Project Andante - Google CloudSQL Runbook

Authors: Christina Liu | July 2024

Primary Maintenance SQL Queries

SQL query to update performers' andante levels and sort

```sql
/* Drop the concert_counts_by_performers table if exists */
DROP TABLE IF EXISTS concert_counts_by_performers;

/* Create a new concert_counts_by_performers table to hold
  the results of counting concerts by performers */
CREATE TABLE concert_counts_by_performers (
  performer_id INT NOT NULL,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NOT NULL,
```

```
  school VARCHAR(255) NULL,
  grade INT NULL,
  concert_count INT NOT NULL,
  PRIMARY KEY(performer_id)
);

/* Populate concert_counts_by_performers table with results of counting
  concerts by performers */
INSERT INTO concert_counts_by_performers(performer_id, first_name, last_name, school, grade,
concert_count)
SELECT
   performer_id,
   first_name,
   last_name,
   school,
   grade,
   concert_count
FROM
   (SELECT
       performers.performer_id AS performer_id,
       performers.first_name AS first_name,
       performers.last_name AS last_name,
       performers.school AS school,
       performers.grade AS grade,
       ps_count.concert_count AS concert_count
   FROM
       performers
   INNER JOIN
     (SELECT
           cp_table.performer_id AS performer_id,
           COUNT(DISTINCT(concert_id)) AS concert_count
       FROM
           (SELECT
               jsontable.performer_ids AS performer_id,
               performances.concert_id AS concert_id
           FROM
               performances
           CROSS JOIN JSON_TABLE(CONCAT('["', REPLACE(performer_ids, ',', '","'), '"]'),
               '$[*]' COLUMNS (performer_ids VARCHAR(255) PATH '$')) jsontable
```

```
            ) AS cp_table
        GROUP BY
            cp_table.performer_id
      ) AS ps_count
    WHERE
        performers.performer_id = ps_count.performer_id
    ORDER BY
        concert_count DESC, grade DESC, first_name ASC, last_name ASC
    ) AS concert_count_results;

/* Update each performer in performers table with concert_count */
UPDATE
    performers p
INNER JOIN
    concert_counts_by_performers c
ON
    p.performer_id = c.performer_id
SET
  p.concert_count = c.concert_count;

/* Update performers' andante_level to Half Note Performer if they participated
  in at least 10 concerts */
UPDATE
    performers p
INNER JOIN
    concert_counts_by_performers c
ON
    p.performer_id = c.performer_id AND c.concert_count >= 10
SET
  p.andante_level = "Half Note Performer";

/* Update performers' andante_level to Quarter Note Performer if they participated
  in at least 5 concerts */
UPDATE
    performers p
INNER JOIN
    concert_counts_by_performers c
ON
    p.performer_id = c.performer_id AND c.concert_count >= 5 AND c.concert_count < 10
```

```
SET
  p.andante_level = "Quarter Note Performer";

/* Update performers' andante_level to Eighth Note Performer if they participated
  in at least 3 concerts */
UPDATE
   performers p
INNER JOIN
   concert_counts_by_performers c
ON
   p.performer_id = c.performer_id AND c.concert_count >= 3 AND c.concert_count < 5
SET
  p.andante_level = "Eighth Note Performer";

/* Update performers' andante_level to Sixteenth Note Performer if they participated
  in at least 1 concerts */
UPDATE
   performers p
INNER JOIN
   concert_counts_by_performers c
ON
   p.performer_id = c.performer_id AND c.concert_count >= 1 AND c.concert_count < 3
SET
  p.andante_level = "Sixteenth Note Performer";
```

SQL query to count number of performances

```
SELECT COUNT(*) FROM performances;
```

SQL query to count number of performers

```
SELECT COUNT(*) FROM performers;
```

SQL query to count number of concerts

```
SELECT COUNT(*) FROM concerts;
```

SQL query to count number of venues

```
SELECT COUNT(*) FROM venues;
```

Insert a performer into performers table

```
INSERT INTO performers(first_name, last_name, school, grade, instruments, bio)
VALUES("Christina", "Liu", "Lakeside School", 11, "Cellist/Guitarist", "bio to be updated");
```

Update a performer's data field in the performers table

```
UPDATE
   performers
SET
   bio = "bio was updated"
WHERE
   performer_id = 1;
```

Insert a venue into venues table

```
INSERT INTO venues(name, address, phone_number, website)
VALUES("Senior Living Center Name", "Senior Living Center Address", "Senior Living Center Phone
Number", "https://www.example-seniorlivingcenter-website.com");
```

Update a venue's data field in the venues table

```
UPDATE
   venues
```

```
SET
   description = "description was updated"
WHERE
   venue_id = 1;
```

Insert a concert into concerts table

```
INSERT INTO concerts(venue_id, year, month, day)
VALUES(1, 2022, 8, 19);
```

Update a venue’s data field in the concerts table

```
UPDATE
  concerts
SET
  description = "description was updated"
WHERE
  concert_id = 1;
```

Insert a performance into performances table

```
INSERT INTO performances(concert_id, performer_ids)
VALUES(1, '1,2');
```

Update a performer’s data field in the performers table

```
UPDATE
   performances
SET
   Performer_ids = '1,2,3'
WHERE
   performance_id = 1;
```

Useful SQL Queries

SQL query to count and sort performers by participated concert count, grade, name

```sql
SELECT
  performers.first_name AS performer_first_name,
  performers.last_name AS performer_last_name,
  performers.school AS performer_school,
  performers.grade AS performer_grade,
  ps_count.concert_count AS concert_count
FROM
  performers
INNER JOIN
  (SELECT
       cp_table.performer_id AS performer_id,
       COUNT(DISTINCT(concert_id)) AS concert_count
   FROM
       (SELECT
            jsontable.performer_ids AS performer_id,
            performances.concert_id AS concert_id
        FROM
            performances
        CROSS JOIN JSON_TABLE(CONCAT('["', REPLACE(performer_ids, ',', '","'), '"]'),
            '$[*]' COLUMNS (performer_ids VARCHAR(255) PATH '$')) jsontable
       ) AS cp_table
   GROUP BY
       cp_table.performer_id
  ) AS ps_count
WHERE
  performers.performer_id = ps_count.performer_id
ORDER BY
  concert_count DESC, performer_grade DESC, performer_first_name ASC, performer_last_name ASC;
```

Split performers in the performances table into multiple rows

```
SELECT performances.concert_id, jsontable.performer_ids
FROM performances
CROSS JOIN JSON_TABLE(CONCAT('["', REPLACE(performer_ids, ',', '","'), '"]'),
                      '$[*]' COLUMNS (performer_ids VARCHAR(255) PATH '$')) jsontable;
```

Add a new column into a table

```
ALTER TABLE performers
ADD COLUMN concert_count INT NULL
AFTER andante_level;
```

Move column to a specific position

```
ALTER TABLE performers MODIFY concert_count INT AFTER andante_level;
```

Create Tables

Create performers table

```
CREATE TABLE performers (
   performer_id INT NOT NULL AUTO_INCREMENT,
   first_name VARCHAR(255) NOT NULL,
   last_name VARCHAR(255) NOT NULL,
   middle_name VARCHAR(255) NULL,
   school VARCHAR(255) NULL,
   grade INT NULL,
   instruments VARCHAR(255) NULL,
   andante_level VARCHAR(255) NULL,
   concert_count INT NULL,
   bio TEXT NULL,
   PRIMARY KEY(performer_id)
);
```

Create venues table

```
CREATE TABLE `main_db`.venues (
   venue_id INT NOT NULL AUTO_INCREMENT,
   name VARCHAR(255) NOT NULL,
   address VARCHAR(1023) NOT NULL,
   phone_number VARCHAR(255) NULL,
   contact_email VARCHAR(255) NULL,
   website VARCHAR(255) NULL,
   description TEXT NULL,
   PRIMARY KEY(venue_id)
);
```

Create concerts table

```
CREATE TABLE `main_db`.concerts (
   concert_id INT NOT NULL AUTO_INCREMENT,
   venue_id INT NOT NULL,
   year INT NOT NULL,
   month INT NOT NULL,
   day INT NULL,
   description TEXT NULL,
   photos_link VARCHAR(2047) NULL,
   videos_link VARCHAR(2047) NULL,
   PRIMARY KEY(concert_id)
);
```

Create performs table

```
CREATE TABLE performances (
  performance_id INT NOT NULL AUTO_INCREMENT,
  concert_id INT NOT NULL,
  performer_ids VARCHAR(255) NOT NULL,
  performance_piece_name VARCHAR(255) NULL,
  performance_type VARCHAR(255) NULL,
  description TEXT NULL,
```

```
  PRIMARY KEY(performance_id)
);
```