ID)
        VALUES (matched_buyer, NEW.seller_ID, NEW.ticket_ID);
    END IF;
END IF;
```

```
END$$

DELIMITER ;




-- Event Triggers --
-- Event Trigger 1 --
-- Ensure that the festival_day is within the festival duration
DELIMITER $$

CREATE TRIGGER check_festival_day
BEFORE INSERT ON events
FOR EACH ROW
BEGIN
   DECLARE fest_duration INT;

   SELECT duration INTO fest_duration
   FROM festival
   WHERE festival_ID = NEW.festival_ID;

   IF NEW.festival_day > fest_duration THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'festival_day cannot be greater than festival duration.';
   END IF;
END$$

DELIMITER ;




-- Gerne Triggers --
```

```
-- Genre Trigger 1 --
-- Ensure that each genre is linked to either one artist or one group, but not both or neither
DELIMITER $$

CREATE TRIGGER check_genre_entity_exclusivity
BEFORE INSERT ON genre
FOR EACH ROW
BEGIN
   IF (NEW.artist_ID IS NOT NULL AND NEW.group_ID IS NOT NULL) OR
      (NEW.artist_ID IS NULL AND NEW.group_ID IS NULL) THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Each genre must be linked to either one artist OR one group (not both or neither).';
   END IF;
END $$

DELIMITER ;



--Performance Triggers--
-- Performance Trigger 1 --
-- Ensure a minimum 5-minute break between performances of the same event in the same building

DELIMITER $$

DELIMITER $$
CREATE TRIGGER check_performance_overlap
BEFORE INSERT ON performances
FOR EACH ROW
BEGIN
   DECLARE conflict_count INT;

   SELECT COUNT(*)
   INTO conflict_count
   FROM performances
```

```
    WHERE
        building_ID = NEW.building_ID

        AND (
            (NEW.performance_start_time BETWEEN performance_start_time - INTERVAL 301 SECOND AND performance_end_time + INTERVAL 299 SECOND )
            OR
            (NEW.performance_end_time BETWEEN performance_start_time - INTERVAL 301 SECOND AND performance_end_time + INTERVAL 299 SECOND )
        );

    IF conflict_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Conflict with another performance in the same building, same day, time range +/-5 minutes';
    END IF;
END$$

DELIMITER ;




-- Performance Trigger 2 --
-- Check for consecutive years of participation for artists
DELIMITER $$

CREATE TRIGGER trg_check_consecutive_years_artists
BEFORE INSERT ON performances
FOR EACH ROW
BEGIN
    DECLARE fest_year INT;
    DECLARE found_current_festival INT DEFAULT 0;
    DECLARE prev_year_exists INT DEFAULT 0;
    DECLARE curr_num INT DEFAULT 0;

    IF NEW.artist_ID IS NOT NULL THEN
```

```
    -- Πάρε τη χρονιά του φεστιβάλ μέσω του event_ID -> festival_ID
    SELECT YEAR(f.starting_date) INTO fest_year
    FROM events e
    JOIN festival f ON e.festival_ID = f.festival_ID
    WHERE e.event_ID = NEW.event_ID;

    -- Ελέγχει αν συμμετέχει ήδη φέτος
    SELECT COUNT(*) INTO found_current_festival
    FROM performances p
    JOIN events e ON p.event_ID = e.event_ID
    WHERE p.artist_ID = NEW.artist_ID
      AND YEAR((SELECT starting_date FROM festival WHERE festival_ID = e.festival_ID))
= fest_year;

    IF found_current_festival = 0 THEN

      -- Έλεγξε συμμετοχή το προηγούμενο έτος
      SELECT COUNT(*) INTO prev_year_exists
      FROM performances p
      JOIN events e ON p.event_ID = e.event_ID
      WHERE p.artist_ID = NEW.artist_ID
        AND YEAR((SELECT starting_date FROM festival WHERE festival_ID =
e.festival_ID)) = fest_year - 1;

      IF prev_year_exists > 0 THEN
        SELECT num_of_consecutive_years_participating INTO curr_num
        FROM artist
        WHERE artist_ID = NEW.artist_ID;

        IF curr_num >= 3 THEN
          SIGNAL SQLSTATE '45001'
          SET MESSAGE_TEXT = 'The artist cannot participate in more than 3 consecutive
years.';
        ELSE
          UPDATE artist
          SET num_of_consecutive_years_participating = curr_num + 1
          WHERE artist_ID = NEW.artist_ID;
```

```
            END IF;
        ELSE
            -- Reset if skipped a year
            UPDATE artist
            SET num_of_consecutive_years_participating = 1
            WHERE artist_ID = NEW.artist_ID;
        END IF;

    END IF;
  END IF;
END$$

DELIMITER ;




-- Performance Trigger 3 --
-- Check for consecutive years of participation for groups
DELIMITER $$

CREATE TRIGGER trg_check_consecutive_years_groups

BEFORE INSERT ON performances
FOR EACH ROW
BEGIN
  DECLARE fest_year INT;
  DECLARE found_current_festival INT DEFAULT 0;
  DECLARE prev_year_exists INT DEFAULT 0;
  DECLARE curr_num INT DEFAULT 0;

  IF NEW.group_ID IS NOT NULL THEN

    -- Πάρε τη χρονιά του φεστιβάλ μέσω του event_ID -> festival_ID
    SELECT YEAR(f.starting_date) INTO fest_year
    FROM events e
```

```
    JOIN festival f ON e.festival_ID = f.festival_ID
    WHERE e.event_ID = NEW.event_ID;

    -- Ελέγχει αν συμμετέχει ήδη φέτος
    SELECT COUNT(*) INTO found_current_festival
    FROM performances p
    JOIN events e ON p.event_ID = e.event_ID
    WHERE p.group_ID = NEW.group_ID
      AND YEAR((SELECT starting_date FROM festival WHERE festival_ID = e.festival_ID))
= fest_year;

    IF found_current_festival = 0 THEN

      -- Έλεγξε συμμετοχή το προηγούμενο έτος
      SELECT COUNT(*) INTO prev_year_exists
      FROM performances p
      JOIN events e ON p.event_ID = e.event_ID
      WHERE p.group_ID = NEW.group_ID
        AND YEAR((SELECT starting_date FROM festival WHERE festival_ID =
e.festival_ID)) = fest_year - 1;

      IF prev_year_exists > 0 THEN
        SELECT num_of_consecutive_years_participating INTO curr_num
        FROM `group`
        WHERE group_ID = NEW.group_ID;

        IF curr_num >= 3 THEN
          SIGNAL SQLSTATE '45001'
          SET MESSAGE_TEXT = 'The group cannot participate in more than 3 consecutive
years.';
        ELSE
          UPDATE `group`
          SET num_of_consecutive_years_participating = curr_num + 1
          WHERE group_ID = NEW.group_ID;
        END IF;
      ELSE
        -- Reset if skipped a year
```

```
            UPDATE `group`
            SET num_of_consecutive_years_participating = 1
            WHERE group_ID = NEW.group_ID;
        END IF;

    END IF;
  END IF;
END$$

DELIMITER ;



-- Performance Trigger 5 --
-- Check for overlapping performances for the same artist or group
DELIMITER $$

CREATE TRIGGER prevent_artist_group_overlap
BEFORE INSERT ON performances
FOR EACH ROW
BEGIN
  DECLARE overlap_count INT;

  SELECT COUNT(*) INTO overlap_count
  FROM performances p
  WHERE (
        (NEW.artist_ID IS NOT NULL AND p.artist_ID = NEW.artist_ID)
      OR (NEW.group_ID IS NOT NULL AND p.group_ID = NEW.group_ID)
      )
    AND (
      p.performance_start_time < NEW.performance_end_time AND
      p.performance_end_time > NEW.performance_start_time
    );

  IF overlap_count > 0 THEN
    SIGNAL SQLSTATE '45000'
```

```
    SET MESSAGE_TEXT = 'Artist or group has an overlapping performance.';
  END IF;
END$$

DELIMITER ;

-- Performance Trigger 6 --
-- Check for overlapping performances for the same artist or group on update
DELIMITER $$

CREATE TRIGGER prevent_artist_group_overlap_update
BEFORE UPDATE ON performances
FOR EACH ROW
BEGIN
  DECLARE overlap_count INT;

  SELECT COUNT(*) INTO overlap_count
  FROM performances p
  WHERE p.performance_ID != NEW.performance_ID
    AND (
        (NEW.artist_ID IS NOT NULL AND p.artist_ID = NEW.artist_ID)
      OR (NEW.group_ID IS NOT NULL AND p.group_ID = NEW.group_ID)
      )
    AND (
      p.performance_start_time < NEW.performance_end_time AND
      p.performance_end_time > NEW.performance_start_time
    );

  IF overlap_count > 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Artist or group has an overlapping performance (on update).';
  END IF;
END$$

DELIMITER ;
```

```
-- Ticket Triggers --
-- Ticket Trigger 1 --
-- VIP ticket limit check
-- Ensure that the number of VIP tickets does not exceed 10% of total tickets for the event
-- This is done using a trigger before inserting a new ticket

DELIMITER $$

CREATE TRIGGER check_vip_limit
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
   DECLARE vip_count INT;
   DECLARE total_count INT;

   IF NEW.ticket_type = 'VIP' THEN
      SELECT COUNT(*) INTO vip_count
      FROM ticket
      WHERE event_ID = NEW.event_ID AND ticket_type = 'VIP';

      SELECT COUNT(*) INTO total_count
      FROM ticket
      WHERE event_ID = NEW.event_ID;

      IF (vip_count + 1) > (0.1 * (total_count + 1)) THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'VIP ticket limit exceeded for this event.';
      END IF;
   END IF;
END$$

DELIMITER ;

-- Ticket Trigger 2 --
-- Prevent duplicate tickets for the same visitor and event
```

```
DELIMITER $$

CREATE TRIGGER prevent_duplicate_ticket
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
   DECLARE duplicate_count INT;

   SELECT COUNT(*) INTO duplicate_count
   FROM ticket
   WHERE event_ID = NEW.event_ID
     AND visitor_ID = NEW.visitor_ID;

   IF duplicate_count > 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Visitor already has a ticket for this event.';
   END IF;
END$$

DELIMITER ;

-- Group Triggers --
-- Group Trigger 1 --
-- When a new group member is added, update the member_names field in the group table

DELIMITER $$

CREATE TRIGGER group_member_names
AFTER INSERT ON group_members
FOR EACH ROW
BEGIN
   DECLARE artist_name_var VARCHAR(255);

   -- Get the artist_name from the artist table
   SELECT artist_name INTO artist_name_var
   FROM artist
```

```
    WHERE artist_ID = NEW.artist_ID;

    -- Update the member_names field in the group table
    UPDATE `group`
    SET member_names = CONCAT(member_names, artist_name_var, ', ')
    WHERE group_ID = NEW.group_ID;
END$$

DELIMITER ;



-- Review Triggers --
-- Review Trigger 1 --
-- Ensure that a review can only be created if the ticket is activated

DELIMITER $$

CREATE TRIGGER check_ticket_activation
BEFORE INSERT ON review
FOR EACH ROW
BEGIN
    DECLARE is_active BOOLEAN;

    SELECT activated_status INTO is_active
    FROM ticket
    WHERE ticket_ID = NEW.ticket_ID;

IF is_active = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot review with inactive ticket.';
    END IF;
END$$

DELIMITER ;
```

```
-- Review Trigger 2 --
-- Ensure that the performance belongs to the same event as the ticket and ticket is activated

DELIMITER $$

CREATE TRIGGER check_review_validity
BEFORE INSERT ON review
FOR EACH ROW
BEGIN
    DECLARE ticket_event INT;
    DECLARE performance_event INT;
    DECLARE is_activated BOOLEAN;

    -- Πάρε το event στο οποίο ανήκει το εισιτήριο και αν είναι ενεργοποιημένο
    SELECT event_ID, activated_status INTO ticket_event, is_activated
    FROM ticket
    WHERE ticket_ID = NEW.ticket_ID;

    -- Πάρε το event στο οποίο ανήκει το performance
    SELECT event_ID INTO performance_event
    FROM performances
    WHERE performance_ID = NEW.performance_ID;

    -- Έλεγχος 1: Το εισιτήριο πρέπει να είναι ενεργοποιημένο
    IF is_activated = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'You cannot review a performance unless your ticket is activated.';
    END IF;

    -- Έλεγχος 2: Το performance πρέπει να ανήκει στο event του εισιτηρίου
    IF ticket_event <> performance_event THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The performance you are trying to review does not belong to the
event of your ticket.';
    END IF;
```

```
END $$

DELIMITER ;

-- role_of_personel_on_event Triggers --
-- Role Trigger 1 --
-- Ensure that the same personel cannot have multiple roles in the same event
DELIMITER $$

CREATE TRIGGER check_personel_availability
BEFORE INSERT ON role_of_personel_on_event
FOR EACH ROW
BEGIN
   DECLARE overlap_count INT;

   SELECT COUNT(*) INTO overlap_count
   FROM role_of_personel_on_event r
   JOIN events e1 ON r.event_ID = e1.event_ID
   JOIN events e2 ON NEW.event_ID = e2.event_ID
   WHERE r.personel_ID = NEW.personel_ID
    AND (
        (e1.event_start_time <= e2.event_end_time AND e1.event_end_time >=
e2.event_start_time)
        );

   IF overlap_count > 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'This person is already assigned to an overlapping event.';
   END IF;
END $$

DELIMITER ;
```

```
-- == CONSTRAINTS == --
-- Resale Constraints --
-- Resale Constraint 1 --
ALTER TABLE resale_queue
ADD CONSTRAINT chk_seller_or_buyer CHECK (
   (
   ((ticket_ID IS NULL) AND (event_name IS NOT NULL) AND (ticket_type IS NOT NULL))
   OR
   ((ticket_ID IS NOT NULL) AND (event_name IS NULL) AND (ticket_type IS NULL))
    AND (buyer_ID IS NOT NULL)
   )
   OR
   (
   ((ticket_ID IS NOT NULL) AND (event_name IS NOT NULL) AND (ticket_type IS NOT NULL))
   AND (seller_ID IS NOT NULL)
   )
);


-- Resale Constraint 2 --
ALTER TABLE resale_queue
ADD CONSTRAINT chk_one_side_only CHECK (
   (buyer_ID IS NOT NULL AND seller_ID IS NULL)
   OR
   (buyer_ID IS NULL AND seller_ID IS NOT NULL)
);
-- Event Constraints --
-- Ο έλεγχος για  παράλληλα event γίνεται με trigger --



-- == CASCADES == --
ALTER TABLE role_of_personel_on_event
```

```
ADD CONSTRAINT fk_role_personel
FOREIGN KEY (personel_ID) REFERENCES personel(personel_ID)
ON DELETE CASCADE;


ALTER TABLE review
ADD CONSTRAINT fk_review_ticket
FOREIGN KEY (ticket_ID) REFERENCES ticket(ticket_ID)
ON DELETE CASCADE;


ALTER TABLE role_of_personel_on_event
ADD CONSTRAINT fk_role_event
FOREIGN KEY (event_ID) REFERENCES events(event_ID)
ON DELETE CASCADE;


ALTER TABLE group_members
ADD CONSTRAINT fk_group_members_group
FOREIGN KEY (group_ID) REFERENCES `group`(group_ID)
ON DELETE CASCADE;

-- == EVENTS == --
--  Delete the matched resale entry from the resale_queue after some time
CREATE EVENT delete_matched_resale_entries
ON SCHEDULE EVERY 1 HOUR
DO
  DELETE FROM resale_queue
  WHERE ticket_ID IN (SELECT ticket_ID FROM temp_resale_matches)
    AND (
      (buyer_ID IS NOT NULL AND seller_ID IS NULL) OR
      (buyer_ID IS NULL AND seller_ID IS NOT NULL)
    );
```

8.

ΔΙΕΥΚΡΙΝΙΣΕΙΣ

a. Διαγράμματα

Σχεδιασμός και Οπτικοποίηση της Βάσης Δεδομένων

- Το **ER διάγραμμα** δημιουργήθηκε με χρήση του εργαλείου **draw.io**.
- Χρησιμοποιήθηκε η λειτουργία **Export Schema** από τον **PHPMyAdmin Designer** για την εξαγωγή του αρχικού relational diagram.
- Για καλύτερη αισθητική και καθαρή παρουσίαση, δημιουργήθηκαν δύο εκδόσεις:
    - Μία **relational έκδοση** με βάση την εξαγωγή του PHPMyAdmin.
    - Μία **βελτιωμένη (prettier)** έκδοση στο **DB diagram.io**, με καλύτερη στοίχιση και οργάνωση των οντοτήτων.

Σημαντική παρατήρηση: Η τελευταία εφαρμογή (DB diagram.io) απαιτεί τροποποίηση του SQL κώδικα, καθώς δεν υποστηρίζει πλήρως τη σύνταξη της MySQL. Πραγματοποιήθηκαν αλλαγές για συμβατότητα.

b. Κώδικας

i. 4.2.1 Δομή κώδικα

Το αρχείο festival_marina.sql έχει την εξής δομή:

- Tables
- Indexes
- Triggers
- Constrains
- Cascades

ii. 4.2.2 Μεθοδολογία ανάπτυξης

Τα βήματα που ακολουθήθηκαν στην εργασία ήταν τα εξής

1. ER Diagram development
2. Table development (sql) in festival_marina.sql
3. Python dummy data creator (python) data_creator.py
4. Triggers incorporated (sql) in festival_marina.sql
5. Update python accordingly (python) data_creator.py
6. Backup creation backup
7. Queries (sql)
8. Indexes to facilitate queries
9. Cascades
10. Export ER Diagram
11. Export relational diagram
12. Renaming according to the instructions
13. Readme και Αναφορά

iii. 4.2.3 Περιβάλλον ανάπτυξης

iv. Τεχνολογίες και Περιβάλλον Ανάπτυξης

- **DBMS (Data Base Management System):** Χρησιμοποιήσαμε MySQL / MariaDB.
- **Περιβάλλον Ανάπτυξης & Web Stack:** Εργαστήκαμε με το XAMPP, το οποίο περιλαμβάνει Apache, MySQL / MariaDB και PHP.
- **Συνεργατική ανάπτυξη:** Για την απομακρυσμένη συνεργασία και τον συγχρονισμό του κώδικα χρησιμοποιήσαμε το GitHub.

v. Ανάπτυξη της Βάσης Δεδομένων

- Η ανάπτυξη της βάσης πραγματοποιήθηκε στο **Visual Studio Code**.
- Τα **dummy δεδομένα** δημιουργήθηκαν αυτόματα με το script data_creator2304.py, αξιοποιώντας τη βιβλιοθήκη **faker**.

c. Παραδοχές

i. Παραδοχές βάσει ER Διαγράμματος και Triggers

1. Εισιτήρια & Επισκέπτες

- Κάθε επισκέπτης μπορεί να αγοράσει μόνο ένα εισιτήριο για μία συγκεκριμένη παράσταση και ημέρα του φεστιβάλ.

- Ένας επισκέπτης μπορεί να έχει συνολικά πολλά εισιτήρια, αρκεί κάθε εισιτήριο να αφορά διαφορετική παράσταση ή/και ημέρα.

2. Κανόνες Μεταπώλησης (Resale)

- Η μεταπώληση εισιτηρίων ενεργοποιείται όχι όταν εξαντληθούν όλα τα εισιτήρια ενός event, αλλά όταν εξαντληθεί ένας συγκεκριμένος τύπος εισιτηρίου (π.χ. μόνο τα VIP).

- Η εισαγωγή στην ουρά μεταπώλησης (resale_queue) επιτρέπεται μόνο όταν ισχύει μία από τις παρακάτω περιπτώσεις:

  Πωλητής (Seller):

  - ticket_ID IS NOT NULL
  - event_name IS NOT NULL
  - ticket_type IS NOT NULL

- Αγοραστής (Buyer):

  - Περίπτωση 1: Θέλει εισιτήριο τύπου χωρίς να ξέρει ποιο ακριβώς:

        - ticket_ID IS NULL
        - event_name IS NOT NULL
        - ticket_type IS NOT NULL
    - Περίπτωση 2: Θέλει να αγοράσει συγκεκριμένο εισιτήριο:
        - ticket_ID IS NOT NULL
        - event_name IS NULL
        - ticket_type IS NULL

- Η μεταπώληση επιτρέπεται μόνο για μη ενεργοποιημένα εισιτήρια.
- Η μεταπώληση επιτρέπεται μόνο πριν από το DATETIME έναρξη του event.
- Οι αγοραστές μπορούν να εκδηλώσουν ενδιαφέρον είτε για συγκεκριμένο ticket_ID, είτε για συνδυασμό event_name και ticket_type.
- Η αντιστοίχιση αγοραστών και πωλητών γίνεται αυτόματα μέσω trigger (match_resale_after_insert), και το αποτέλεσμα καταχωρείται στον πίνακα temp_resale_matches. Δηλαδή στον τελευταίο πίνακα
- Η ουρά λειτουργεί με σειρά προτεραιότητας FIFO.

3. Χρηματικές Συναλλαγές

- Οι χρεώσεις και πιστώσεις δεν υλοποιούνται σε επίπεδο βάσης δεδομένων (δεν υπάρχουν σχετικά πεδία ή πίνακες).
- Η λειτουργία αυτή θεωρείται ότι καλύπτεται εξωτερικά, σε επίπεδο εφαρμογής (backend), και δεν υλοποιείται με triggers ή SQL logic.

4. Αξιολόγηση Εμφανίσεων (Reviews)

    - Μόνο κάτοχοι ενεργοποιημένων εισιτηρίων μπορούν να υποβάλουν αξιολόγηση, σύμφωνα με trigger (check_ticket_activation).

    - Επιτρέπεται η αξιολόγηση μόνο αν το performance ανήκει στο ίδιο event με το εισιτήριο (check_review_validity).

    - Η αξιολόγηση γίνεται με βάση πέντε κριτήρια (ερμηνεία, ήχος και φωτισμός, σκηνική παρουσία, οργάνωση, συνολική εντύπωση), με χρήση κλίμακας Likert (ENUM: '1', '2', '3', '4', '5') — όχι με αριθμητικούς τύπους δεδομένων.

5. Εξοπλισμός Κτιρίων

    - Το πεδίο technical_equipment του πίνακα building περιγράφει τον αναμενόμενο εξοπλισμό για τις εκδηλώσεις, όχι τον πραγματικά διαθέσιμο — αποτελεί στατική απαίτηση του εκάστοτε χώρου.

6. Εικόνες και Περιγραφές

    - Στη βάση δεδομένων δεν υπάρχουν πεδία για εικόνες ή λεκτικές περιγραφές για οντότητες όπως festival, artist, building, equipment.

    - Η δυνατότητα προσθήκης εικόνων και πολυμέσων αποτελεί παραδοχή σχεδιαστικής επέκτασης, η οποία μπορεί να καλυφθεί με την προσθήκη πεδίων όπως image_url, image_description.

Παρατήρηση για την Python Η Python τα εισιτήρια μας τα δίνει ήδη κάποια εργοποιημένα και κάποια όχι ενεργοποιημένα.