Βάσεις Δεδομένων

Εξαμηνιαία Εργασία

6ο Εξάμηνο

Pulse University

Μέλη ομάδας:

Βασίλης Πάντζος ΑΜ:03121281

Αχχιλέας Φέλιξ ΑΜ:03120902

Σπύρος Μάλλιος ΑΜ:03122145

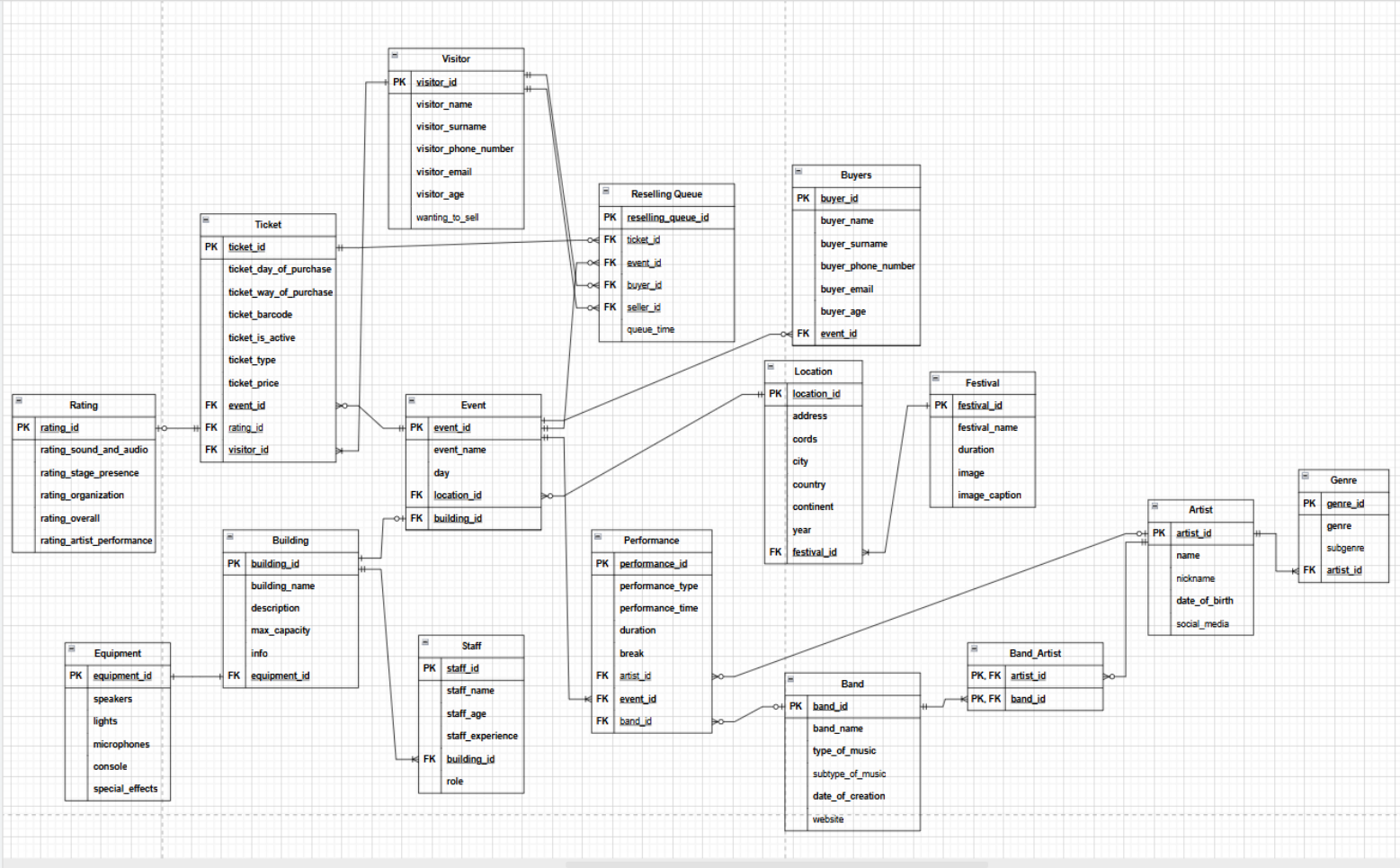
Github Link: https://github.com/pulseuni/pulse_university.git

Περιεχόμενα:

1) Βάση Δεδομένων

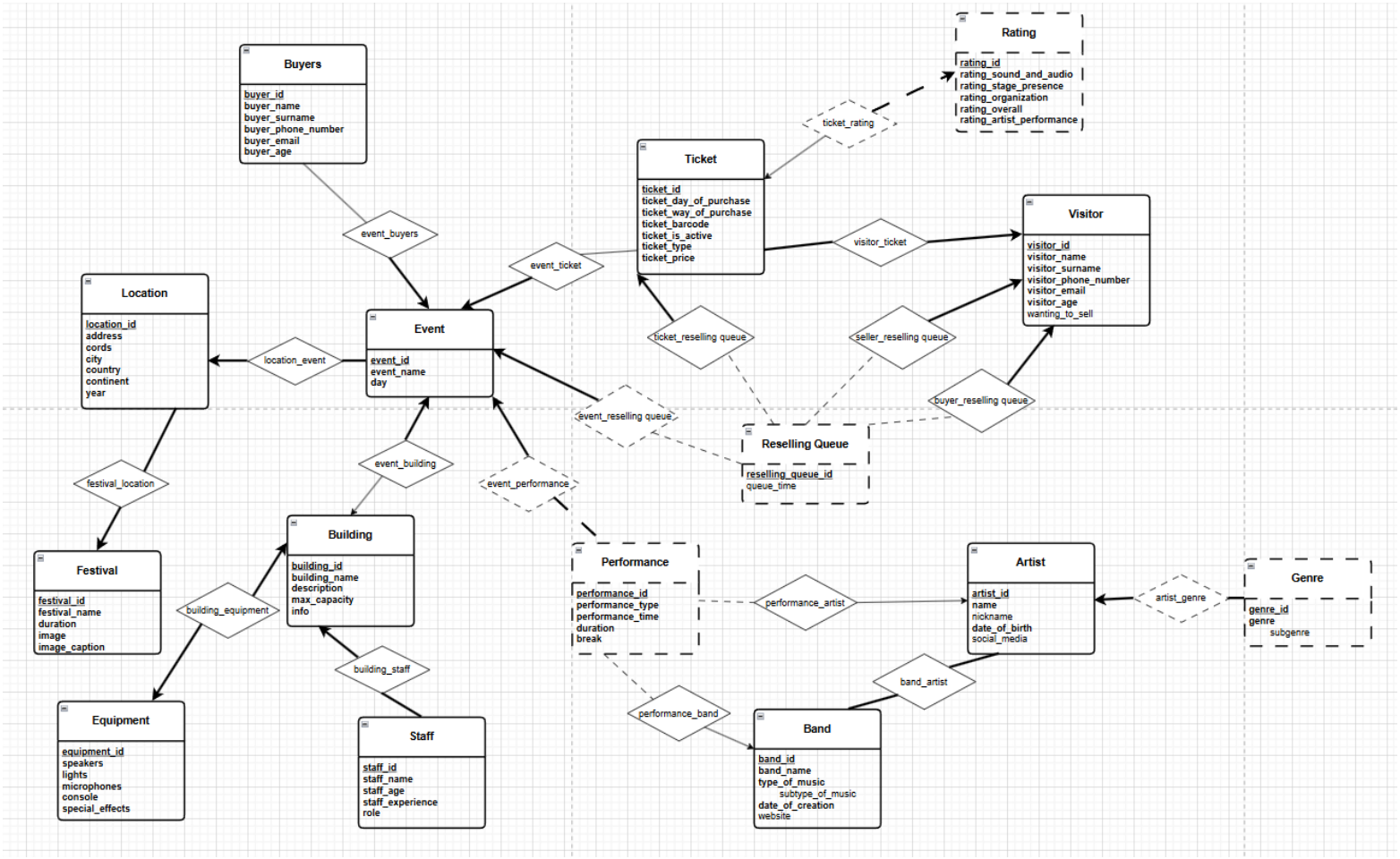
1.1) ER Diagram

Παρακάτω φαίνεται το ER διάγραμμα για το pulse university:



1.2) Relational Schema

Παρακάτω φαίνεται το Relational σχήμα για το pulse university:



1.3) Επεξήγηση του Relational Schema

Πέρα από την σύνδεση των tables όπως φαίνεται στο 1.2, έχει γίνει χρήση αρκετών trigger για να εξασφαλιστεί η ορθότητα της βάσης.

Τα trigger είναι τα ακόλουθα:

Prevent_band_double_booking: Αυτό το trigger μας διασφαλίζει ότι δεν υπάρχει χρονική επικάλυψη μεταξύ των performances.

Prevent_band_consecutive_years: Αυτό το trigger είναι για τον περιορισμό εμφάνισης της ίδιας band για περισσότερα από 3 συνεχόμενα χρόνια .

Τα trigger Prevent_artist_double_booking και prevent_artist_consecutive_years έχουν παρόμοια λειτουργία με τα παραπάνω αλλά για τους artist.

Check_ticket_capacity: Αυτό το trigger ελέγχει μέσω του max_capacity για κάθε bulding έτσι ώστε να μην έχουμε παραπάνω tickets από το max_capacity

Check_vip_ticket_limit: Αυτό το trigger χρησιμοποιείται για τον περιορισμό του 10% για τα VIP tickets από την εκφώνηση

Check_consecutive_year_location_insert και Check_consecutive_year_location_update: Αυτό το trigger μας βοηθάει έτσι ώστε το ίδο φεστιβάλ να μην γίνεται στην ίδια τοποθεσία για 2 συνεχόμενα χρόνια.

Check_performance_schedule: Μας εξασφαλίζει ότι δεν θα έχουμε performances που κάνουν overlap.

Prevent_staff_drop: Μας εξασφαλίζει τους περιορισμούς της εκφώνησης για τους security και helpers.

Prevent_duplicate_visitor_ticket: Αποτρέπει έναν visitor να έχει 2 ticket για το ίδιο event την ίδια μέρα.

Reselling queue implimentation:

Ορίζουμε την δυνατότητα πώλησης ενός εισιτηρίου όταν κάποιος επισκέπτης είναι πρόθυμος να πουλήσει το εισιτήριο του και το ίδιο το εισιτήριο είναι active,δεδομένου ότι έχουμε φτάσει το συγκεκριμένο max_capacity για το building που γίνεται το event.Άρα εάν ένας visitor θέλει να πουλήσει το εισιτήριο του πρέπει να πληρούνται οι παραπάνω προϋποθέσεις. Έπειτα άμα υπάρχει insert ενός buyer τότε ο πρωτος seller (που είναι ο visitor που πληρεί τις προυποθέσεις που είπαμε) για το συγκεκριμένο event που εδιαφέρεται ο buyer τότε θα γίνει ενα update στον πίνακα visitor και θα εισάγουμε τα στοιχεία του buyer με το ticket_id του seller.Τέλος, θα γίνει το αντίστοιχο query pop από το reselling_queue.Αυτή η λειτουργία γίνεται με τα triggers:

Trg_visitor_after_insert

Trg_after_buyer_insert

Trg_visitor_after_update

Για όλα τα trigger η λογική και η διαδικασία που θα κάνει το καθένα υλοποιήθηκε από εμάς και χρησιμοποιήθηκε llm για την αποφυγή συντακτικού λάθους.

INDEXES

Χρησιμοποιήσαμε indexes στην άσκηση μας για να βελτιώσουμε την ταχύτητα εκτέλεσης των εντολών WHERE, ORDER BY και GROUP BY. Ειδικότερα,

```
CREATE INDEX idx_genre ON genre(genre);

CREATE INDEX idx_artist_name ON artist(name);
CREATE INDEX idx_artist_date_of_birth ON artist(date_of_birth);

CREATE INDEX idx_performance_type ON performance(performance_type);

CREATE INDEX idx_staff_role ON staff(role);

CREATE INDEX idx_festival_name ON festival(festival_name);

CREATE INDEX idx_event_day ON event(day);
```

Το idx_genre χρησιμοποείται ώστε επειδή ψάχνουμε τους καλλιτέχνες που ανήκουν σε ένα συγκεκριμένο μουσικό είδος (genre).

Το idx_artist_name χρησιμοποιείται επειδή ψάχνουμε αρκετές φορές συγκεκριμένους καλλιτέχνες μέσω του ονόματός τους.

Το idx_artist_date_of_birth χρησιμοποιείται όταν ρωτάμε ποιοι καλλιτέχνες είναι μικρότεροι των 30 ετών.

Το idx_performance_type βοηθάει στο να βρεθεί πιο γρήγορα ποιες εμφανίσεις είναι Warmup.

Το idx_staff_role μας βοηθάει στο να βρίσκουμε πιο γρήγορα το προσωπικό που ανήκει σε ένα συγκεκριμένο ρόλο, διαχωρισμός ο οποίος εμφανίζεται αρκετές φορές.

Το idx_festival_name διευκολύνει την ομαδοποίηση των στοιχείων ανά φεστιβάλ, το οποίο και μας ζητείται σε διάφορα ερωτήματα.

Το idx_event_day χρησιμοποιείται όταν ψάχνουμε μία συγκεκριμένη ημερομηνία παράστασης, και το προσωπικό που δεν δουλεύει εκείνη την ημέρα.

Όμως παρατηρήσαμε πως δεν υπάρχει έντονη διαφορά στον χρόνο εκτέλεσης, καθώς είναι χρήσιμα σε μεγάλο όγκο δεδομένων όπου και ελαττώνουν σε μεγάλο βαθμό την ταχύτητα εκτέλεσης των παραπάνω εντολών. Τελικά, φαίνεται ότι βοηθάνε στο να μειώσουν την πολυπλοκότητα (μειώνοντας τον χρόνο εκτέλεσης) των εντολών αυτών.

1.4) Dummy Data

Στο αρχείο load.sql έχουμε τα dummy data που χρησιμοποιήθηκαν για την ομαλή λειτουργία της βάσης καθώς και τον έλεγχο της ορθότητας των trigger. Επίσης στο αρχείο buyer_example.sql υπάρχει ένα παράδειγμα insert για το reselling_queue .Για την δημιουργία των insert χρησιμοποιήθηκε llm με συγκεκριμένες οδηγίες απο εμάς για το πως θα είναι το κάθε insert σε κάθε table με βάση τον σχεδιασμό μας. Η αφαίρεση των foreign keys στην αρχή και η πρόσθεση τους στο τέλος έγινε από εμάς.

Ενδεικτικά έχουμε 10 festival, 60 event ,200 performance,200 tickets ανα building,60 building,25 staff ανά building, και 9600 συνολικα ratings.

2) Select Queries

1)

```
SELECT fest.festival_name, loc.year, SUM(tick.ticket_price) as total_price, tick.ticket_way_of_purchase
FROM festival fest
INNER JOIN location loc
ON loc.festival_id = fest.festival_id
INNER JOIN event ev
ON ev.location_id = loc.location_id
INNER JOIN ticket tick
ON tick.event_id = ev.event_id
GROUP BY loc.year, fest.festival_name, tick.ticket_way_of_purchase;
```

SUM(tick.ticket_price) μας δίνει της συνολική τιμή των εισιτηρίων και με το GROUP BY φροντίζουμε η τιμή να υπολογίζεται για κάθε ξεχωριστό συνδυασμό φεστιβάλ, τρόπο πληρωμής και χρονιά

2)

```
SELECT art.name, art.nickname, art.date_of_birth, gen.genre, gen.subgenre, art.social_media, IF(count(IF(loc.year = 2025, 1, NULL)) > 0, 'True', 'False') as appeared
FROM genre gen
INNER JOIN artist art
ON gen.artist_id = art.artist_id
LEFT JOIN performance perf
ON art.artist_id = perf.artist_id
LEFT JOIN event ev
ON perf.event_id = ev.event_id
LEFT JOIN location loc
ON ev.location_id = loc.location_id
WHERE gen.genre = 'Metal'
GROUP BY art.name;
```

IF(loc.year = 2025, 1, NULL) μας κάνει την τιμή 1αν η χρονιά είναι το 2025. Με το count βρίσκουμε σε πόσες εκδηλώσεις του 2025 συμμετείχε κάθε καλλιτέχνης και το εξωτερικό if μας δείχνει αν ο καλλιτέχνης συμμετείχε σε κάποια εκδήλωση το 2025 ή όχι.

3)

```
SELECT DISTINCT name, nickname, date_of_birth, social_media, festival_id FROM (
        SELECT art.name as name, art.nickname as nickname, art.date_of_birth as date_of_birth, art.social_media as social_media, fest.festival_id as festival_id,
        ROW_NUMBER() OVER (PARTITION BY art.name, art.nickname, art.date_of_birth, art.social_media, fest.festival_id ORDER BY fest.festival_id) as count
        FROM artist art
        LEFT JOIN performance perf
        ON art.artist_id = perf.artist_id
        LEFT JOIN event ev
        ON perf.event_id = ev.event_id
        LEFT JOIN location loc
        ON ev.location_id = loc.location_id
        LEFT JOIN festival fest
        ON loc.festival_id = fest.festival_id
        WHERE perf.performance_type = 'Warmup'
) as count_table
WHERE count > 2;
```

ROW_NUMBER() OVER (PARTITION BY μας δείχνει σε μια στήλη το κάθε warm up performance του κάθε καλλιτέχνη και όταν πρόκειται για τον ίδιο καλλιτέχνη αυξάνεται ο αριθμός της στήλης αυτής κατά 1.

SELECT DISTINCT μας δίνει τους καλλιτέχνες αυτούς που ΄έχουν πάνω από 2 warm up performances.

4)

```
SELECT art.name, avg(rat.rating_artist_performance) as average_artist_rating, avg(rat.rating_overall) as average_overall_rating
FROM rating rat
INNER JOIN ticket tick
ON rat.rating_id = tick.rating_id
INNER JOIN event ev
ON ev.event_id = tick.event_id
INNER JOIN performance perf
ON perf.event_id = ev.event_id
INNER JOIN artist art
ON perf.artist_id = art.artist_id
WHERE art.name = 'Sara Jones'
GROUP BY art.name;
```

WHERE art.name = 'Sara Jones' διαλέγεις τον καλλιτέχνη για τον οποίο θες να βρεις τον μέσο όρο.

Με τα avg βρίσκεις τον μέσο όρο των ratings ανά κατηγορία.

5)

```
SELECT name, nickname, date_of_birth, social_media, appearances FROM (
        SELECT art.name as name, art.nickname as nickname, art.date_of_birth as date_of_birth, art.social_media as social_media, count(fest.festival_id) as appearances
        FROM artist art
        LEFT JOIN performance perf
        ON art.artist_id = perf.artist_id
        LEFT JOIN event ev
        ON perf.event_id = ev.event_id
        LEFT JOIN location loc
        ON ev.location_id = loc.location_id
        LEFT JOIN festival fest
        ON loc.festival_id = fest.festival_id
        WHERE art.date_of_birth > DATE_SUB(CURDATE(), INTERVAL 30 YEAR)
        GROUP BY art.name
) as appearances_table_1
WHERE appearances = (
        SELECT MAX(appearances) FROM (
                SELECT art.name as name, art.nickname as nickname, art.date_of_birth as date_of_birth, art.social_media as social_media, count(fest.festival_id) as appearances
                FROM artist art
                LEFT JOIN performance perf
                ON art.artist_id = perf.artist_id
                LEFT JOIN event ev
                ON perf.event_id = ev.event_id
                LEFT JOIN location loc
                ON ev.location_id = loc.location_id
                LEFT JOIN festival fest
                ON loc.festival_id = fest.festival_id
                WHERE art.date_of_birth > DATE_SUB(CURDATE(), INTERVAL 30 YEAR)
                GROUP BY art.name
        ) as appearances_table_2
);
```

Με το 2ο SELECT βρίσκουμε τις συμμετοχές σε κάθε φεστιβάλ για κάθε καλιτέχνη. Τον πίνακα που δημιουργήθηκε τον χρησιμοποιούμε για να βρούμε τον μέγιστο αριθμό εμφανίσεων. Με το 1ο SELECT βρίσκουμε τους καλλιτέχνες που έχουν μέγιστο αριθμό εμφανίσεων.

6)

```
SELECT vis.visitor_id, vis.visitor_name, vis.visitor_surname, ev.event_name, ev.event_id, (rating_sound_and_audio + rating_stage_presence + rating_organization + rating_overall +
rating_artist_performance)/5 as average_rating
FROM rating rat
INNER JOIN ticket tick
ON rat.rating_id = tick.rating_id
INNER JOIN event ev
ON ev.event_id = tick.event_id
INNER JOIN visitor vis
ON tick.visitor_id = vis.visitor_id
WHERE vis.visitor_id = 2;
```

Με το WHERE διαλέγουμε τον visitor που θέλουμε και με την πράξη αυτή rating_sound_and_audio + rating_stage_presence + rating_organization + rating_overall + rating_artist_performance)/5 βρίσκουμε τον μέσο όρο του συνολικού rating ενός visitor για μια παράσταση.

7)

```
SELECT festival_location, average_experience as experience FROM (
        SELECT fest.festival_name as festival_location, avg(st.staff_experience) as average_experience
        FROM staff st
        INNER JOIN building build
        ON st.building_id = build.building_id
        INNER JOIN event ev
        ON build.building_id = ev.building_id
        LEFT JOIN location loc
        ON ev.location_id = loc.location_id
        LEFT JOIN festival fest
        ON loc.festival_id = fest.festival_id
        WHERE st.role = 'technical'
        GROUP BY fest.festival_name
) as experience_table_1
WHERE average_experience = (
        SELECT MIN(average_experience) FROM (
                SELECT fest.festival_name as festival_location, avg(st.staff_experience) as average_experience
                FROM staff st
                INNER JOIN building build
                ON st.building_id = build.building_id
                INNER JOIN event ev
                ON build.building_id = ev.building_id
                LEFT JOIN location loc
                ON ev.location_id = loc.location_id
                LEFT JOIN festival fest
                ON loc.festival_id = fest.festival_id
                WHERE st.role = 'technical'
                GROUP BY fest.festival_name
        ) as experience_table_2
);
```

Με το 2ο SELECT βρίσκουμε τη μέση εμπειρία του προσωπικού για κάθε φεστιβάλ. Τον πίνακα που δημιουργήθηκε τον χρησιμοποιούμε για να βρούμε την ελάχιστη μέση εμπειρία του προσωπικού. Με το 1ο SELECT βρίσκουμε το φεστιβάλ με την ελάχιστη εμπειρία προσωπικού.

8)

```
SELECT st.staff_name, st.staff_age, st.staff_experience, st.role, ev.day
FROM staff st
INNER JOIN building build
ON st.building_id = build.building_id
INNER JOIN event ev
ON ev.building_id = build.building_id
WHERE ev.day != '2026-04-22';
```

Με το WHERE βρίσκουμε το προσωπικό, το οποίο δεν δούλεψε στις ‘2026-04-22’.

9)

```
WITH visited as (
        SELECT vis.visitor_id as visitor_id, vis.visitor_name as visitor_name, vis.visitor_surname as visitor_surname, count(ev.event_id) as events_seen, loc.year as year
        FROM visitor vis
        INNER JOIN ticket tick
        ON tick.visitor_id = vis.visitor_id
        INNER JOIN event ev
        ON ev.event_id = tick.event_id
        INNER JOIN location loc
        ON ev.location_id = loc.location_id
        GROUP BY vis.visitor_id, loc.year
        HAVING events_seen > 3
)
SELECT visited_1.*, visited_2.*
FROM visited visited_1
INNER JOIN visited visited_2
ON visited_1.visitor_id != visited_2.visitor_id
WHERE visited_1.events_seen = visited_2.events_seen;
```

Με το table visited βρίσκουμε τον αριθμό παραστάσεων που είδε κάθε επισκέπτης, εφόσον αυτός α αριθμός είναι μεγαλύτερος από 3. Κάνοντας ένα self join στο table visited μπορούμε να βρούμε τους διαφορετικούς επισκέπτες, που έχουν δει τον ίδιο αριθμό παραστάσεων.

10)

```
SELECT LEAST(gen_1.genre, gen_2.genre) as genre_1, GREATEST(gen_1.genre, gen_2.genre) as genre_2, (count(perf.performance_id)) DIV 2 as appearances
FROM genre gen_1
INNER JOIN genre gen_2
ON gen_1.artist_id = gen_2.artist_id AND gen_1.genre != gen_2.genre
INNER JOIN artist art
ON gen_2.artist_id = art.artist_id
INNER JOIN performance perf
ON perf.artist_id = art.artist_id
GROUP BY genre_1, genre_2
ORDER BY appearances DESC
LIMIT 3;
```

Στο join βάζουμε 2 φορές τον πίνακα genre ώστε να εμφανιστούν τα πιθανά ζεύγη μουσικών ειδών. Με τα LEAST και GREATEST φροντίζουμε ώστε τα ζεύγη να μην επαναλαμβάνονται δύο φορές όταν αλλάζει η σειρά των μουσικών ειδών μέσα στο ζεύγος.

11)

```
SELECT name, nickname, date_of_birth, social_media, appearances FROM (
        SELECT art.name as name, art.nickname as nickname, art.date_of_birth as date_of_birth, art.social_media as social_media, count(fest.festival_id) as appearances
        FROM artist art
        LEFT JOIN performance perf
        ON art.artist_id = perf.artist_id
        LEFT JOIN event ev
        ON perf.event_id = ev.event_id
        LEFT JOIN location loc
        ON ev.location_id = loc.location_id
        LEFT JOIN festival fest
        ON loc.festival_id = fest.festival_id
        GROUP BY art.name
        ORDER BY appearances DESC
) as appearances_table_1
WHERE appearances <= (
        SELECT MAX(appearances) - 5 FROM (
                SELECT art.name as name, art.nickname as nickname, art.date_of_birth as date_of_birth, art.social_media as social_media, count(fest.festival_id) as appearances
                FROM artist art
                LEFT JOIN performance perf
                ON art.artist_id = perf.artist_id
                LEFT JOIN event ev
                ON perf.event_id = ev.event_id
                LEFT JOIN location loc
                ON ev.location_id = loc.location_id
                LEFT JOIN festival fest
                ON loc.festival_id = fest.festival_id
                GROUP BY art.name
        ) as appearances_table_2
);
```

Με το 2ο SELECT βρίσκουμε τις συμμετοχές σε κάθε φεστιβάλ για κάθε καλιτέχνη. Τον πίνακα που δημιουργήθηκε τον χρησιμοποιούμε για να βρούμε τον μέγιστο αριθμό εμφανίσεων μείον το 5. Με το 1ο SELECT βρίσκουμε τους καλλιτέχνες που έχουν αριθμό εμφανίσεων κάτω από το όριο που υπολογίσαμε.

12)

```
SELECT fest.festival_name, ev.day, count(IF(st.role = 'technical', 1, NULL)) as security_staff, count(IF(st.role = 'helper', 1, NULL)) as helpers_staff
FROM staff st
INNER JOIN building build
ON build.building_id = st.building_id
INNER JOIN event ev
ON ev.building_id = build.building_id
INNER JOIN location loc
ON ev.location_id = loc.location_id
INNER JOIN festival fest
ON loc.festival_id = fest.festival_id
GROUP BY fest.festival_name, ev.day;
```

Με το GROUP BY βρίσκουμε τον αριθμό των security staff και helper staff, που δουλεύουν για κάθε ημέρα ενός φεστιβάλ.

13)

```
SELECT art.name, art.nickname, art.date_of_birth, art.social_media
FROM artist art
INNER JOIN performance perf
ON art.artist_id = perf.artist_id
INNER JOIN event ev
ON perf.event_id = ev.event_id
INNER JOIN location loc
ON ev.location_id = loc.location_id
GROUP BY art.name
HAVING count(DISTINCT loc.continent) >= 3;
```

Με το GROUP BY κατηγοριοποιούμε τα αποτελέσματα ανά καλλιτέχνη, και με το HAVING count(DISTINCT ...) βρίσκουμε αυτούς που έχουν πάει σε 3 τουλάχιστον διαφορετικές ηπείρους.

14)

```
WITH type as (
        SELECT gen.genre as genre, count(perf.performance_id) as appearances, loc.year as year
        FROM genre gen
        INNER JOIN artist art
        ON gen.artist_id = art.artist_id
        INNER JOIN performance perf
        ON art.artist_id = perf.artist_id
        INNER JOIN event ev
        ON perf.event_id = ev.event_id
        INNER JOIN location loc
        ON ev.location_id = loc.location_id
        GROUP BY loc.year, gen.genre
        HAVING appearances >= 3
)
SELECT type_1.*, type_2.*
FROM type type_1
INNER JOIN type type_2
ON (type_1.genre = type_2.genre) AND (type_1.appearances = type_2.appearances) AND (type_1.year = type_2.year - 1);
```

Με το table type βρίσκουμε τον αριθμό εμφανίσεων για κάθε μουσικό είδος ανά χρόνο. Κάνοντας ένα self join στο table type μπορούμε να βρούμε τα ξεχωριστά μουσικά είδη που έχουν τον ίδιο αριθμό εμφανίσεων για δύο συνεχόμενα χρόνια.

15)

```
SELECT vis.visitor_id, vis.visitor_name, vis.visitor_surname, art.name as artist_name, SUM(rat.rating_artist_performance) as total_artist_rating
FROM rating rat
INNER JOIN ticket tick
ON rat.rating_id = tick.rating_id
INNER JOIN event ev
ON ev.event_id = tick.event_id
INNER JOIN performance perf
ON perf.event_id = ev.event_id
INNER JOIN artist art
ON perf.artist_id = art.artist_id
INNER JOIN visitor vis
ON vis.visitor_id = tick.visitor_id
GROUP BY vis.visitor_id, art.name
ORDER BY total_artist_rating DESC, vis.visitor_id
LIMIT 5;
```

Με το SUM(...) βρίσκουμε τον συνολικό βαθμό των καλλιτεχνών και με το GROUP BY, υπολογίζουμε αυτό τον βαθμό ανά επισκέπτη και καλλιτέχνη. Με το LIMIT 5 και το ORDER BY βρίσκουμε τις 5 συνολικά υψηλότερες βαθμολογίες που έχει πάρει κάποιος καλλιτέχνης από έναν μόνο επισκέπτη.