Main Diagram

7-05-2025 DbSchema.com © DATE Wise Coders

Layouts

Tables

MusicGenre

Artist
- Artist_ID
- Real_Name t
- Stage_Name t
- Birthdate d
- Website t
- Instagram_Profile t
- Image t
- Image_Description t

ResaleBuyerQueue
- Queue_ID #
- Buyer_ID
- Performance_ID
- Category t
- Requested_At d
- Status t

MusicGenre
- Genre_ID
- Name t
- Subgenre t

Band
- Band_ID
- Name t
- Formation_Date d
- Website t
- Instagram_Profile t

ArtistGenre
- Artist_ID
- Genre_ID

Band_Member
- Band_ID
- Artist_ID

Performance
- Performance_ID
- Event_ID
- Artist_ID
- Band_ID
- Type t
- Start_Time d
- Duration #

EntityImage
- Image_ID #
- Entity_Type t
- Entity_ID #
- Image_Description t
- Image_Data ~

Location

StaffAssignment
- Staff_ID
- Event_ID

Location
- Location_ID
- Address t
- Coordinates t
- City t
- Country t
- Continent t
- Image t
- Image_Description t

Stage
- Stage_ID
- Name t
- Description t
- Max_Capacity #
- Technical_Equipment t
- Image t
- Image_Description t

Festival
- Festival_ID
- Year #
- Start_Date d
- End_Date d
- Image t
- Image_Description t
- Location_ID
- Status t

Visitor

TicketResaleQueue
- Queue_ID #
- Ticket_ID
- Queue_Type t
- Queue_Timestamp d

ResaleTicket
- ResaleTicket_ID #
- Ticket_ID
- Seller_ID
- Listed_At d
- Status t

Event
- Event_ID
- Festival_ID
- Stage_ID
- Start_Time d
- End_Time d
- Status t

Staff
- Staff_ID
- Name t
- Age #
- Role t
- Experience_Level t

Visitor
- Visitor_ID
- First_Name t
- Last_Name t
- Contact_Info t
- Age #

Evaluation
- Evaluation_ID #
- Visitor_ID
- Performance_ID
- Evaluation_Date d
- Artist_Performance #
- Sound_Lighting #
- Stage_Presence #
- Organization #
- Overall_Impression #

Ticket
- Ticket_ID
- Performance_ID
- Visitor_ID
- Category t
- Purchase_Date d
- Cost #
- Payment_Method t
- EAN131_Code #
- Activated b

Main Diagram

Table Artist

Idx	Name	Data Type
* Pk	Artist_ID	INT AUTO_INCREMENT
*	Real_Name	VARCHAR(100)
	Stage_Name	VARCHAR(100)
	Birthdate	DATE
	Website	VARCHAR(255)
	Instagram_Profile	VARCHAR(255)
	Image	TEXT
	Image_Description	TEXT

Indexes

Type	Name	On
Pk	pk_artist	Artist_ID

Options

```
ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table ArtistGenre

Idx	Name	Data Type
* Pk	Artist_ID	INT
* Pk	Genre_ID	INT

Indexes

Type	Name	On
Pk	pk_artistgenre	Artist_ID, Genre_ID
	Genre_ID	Genre_ID

Foreign Keys

Type	Name	On
	ArtistGenre_ibfk_1 (Artist_ID) ref Artist (Artist_ID)	
	ArtistGenre_ibfk_2 (Genre_ID) ref MusicGenre (Genre_ID)	

Options

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Band

Idx	Name	Data Type
* Pk	Band_ID	INT AUTO_INCREMENT
*	Name	VARCHAR(100)
	Formation_Date	DATE
	Website	VARCHAR(255)
	Instagram_Profile	VARCHAR(255)

Indexes

Type	Name	On
Pk	pk_band	Band_ID

Options

```
ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Band_Member

Idx	Name	Data Type
* Pk	Band_ID	INT

Table Band_Member

* Pk	Artist_ID	INT

Indexes

Type	Name	On
Pk	pk_band_member	Band_ID, Artist_ID
	Artist_ID	Artist_ID

Foreign Keys

Type	Name	On
	Band_Member_ibfk_1 (Band_ID) ref Band (Band_ID)	
	Band_Member_ibfk_2 (Artist_ID) ref Artist (Artist_ID)	

Options

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table EntityImage

Idx	Name	Data Type
* Pk	Image_ID	INT AUTO_INCREMENT
*	Entity_Type	ENUM('Festival','Artist','Band','Stage','Equipment')
*	Entity_ID	INT
	Image_Description	VARCHAR(255)
*	Image_Data	LONGBLOB

Indexes

Type	Name	On
Pk	pk_entityimage	Image_ID

Options

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Evaluation

Idx	Name	Data Type
* Pk	Evaluation_ID	INT AUTO_INCREMENT
* Unq	Visitor_ID	INT
* Unq	Performance_ID	INT
*	Evaluation_Date	DATETIME DEFAULT current_timestamp()
*	Artist_Performance	TINYINT
*	Sound_Lighting	TINYINT
*	Stage_Presence	TINYINT
*	Organization	TINYINT
*	Overall_Impression	TINYINT

Indexes

Type	Name	On
Pk	pk_evaluation	Evaluation_ID
Unq	Visitor_ID	Visitor_ID, Performance_ID
	idx_evaluation_visitor	Visitor_ID
	idx_evaluation_performance	Performance_ID

Foreign Keys

Type	Name	On
	Evaluation_ibfk_1 (Visitor_ID) ref Visitor (Visitor_ID)	
	Evaluation_ibfk_2 (Performance_ID) ref Performance (Performance_ID)	

Constraints

	Name	Definition

Table Evaluation

	cns_evaluation_artist_performance	`Artist_Performance` between 1 and 3
	cns_evaluation_sound_lighting	`Sound_Lighting` between 1 and 3
	cns_evaluation_stage_presence	`Stage_Presence` between 1 and 3
	cns_evaluation_organization	`Organization` between 1 and 3
	cns_evaluation_overall_impression	`Overall_Impression` between 1 and 3

Triggers

	Name	Definition
	check_evaluation_ticket_activation	

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Evaluation FOR EACH ROW BEGIN
  DECLARE ticket_count INT;
  SELECT COUNT(*) INTO ticket_count
  FROM Ticket t
  JOIN Performance p ON t.Performance_ID = p.Performance_ID
  WHERE t.Visitor_ID = NEW.Visitor_ID
    AND t.Performance_ID = NEW.Performance_ID
    AND t.Activated = 1;
  IF ticket_count = 0 THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Evaluation not allowed: visitor must have an activated ticket for the performance.';
  END IF;
END
```

Options

```
ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Event

Idx	Name	Data Type
* Pk	Event_ID	INT AUTO_INCREMENT
* Idx	Festival_ID	INT
* Unq	Stage_ID	INT
* Unq	Start_Time	DATETIME
*	End_Time	DATETIME
*	Status	ENUM('Scheduled','Ongoing','Completed') DEFAULT 'Scheduled'

Indexes

Type	Name	On
Pk	pk_event	Event_ID
Unq	Stage_ID	Stage_ID, Start_Time
	idx_event_festival	Festival_ID
	idx_event_stage	Stage_ID

Foreign Keys

Type	Name	On
	Event_ibfk_1 (Festival_ID) ref Festival (Festival_ID)	
	Event_ibfk_2 (Stage_ID) ref Stage (Stage_ID)	

Constraints

	Name	Definition
	CONSTRAINT_1	`End_Time` > `Start_Time`
	CONSTRAINT_2	`Status` <> 'Canceled'

Triggers

	Name	Definition
	prevent_event_overlap	

Table Event

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Event FOR EACH ROW BEGIN
  DECLARE overlap_count INT;
  SELECT COUNT(*) INTO overlap_count
  FROM Event
  WHERE Stage_ID = NEW.Stage_ID
    AND Festival_ID = NEW.Festival_ID
    AND DATE(Start_Time) = DATE(NEW.Start_Time)
    AND (
      NEW.Start_Time < End_Time AND
      NEW.End_Time > Start_Time
    );
  IF overlap_count > 0 THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Error: Overlapping event detected on the same stage.';
  END IF;
END
```

prevent_event_overlap_update

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE UPDATE ON Event FOR EACH ROW BEGIN
  DECLARE overlap_count INT;
  SELECT COUNT(*) INTO overlap_count
  FROM Event
  WHERE Stage_ID = NEW.Stage_ID
    AND Festival_ID = NEW.Festival_ID
    AND Event_ID != NEW.Event_ID
    AND (
      NEW.Start_Time < End_Time AND
      NEW.End_Time > Start_Time
    );
  IF overlap_count > 0 THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Error: Overlapping event detected on the same stage.';
  END IF;
END
```

Options

```
ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Festival

Idx	Name	Data Type
* Pk	Festival_ID	INT AUTO_INCREMENT
*	Year	YEAR(4)
*	Start_Date	DATE
*	End_Date	DATE
	Image	TEXT
	Image_Description	TEXT
* Idx	Location_ID	INT
*	Status	ENUM('Scheduled','Ongoing','Completed') DEFAULT 'Scheduled'

Indexes

Type	Name	On
Pk	pk_festival	Festival_ID
	Location_ID	Location_ID

Foreign Keys

Type	Name	On
	Festival_ibfk_1 (Location_ID) ref Location (Location_ID)	

Constraints

	Name	Definition
	CONSTRAINT_1	`End_Date` > `Start_Date`
	CONSTRAINT_2	`Status` <> 'Canceled'

Options

```
ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Location

Idx	Name	Data Type
* Pk	Location_ID	INT AUTO_INCREMENT
*	Address	VARCHAR(255)
	Coordinates	VARCHAR(100)
*	City	VARCHAR(100)
*	Country	VARCHAR(100)
	Continent	VARCHAR(100)
	Image	TEXT
	Image_Description	TEXT

Indexes

Type	Name	On
Pk	pk_location	Location_ID

Options

```
ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table MusicGenre

Idx	Name	Data Type
* Pk	Genre_ID	INT
	Name	TEXT
	Subgenre	TEXT

Indexes

Type	Name	On
Pk	pk_musicgenre	Genre_ID

Options

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Performance

Idx	Name	Data Type
* Pk	Performance_ID	INT AUTO_INCREMENT
* Idx	Event_ID	INT
Idx	Artist_ID	INT
Idx	Band_ID	INT
*	Type	ENUM('Warm Up','Headline','Special Guest')
*	Start_Time	DATETIME
*	Duration	INT

Indexes

Type	Name	On
Pk	pk_performance	Performance_ID
	idx_performance_artist	Artist_ID
	idx_performance_band	Band_ID
	idx_performance_event	Event_ID

Foreign Keys

Type	Name	On
	Performance_ibfk_1 (Event_ID) ref Event (Event_ID)	
	Performance_ibfk_2 (Artist_ID) ref Artist (Artist_ID)	
	Performance_ibfk_3 (Band_ID) ref Band (Band_ID)	

Constraints

	Name	Definition

Table Performance

	cns_performance_duration	`Duration` > 0 and `Duration` <= 180
	CONSTRAINT_1	`Artist_ID` is not null and `Band_ID` is null or `Artist_ID` is null and `Band_ID` is not null

Triggers

	Name	Definition
	check_artist_consecutive_years	

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Performance FOR EACH ROW BEGIN
  DECLARE year INT;
  DECLARE consecutive_years INT;
  IF NEW.Artist_ID IS NOT NULL THEN

    SELECT f.Year INTO year
    FROM Event e
    JOIN Festival f ON e.Festival_ID = f.Festival_ID
    WHERE e.Event_ID = NEW.Event_ID;

    SELECT COUNT(DISTINCT f.Year) INTO consecutive_years
    FROM Performance p
    JOIN Event e ON p.Event_ID = e.Event_ID
    JOIN Festival f ON e.Festival_ID = f.Festival_ID
    WHERE p.Artist_ID = NEW.Artist_ID
      AND f.Year BETWEEN year - 2 AND year;
    IF consecutive_years > 3 THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Artist cannot perform more than 3 consecutive years.';
    END IF;
  END IF;
END
```

	check_band_consecutive_years

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Performance FOR EACH ROW BEGIN
  DECLARE current_year INT;
  DECLARE consecutive_years INT;
  IF NEW.Band_ID IS NOT NULL THEN

    SELECT f.Year INTO current_year
    FROM Event e
    JOIN Festival f ON e.Festival_ID = f.Festival_ID
    WHERE e.Event_ID = NEW.Event_ID;

    SELECT COUNT(DISTINCT f.Year) INTO consecutive_years
    FROM Performance p
    JOIN Event e ON p.Event_ID = e.Event_ID
    JOIN Festival f ON e.Festival_ID = f.Festival_ID
    WHERE p.Band_ID = NEW.Band_ID
      AND f.Year BETWEEN current_year - 2 AND current_year;
    IF consecutive_years >= 3 THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Band cannot perform more than 3 consecutive years.';
    END IF;
  END IF;
END
```

	check_performance_break

Table Performance

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Performance FOR EACH ROW BEGIN
  DECLARE previous_end_time DATETIME;
  DECLARE break_duration INT;
  DECLARE conflicting_count INT;

  SELECT COUNT(*) INTO conflicting_count
  FROM Performance
  WHERE Event_ID = NEW.Event_ID
    AND DATE(Start_Time) = DATE(NEW.Start_Time)
    AND (

      (NEW.Start_Time BETWEEN Start_Time AND (Start_Time + INTERVAL Duration MINUTE - INTERVAL 1 SECOND))
      OR ((NEW.Start_Time + INTERVAL NEW.Duration MINUTE - INTERVAL 1 SECOND) BETWEEN Start_Time AND (Start_Time + INTERVAL
Duration MINUTE - INTERVAL 1 SECOND))
      OR (Start_Time BETWEEN NEW.Start_Time AND (NEW.Start_Time + INTERVAL NEW.Duration MINUTE - INTERVAL 1 SECOND))
    );
  IF conflicting_count > 0 THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Performance time overlaps with an existing performance.';
  END IF;
  SELECT MAX(Start_Time + INTERVAL Duration MINUTE) INTO previous_end_time
  FROM Performance
  WHERE Event_ID = NEW.Event_ID
    AND Start_Time < NEW.Start_Time
    AND DATE(Start_Time) = DATE(NEW.Start_Time);
  IF previous_end_time IS NOT NULL THEN
    SET break_duration = TIMESTAMPDIFF(MINUTE, previous_end_time, NEW.Start_Time);

    IF break_duration < 5 OR break_duration > 30 THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Break between performances must be between 5 and 30 minutes.';
    END IF;
  END IF;
END
```

prevent_artist_overlap

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Performance FOR EACH ROW BEGIN
  DECLARE overlap_count INT;
  DECLARE festival_id INT;
  IF NEW.Artist_ID IS NOT NULL THEN
    SELECT e.Festival_ID INTO festival_id
    FROM Event e
    WHERE e.Event_ID = NEW.Event_ID;
    SELECT COUNT(*) INTO overlap_count
    FROM Performance p
    JOIN Event e ON p.Event_ID = e.Event_ID
    WHERE p.Artist_ID = NEW.Artist_ID
      AND e.Festival_ID = festival_id
      AND (
        NEW.Start_Time < ADDTIME(p.Start_Time, SEC_TO_TIME(p.Duration * 60)) AND
        ADDTIME(NEW.Start_Time, SEC_TO_TIME(NEW.Duration * 60)) > p.Start_Time
      );
    IF overlap_count > 0 THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Artist has overlapping performance in the same festival.';
    END IF;
  END IF;
END
```

prevent_band_overlap

Table Performance

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Performance FOR EACH ROW BEGIN
  DECLARE overlap_count INT;
  DECLARE festival_id INT;
  IF NEW.Band_ID IS NOT NULL THEN
    SELECT e.Festival_ID INTO festival_id
    FROM Event e
    WHERE e.Event_ID = NEW.Event_ID;
    SELECT COUNT(*) INTO overlap_count
    FROM Performance p
    JOIN Event e ON p.Event_ID = e.Event_ID
    WHERE p.Band_ID = NEW.Band_ID
      AND e.Festival_ID = festival_id
      AND (
        NEW.Start_Time < ADDTIME(p.Start_Time, SEC_TO_TIME(p.Duration * 60)) AND
        ADDTIME(NEW.Start_Time, SEC_TO_TIME(NEW.Duration * 60)) > p.Start_Time
      );
    IF overlap_count > 0 THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Band has overlapping performance in the same festival.';
    END IF;
  END IF;
END
```

Options

```
ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table ResaleBuyerQueue

Idx	Name	Data Type
* Pk	Queue_ID	INT AUTO_INCREMENT
* Idx	Buyer_ID	INT
* Idx	Performance_ID	INT
*	Category	ENUM('General','VIP','Backstage')
*	Requested_At	DATETIME DEFAULT current_timestamp()
*	Status	ENUM('Waiting','Matched','Cancelled') DEFAULT 'Waiting'

Indexes

Type	Name	On
Pk	pk_resalebuyerqueue	Queue_ID
	idx_resalebuyerqueue_buyer	Buyer_ID
	idx_resalebuyerqueue_performance	Performance_ID

Foreign Keys

Type	Name	On
	ResaleBuyerQueue_ibfk_1 (Buyer_ID) ref Visitor (Visitor_ID)	
	ResaleBuyerQueue_ibfk_2 (Performance_ID) ref Performance (Performance_ID)	

Options

```
ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table ResaleTicket

Idx	Name	Data Type
* Pk	ResaleTicket_ID	INT AUTO_INCREMENT
* Idx	Ticket_ID	INT
* Idx	Seller_ID	INT
*	Listed_At	DATETIME DEFAULT current_timestamp()
*	Status	ENUM('Available','Sold','Withdrawn') DEFAULT 'Available'

Indexes

Type	Name	On
Pk	pk_resaleticket	ResaleTicket_ID
	idx_resaleticket_ticket	Ticket_ID
	idx_resaleticket_seller	Seller_ID

Table ResaleTicket

Foreign Keys

Type	Name	On
	ResaleTicket_ibfk_1 (Ticket_ID) ref Ticket (Ticket_ID)	
	ResaleTicket_ibfk_2 (Seller_ID) ref Visitor (Visitor_ID)	

Options

ENGINE=InnoDB AUTO_INCREMENT=51 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Table Staff

Idx	Name	Data Type
* Pk	Staff_ID	INT AUTO_INCREMENT
*	Name	VARCHAR(100)
*	Age	INT
*	Role	ENUM('Technical','Security','Auxiliary')
*	Experience_Level	ENUM('Intern','Beginner','Intermediate','Experienced','Expert')

Indexes

Type	Name	On
Pk	pk_staff	Staff_ID

Constraints

Name	Definition
cns_staff_age	`Age` >= 18

Options

ENGINE=InnoDB AUTO_INCREMENT=301 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Table StaffAssignment

Idx	Name	Data Type
* Pk	Staff_ID	INT
* Pk	Event_ID	INT

Indexes

Type	Name	On
Pk	pk_staffassignment	Staff_ID, Event_ID
	idx_staffassignment_staff	Staff_ID
	idx_staffassignment_event	Event_ID

Foreign Keys

Type	Name	On
	StaffAssignment_ibfk_1 (Staff_ID) ref Staff (Staff_ID)	
	StaffAssignment_ibfk_2 (Event_ID) ref Event (Event_ID)	

Options

ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

Table Stage

Idx	Name	Data Type
* Pk	Stage_ID	INT AUTO_INCREMENT
*	Name	VARCHAR(100)
	Description	TEXT
*	Max_Capacity	INT
	Technical_Equipment	TEXT
	Image	TEXT

Table Stage

	Image_Description	TEXT

Indexes

Type	Name	On
Pk	pk_stage	Stage_ID

Constraints

	Name	Definition
	cns_stage_max_capacity	`Max_Capacity` > 0

Options

```
ENGINE=InnoDB AUTO_INCREMENT=61 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Ticket

Idx	Name	Data Type
* Pk	Ticket_ID	INT AUTO_INCREMENT
* Unq	Performance_ID	INT
* Unq	Visitor_ID	INT
*	Category	ENUM('General','VIP','Backstage')
* Unq	Purchase_Date	DATE
*	Cost	DECIMAL(10,2)
*	Payment_Method	ENUM('Credit Card','Debit Card','Bank Transfer')
* Unq	EAN131_Code	BIGINT
	Activated	BOOLEAN DEFAULT false

Indexes

Type	Name	On
Pk	pk_ticket	Ticket_ID
Unq	EAN131_Code	EAN131_Code
Unq	Visitor_ID	Visitor_ID, Performance_ID, Purchase_Date
	idx_ticket_performance	Performance_ID
	idx_ticket_visitor	Visitor_ID

Foreign Keys

Type	Name	On
	Ticket_ibfk_1 (Performance_ID) ref Performance (Performance_ID)	
	Ticket_ibfk_2 (Visitor_ID) ref Visitor (Visitor_ID)	

Triggers

	Name	Definition
	check_stage_capacity	

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Ticket FOR EACH ROW BEGIN
  DECLARE total_tickets INT;
  DECLARE stage_capacity INT;
  SELECT COUNT(*) INTO total_tickets
  FROM Ticket
  WHERE Performance_ID = NEW.Performance_ID;
  SELECT s.Max_Capacity INTO stage_capacity
  FROM Performance p
  JOIN Event e ON p.Event_ID = e.Event_ID
  JOIN Stage s ON e.Stage_ID = s.Stage_ID
  WHERE p.Performance_ID = NEW.Performance_ID;
  IF total_tickets >= stage_capacity THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Cannot sell ticket: stage capacity exceeded.';
  END IF;
END
```

	check_vip_limit	

Table Ticket

```
CREATE TRIGGER ${nameWithSchemaName} BEFORE INSERT ON Ticket FOR EACH ROW BEGIN
  DECLARE vip_tickets INT;
  DECLARE stage_capacity INT;
  DECLARE max_vip_tickets INT;
  IF NEW.Category = 'VIP' THEN
    SELECT COUNT(*) INTO vip_tickets
    FROM Ticket
    WHERE Performance_ID = NEW.Performance_ID
      AND Category = 'VIP';
    SELECT s.Max_Capacity INTO stage_capacity
    FROM Performance p
    JOIN Event e ON p.Event_ID = e.Event_ID
    JOIN Stage s ON e.Stage_ID = s.Stage_ID
    WHERE p.Performance_ID = NEW.Performance_ID;
    SET max_vip_tickets = FLOOR(stage_capacity * 0.10);
    IF vip_tickets >= max_vip_tickets THEN
      SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot sell VIP ticket: VIP limit exceeded.';
    END IF;
  END IF;
END
```

Options

```
ENGINE=InnoDB AUTO_INCREMENT=401 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table TicketResaleQueue

Idx	Name	Data Type
* Pk	Queue_ID	INT
Idx	Ticket_ID	INT
	Queue_Type	TEXT
	Queue_Timestamp	DATETIME

Indexes

Type	Name	On
Pk	pk_ticketresalequeue	Queue_ID
	Ticket_ID	Ticket_ID

Foreign Keys

Type	Name	On
	TicketResaleQueue_ibfk_1 (Ticket_ID) ref Ticket (Ticket_ID)	

Options

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Table Visitor

Idx	Name	Data Type
* Pk	Visitor_ID	INT AUTO_INCREMENT
*	First_Name	VARCHAR(100)
*	Last_Name	VARCHAR(100)
*	Contact_Info	VARCHAR(255)
*	Age	INT

Indexes

Type	Name	On
Pk	pk_visitor	Visitor_ID

Constraints

Name	Definition
cns_visitor_age	`Age` >= 0

Options

```
ENGINE=InnoDB AUTO_INCREMENT=401 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci
```

Schema kosni_db

Procedures

ProcessResaleQueue