Project Phase 2 Report
Due 28.04.2024

Name: İpek
Surname: Akkuş
ID: 30800

Part 1: Table Selection

For this section concert and venue tables are chosen as they are connected through VenueID, which is the primary key in Venue table and foreign key in Concert table. In order to create these 2 tables, these CREATE TABLE statements are used.

- **For Concert Table**

```
CREATE TABLE IF NOT EXISTS `mydb`.`Concert` (
 `ConcertId` INT NOT NULL,
 `Date` DATE NULL,
 `VenueId` INT NOT NULL,
 `BandId` INT NOT NULL,
 `Genre` VARCHAR(45) NOT NULL,
 `Time` VARCHAR(6) NULL,
 `Name` VARCHAR(45) NOT NULL,
 PRIMARY KEY (`ConcertId`),
 UNIQUE INDEX `ConcertId_UNIQUE` (`ConcertId` ASC) VISIBLE,
 INDEX `VenueId_idx` (`VenueId` ASC) VISIBLE,
 INDEX `GroupId_idx` (`BandId` ASC) VISIBLE,
 CONSTRAINT `VenueId`
  FOREIGN KEY (`VenueId`)
  REFERENCES `mydb`.`Venue` (`VenueId`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `BandId`
  FOREIGN KEY (`BandId`)
```

```
    REFERENCES `mydb`.`Band` (`BandId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

- **For Venue Table**

```
CREATE TABLE IF NOT EXISTS `mydb`.`Venue` (
 `VenueId` INT NOT NULL,
 `Name` VARCHAR(45) NOT NULL,
 `Country` VARCHAR(45) NOT NULL,
 `City` VARCHAR(45) NOT NULL,
 `Capacity` INT NOT NULL,
 `Address` VARCHAR(45) NOT NULL,
 PRIMARY KEY (`VenueId`),
 UNIQUE INDEX `VenueId_UNIQUE` (`VenueId` ASC) VISIBLE)
```

Part 2: Functional Dependencies and BCNF Conditions

BCNF (Boyce-Codd Normal Form) is a level of database normalization that eliminates redundancy by ensuring that every determinant in the table is a candidate key. A table is in BCNF if, for every one of its non-trivial functional dependencies X -> Y, X is a superkey. Boyce Codd Normal Form (BCNF) requires these 2 rules to be satisfied.

Rule 1: $No\ set\ valued\ attributes\ (only\ atomic\ values)$

Rule 2: $For\ all\ FDs\colon\ X \rightarrow Y\ \ The\ left\ hand\ side\ (X)\ should\ be\ a\ key\ or\ super\ key$

As there is no Set value type column in either of the two tables, there is no violation of BCNF according to the first rule. Below, Rule 2 will be checked for two different tables, and trivial functional dependencies will be ignored since there is no violation of Rule 2 in terms of trivial FDs.

2.1 Concert Table

The functional dependencies within the Concert Table are listed below. When the attributes **VenueId, Date, and Time** are combined, they establish a functional dependency because it is not possible to have multiple concerts at the same venue at the same date and time. This combination uniquely determines any other information related to a specific event scheduled in the database. The reason why the Name attribute does not establish any functional dependencies is entirely due to the database design. The name of a concert does not uniquely determine any other attributes in the database; it serves primarily as a descriptive label without implying unique identification or constraints on other data fields.

$$\begin{aligned} FD = \{ & ConcertId \ -> \ VenueId\ BandId\ Date\ Name\ Time\ Genre \\ & VenueId\ Date\ Time \ -> \ ConcertId\ BandId\ Name\ Genre \} \end{aligned}$$

Since **ConcertId** is the **primary key**, it defines functional dependencies for all attributes. Additionally, when the **VenueId, Date and Time** attributes come together to form the Left Hand Side, they define functional dependencies for all attributes, making them a **Candidate Key**.

In conclusion, there is no violation of the second rule of BCNF, as the attributes on the left-hand side are either keys or superkeys. The Concert Table has been designed in accordance with Boyce-Codd Normal Form. Therefore, there is **no need for decomposition**.

2.2 Venue Table

The functional dependencies within the Venue Table are listed below. The Address attribute contains the complete address of a venue, and therefore, it establishes a functional dependency for the Country and City attributes.

$$\begin{aligned} FD = \{ & VenueId \ -> \ Name\ Address\ Country\ City\ Capacity \\ & Address\ Name \ -> \ VenueId\ Country\ City\ Capacity \\ & Address \ -> \ Country\ City \} \end{aligned}$$

Since **VenueId** is the **primary key**, it defines functional dependencies for all attributes by default. Additionally, when the **Address and Name** attributes come together to form the Left Hand Side, they define functional dependencies for all attributes, making them a **Candidate Key**. However when Only Address itself forms Left Hand Side, it does not establish dependencies for other attributes, which means Address is not a key.

Consequently, this situation violates Rule 2 of the Boyce-Codd Normal Form (BCNF), which states that for a table to be in BCNF, every determinant must be a candidate key. Since Address is not a candidate key but determines some attributes, it breaches this rule. To achieve a design that conforms to Boyce-Codd Normal Form (BCNF), **decomposition is necessary.** This involves splitting the table into two or more tables in such a way that each smaller table adheres to BCNF rules. Specifically, this means ensuring that in each of the resultant tables, every determinant (the attribute or set of attributes on the left-hand side of a functional dependency) is a candidate key. This process helps eliminate the dependencies where a non-candidate **key determines other attributes**, thereby resolving violations of BCNF.

Newly decomposed Venue Table and Address Table are created as below.

- **Venue Table**

```
CREATE TABLE IF NOT EXISTS `mydb`.`Venue` (
  `VenueId` INT NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Capacity` INT NOT NULL,
  `AddressId` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`VenueId`),
  UNIQUE INDEX `VenueId_UNIQUE` (`VenueId` ASC) VISIBLE)
```

- **Address Table**

```
CREATE TABLE IF NOT EXISTS `mydb`.`Address` (
  `AddressId` INT NOT NULL,
  `Address` VARCHAR(255) NOT NULL,
  `Country` VARCHAR(45) NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`AddressId`),
  UNIQUE INDEX `Address_UNIQUE` (`Address` ASC) VISIBLE)
```

Part 3: “INSERT INTO” Statements for Tables

Previously created Venue table is decomposed to satisfy the second rule of BCNF form, as explained above. 10 rows are added for both Address table and Venue table. The “insert into” statements can be found below. These statements are realistic and added by considering the edge cases such as allowing 2 different concert halls at the same locations (i.e There exists 2 performance halls in Zorlu Center, these have the same AddressId’s whereas their name is differentiated by adding “Yan Salon” in the Name property of Venue Table). Minimum of 10 data rows are added in order to clearly show

- **For Address Table after Decomposition**

```
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (01, 'Huzur, Uniq İstanbul, Maslak Ayazağa Cd. No:4, 34485 Sarıyer/İstanbul', 'İstanbul ',
'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (02, 'Cihannüma, Hasfırın Cd. No:26, 34353 Beşiktaş/İstanbul', 'İstanbul ', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (03, 'Caferağa, Namlı Market Yanı, Neşet Ömer Sk. 3C, 34710 Kadıköy/İstanbul', 'İstanbul
', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (04, 'Sinanpaşa, Ortabahçe Cd. no 23/4, 34022 Beşiktaş/İstanbul', 'İstanbul ', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (05, 'Levazım, Koru Sok. No:2, 34340 Beşiktaş/İstanbul', 'İstanbul ', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (06, 'Harbiye, Kadırgalar Cd. No:4, 34367 Şişli/İstanbul', 'İstanbul ', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (07, 'Vişnezade, Dolmabahçe Cd. No:1, 34357 Beşiktaş/İstanbul', 'İstanbul ', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (08, 'Mavişehir mah. Caher dudayev blv. A apt. No:40, D:304, 35590 Karşıyaka/İzmir',
'İzmir', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (09, 'Bahçelievler, Azerbaycan Cd. No:41, 06490 Çankaya/Ankara', 'Ankara', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (10, 'Pamucak Sahili, Selçuk/İzmir.', 'İzmir', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
```

```
VALUES (11, 'Kavaklıdere, Kızılırmak Cd. No:14, 06420 Çankaya/Ankara', 'Ankara', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (12, 'Mimar Sinan, Şair Eşref Blv. No.50, 35220 Konak/İzmir', 'İzmir', 'Turkey');
INSERT INTO `mydb`.`Address` (`AddressId`, `Address`, `City`, `Country`)
VALUES (13, 'Harbiye, Taşkışla Cd. No:8, 34367 Şişli/İstanbul', 'İstanbul ', 'Turkey');
```

- **For Venue Table after Decomposition**

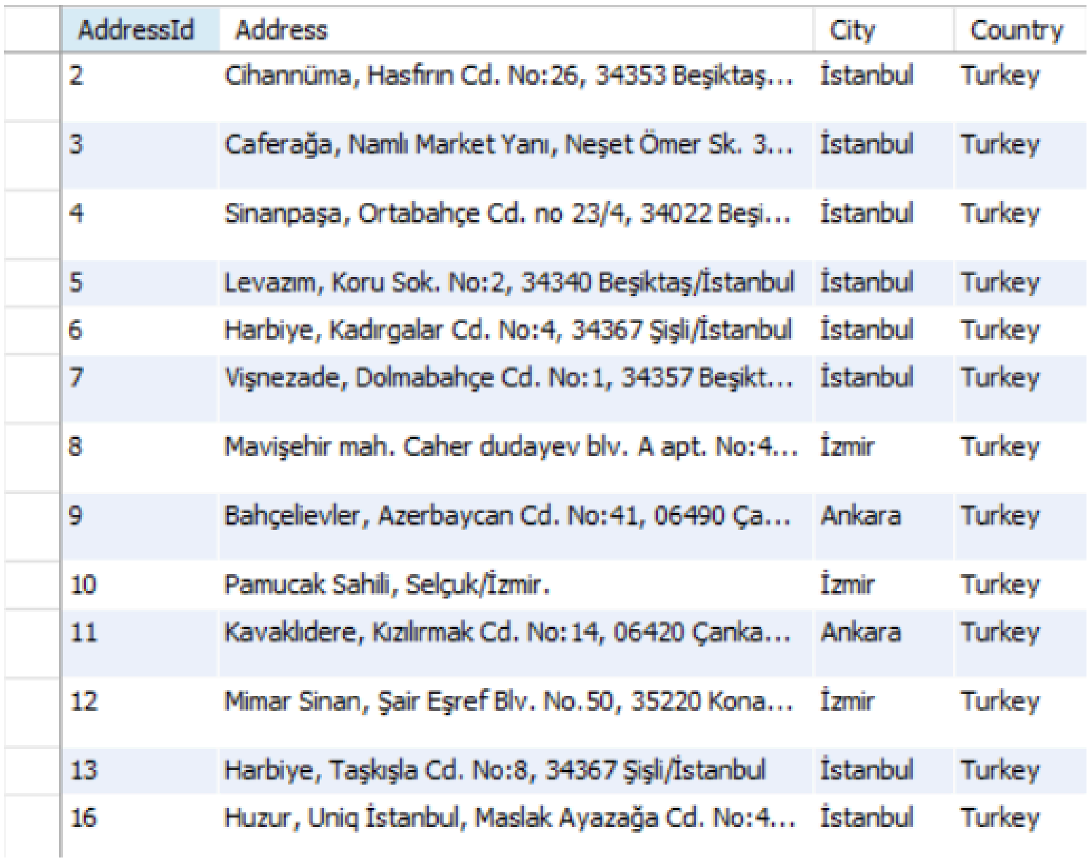
```
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (01, 'Maximum Uniq Hall', 01, 1156);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (02, 'IF Performance Hall ', 02, 4500);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (03, 'Dorock XL', 03, 1000);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (04, 'Dorock XL', 04, 2000);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (05, 'Zorlu PSM', 05, 3200);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (06, 'KüçükÇiftlik Park', 06, 10000);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (07, 'BJK Tüpraş Stadyumu', 07, 42590);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (08, 'SoldOut Performance Hall', 08, 1200);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (09, 'Milyon Performance Hall', 09, 2500);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (10, 'What A Fest Festival Arena', 10, 13600);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (11, 'Jolly Joker Ankara', 11, 1600);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (12, 'Kültürpark Atatürk Outdoor Theatre', 12, 3150);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (13, 'Harbiye Outdoor Theatre', 13, 4532);
INSERT INTO `mydb`.`Venue` (`VenueId`, `Name`, `AddressId`, `Capacity`)
VALUES (14, 'Zorlu PSM Yan Salon', 05, 859);
```

Part 4: Displaying Selected Tables

Commands that are used to display all the rows of the selected two tables are provided below.

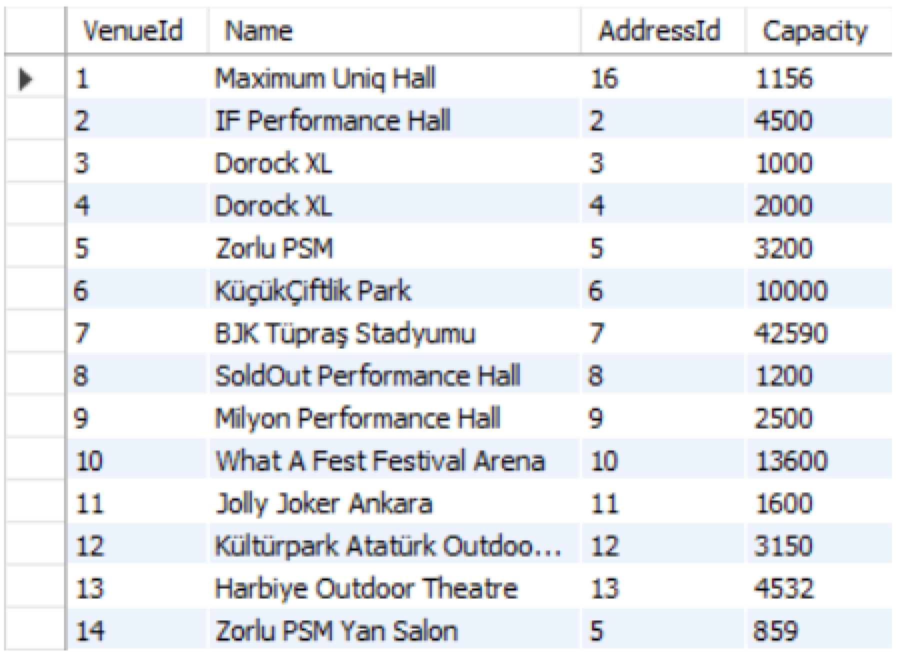
```
SELECT * FROM mydb.Address;
SELECT * FROM mydb.Venue;
```

After running each command, full contents of each table are displayed within the MySQL interface as in Figure 1 and Figure 2, respectively.



AddressId	Address	City	Country
2	Cihannüma, Hasfırın Cd. No:26, 34353 Beşiktaş...	İstanbul	Turkey
3	Caferağa, Namlı Market Yanı, Neşet Ömer Sk. 3...	İstanbul	Turkey
4	Sinanpaşa, Ortabahçe Cd. no 23/4, 34022 Beşi...	İstanbul	Turkey
5	Levazım, Koru Sok. No:2, 34340 Beşiktaş/İstanbul	İstanbul	Turkey
6	Harbiye, Kadırgalar Cd. No:4, 34367 Şişli/İstanbul	İstanbul	Turkey
7	Vişnezade, Dolmabahçe Cd. No:1, 34357 Beşikt...	İstanbul	Turkey
8	Mavişehir mah. Caher dudayev blv. A apt. No:4...	İzmir	Turkey
9	Bahçelievler, Azerbaycan Cd. No:41, 06490 Ça...	Ankara	Turkey
10	Pamucak Sahili, Selçuk/İzmir.	İzmir	Turkey
11	Kavaklıdere, Kızılırmak Cd. No:14, 06420 Çanka...	Ankara	Turkey
12	Mimar Sinan, Şair Eşref Blv. No.50, 35220 Kona...	İzmir	Turkey
13	Harbiye, Taşkışla Cd. No:8, 34367 Şişli/İstanbul	İstanbul	Turkey
16	Huzur, Uniq İstanbul, Maslak Ayazağa Cd. No:4...	İstanbul	Turkey

Figure 1. Address Table



VenueId	Name	AddressId	Capacity
1	Maximum Uniq Hall	16	1156
2	IF Performance Hall	2	4500
3	Dorock XL	3	1000
4	Dorock XL	4	2000
5	Zorlu PSM	5	3200
6	KüçükÇiftlik Park	6	10000
7	BJK Tüpraş Stadyumu	7	42590
8	SoldOut Performance Hall	8	1200
9	Milyon Performance Hall	9	2500
10	What A Fest Festival Arena	10	13600
11	Jolly Joker Ankara	11	1600
12	Kültürpark Atatürk Outdoo...	12	3150
13	Harbiye Outdoor Theatre	13	4532
14	Zorlu PSM Yan Salon	5	859

Figure 2. Venue Table

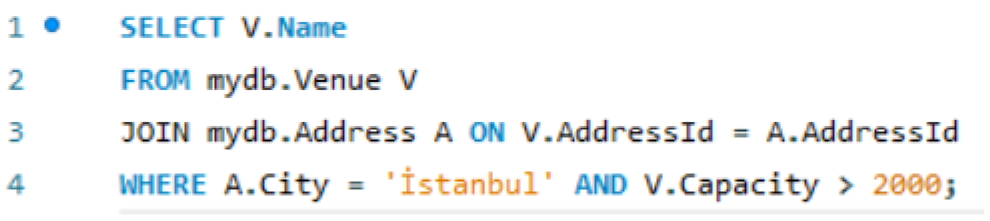
Part 5: Query in English, requires "Join"

Retrieve the names of venues that are located in the city of İstanbul and have a capacity greater than 2000. Relational algebraic expression for this query is as follows.

$$\pi_{Name}(\sigma_{City="stanbul" \ AND \ Capacity>2000}(Address \bowtie_{AddressId} Venue))$$

Part 6: SQL Version of the Query in Part 5

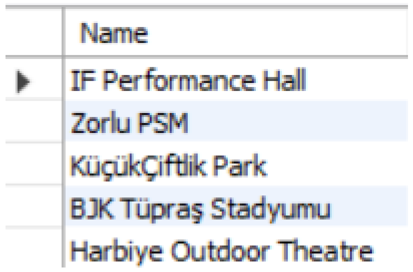
SQL command for given query in part 5 is as follows.



```sql
SELECT V.Name
FROM mydb.Venue V
JOIN mydb.Address A ON V.AddressId = A.AddressId
WHERE A.City = 'İstanbul' AND V.Capacity > 2000;
```

Figure 3 . SQL for Query in Part 5

The output for given SQL is provided in *Figure 4* below.



Name
IF Performance Hall
Zorlu PSM
KüçükÇiftlik Park
BJK Tüpraş Stadyumu
Harbiye Outdoor Theatre

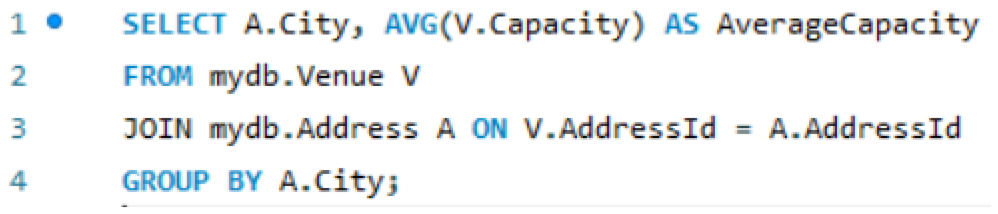
Figure 4. Venues in İstanbul that exceeds the capacity of 2000.

Part 7 : Query in English, requires “Group By” and includes Statistical Operator

Calculate the average capacity of venues in each city.

Part 8: SQL Version of the Query in Part 7

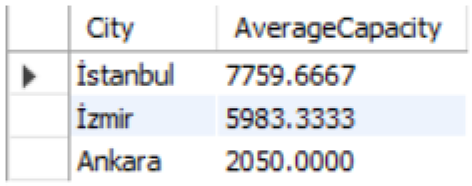
SQL command for given query in part 5 is as follows.



```
SELECT A.City, AVG(V.Capacity) AS AverageCapacity
FROM mydb.Venue V
JOIN mydb.Address A ON V.AddressId = A.AddressId
GROUP BY A.City;
```

Figure 5 . SQL for Query in Part 7

The output for given SQL is provided in *Figure 4* below.

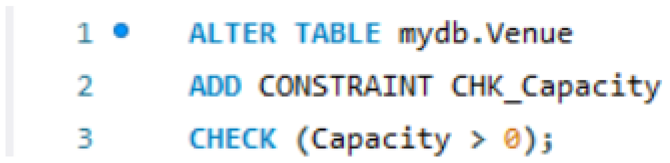


City	AverageCapacity
İstanbul	7759.6667
İzmir	5983.3333
Ankara	2050.0000

Figure 6. Average capacities of venues in each city.

Part 9: Adding a Check Constraint and Validation

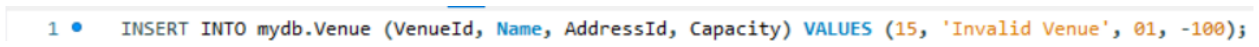
A check constraint to the Venue table will be added to ensure that the Capacity of a Venue is always greater than zero. This is a realistic constraint because a venue should not have a capacity of zero or a negative number. By ensuring that the venue capacity is greater than zero, the creation of venues that cannot accommodate any attendees is prevented. Here is the SQL command for this constraint.



```
ALTER TABLE mydb.Venue
ADD CONSTRAINT CHK_Capacity
CHECK (Capacity > 0);
```

Figure 7 . SQL to add given check constraint

In order to validate the check constraint, a case that it needs to alert is tried, which is provided below..

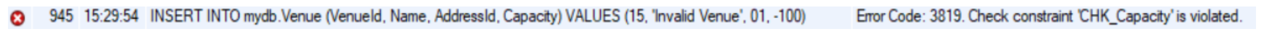


```
INSERT INTO mydb.Venue (VenueId, Name, AddressId, Capacity) VALUES (15, 'Invalid Venue', 01, -100);
```

Figure 8 . SQL to validate the check constraint

This insert statement should fail with an error stating that the check constraint **CHK_Capacity** was violated because the venue capacity is indicated as negative value , -100, which is not proper for the constraint and indicates the logically impossible case.

Error message that MySQL generated can be found in *Figure 5* below.



```
945  15:29:54  INSERT INTO mydb.Venue (VenueId, Name, AddressId, Capacity) VALUES (15, 'Invalid Venue', 01, -100)    Error Code: 3819. Check constraint 'CHK_Capacity' is violated.
```

Figure 9 . Error message that SQL generated