

**TIS1101 Database Fundamentals / TDB2111 Database System**

**Group Project**

**Title: Concert Ticketing System**

**Prepared by:**

|  |  |  |
| --- | --- | --- |
| GOH KUN SHUN | 1151101980 | kunshun225@gmail.com |
| JOHN CHRISTIAN GONZALES ESCOBIA | 1132701350 | escobiajohn@gmail.com |
| CHRISTOPHER TOO WEI BIN | 1151101473 | christopher\_two@hotmail.com |
| NG JING KEONG | 1151100169 | james0523njk@gmail.com |

****

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Dictionary** | | | | | | | | |
| **TABLE NAME** | **ATTRIBUTE NAME** | **CONTENTS** | **TYPE** | **FORMAT** | **RANGE** | **REQUIRED** | **PK OR FK** | **FK REFERENCED TABLE** |
| THEATRE | Theatre\_ID | Theatre identification code | CHAR( 6 ) | Xx9999 | TT0000 - TT9999 | Y | PK |  |
|  | Theatre\_Name | Theatre name | VARCHAR( 20 ) | Xxxxxxxx |  | Y |  |  |
|  | Theatre\_Venue | Theatre venue | VARCHAR( 20 ) | Xxxxxxxx |  | Y |  |  |
|  | Seat\_Capacity | Number of seats | INTEGER | 9999 | 50 - 125000 | Y |  |  |
|  | VIP\_Count | Number of VIP seats. | INTEGER | 9999 | 0 - Seat\_Capacity | Y |  |  |
|  | Zone\_Count | Number of zones in theatre. | INTEGER | 99 | 1 - 5 | Y |  |  |
| CONCERT | Con\_ID | Concert identification code | CHAR( 6 ) | Xx9999 | CN0000 - CN9999 | Y | PK |  |
|  | Con\_Title | Concert title | VARCHAR( 20 ) | Xxxxxxxx |  | Y |  |  |
|  | Con\_Artist | Concert artist | VARCHAR( 20 ) | Xxxxxxxx |  |  |  |  |
|  | Con\_Date | Date of the concert | DATE | YYYY-MM-DD |  | Y |  |  |
|  | Con\_TimeStart | Concert starting time | TIME | HH-MM | 0000 - 2359 | Y |  |  |
|  | Con\_TimeEnd | Concert ending time | TIME | HH-MM | 0000 - 2359 | Y |  |  |
|  | VIP\_Price | VIP ticket price | DECIMAL( 6, 2 ) | 9999.99 | 0.00 - 9999.99 | Y |  |  |
|  | Normal\_Price | Normal ticket price | DECIMAL( 6, 2 ) | 9999.99 | 0.00 - 9999.99 | Y |  |  |
|  | Theatre\_ID | Theatre identification code | CHAR( 6 ) | T9999 | TT0000 - TT9999 | Y | FK | THEATRE |
| CUSTOMER | Cus\_ID | Customer identification code | CHAR( 6 ) | Xx9999 | CS0000 - CS9999 | Y | PK |  |
|  | Cus\_FName | Customer first name | VARCHAR( 15 ) | Xxxxxxxx |  | Y |  |  |
|  | Cus\_LName | Customer last name | VARCHAR( 15 ) | Xxxxxxxx |  | Y |  |  |
|  | Cus\_Age | Customer age | INTEGER | 99 |  |  |  |  |
|  | Cus\_Gender | Customer gender | CHAR( 1 ) | X |  | Y |  |  |
| AGENT | Agent\_ID | Agent identification code | CHAR( 6 ) | Xx9999 | AG0000 - AG9999 | Y | PK |  |
|  | Agent\_FName | Agent first name | VARCHAR( 15 ) | Xxxxxxxx |  | Y |  |  |
|  | Agent\_LName | Agent last name | VARCHAR( 15 ) | Xxxxxxxx |  | Y |  |  |
|  | Agent\_Age | Agent age | INTEGER | 99 |  |  |  |  |
|  | Agent\_Gender | Agent gender | CHAR( 1 ) | X |  | Y |  |  |
| BOOKING | Booking\_ID | Booking identification code | CHAR( 6 ) | Xx9999 | BK0000 - BK9999 | Y | PK |  |
|  | Cus\_ID | Customer identification code | CHAR( 6 ) | Xx9999 | CS0000 - CS9999 | Y | PK, FK1 | CUSTOMER |
|  | Agent\_ID | Agent identification code | CHAR( 6 ) | Xx9999 | AG0000 - AG9999 | Y | PK, FK2 | AGENT |
|  | Booking\_Time | Booking time | TIMESTAMP | YYYY-MM-DD HH24:MI:SS |  | Y |  |  |
|  | Total\_Price | Total price for the booking | decimal( 8, 2 ) | 99999999.99 | 0.00 - 999999.99 |  |  |  |
| TICKET | Ticket\_ID | Ticket identification code | CHAR( 6 ) | Xx9999 | TK0000 - TK9999 | Y | PK |  |
|  | Con\_ID | Concert identification code | CHAR( 6 ) | Xx9999 | CN0000 - CN9999 | Y | PK, FK1 | CONCERT |
|  | Ticket\_Price | Ticket price | DECIMAL( 6, 2 ) | 9999.99 | 0.00 - 9999.99 | Y |  |  |
|  | Ticket\_Type | Ticket type | VARCHAR( 6 ) | Xxxxxx |  | Y |  |  |
|  | Booking\_ID | Booking identification code | CHAR( 6 ) | Xx9999 | BK0000-BK9999 |  | FK2 | BOOKING |
| VIP\_TICKET | Ticket\_ID | Ticket identification code | CHAR( 6 ) | Xx9999 | TK0000 - TK9999 | Y | PK, FK1 | TICKET |
|  | Con\_ID | Concert identification code | CHAR( 6 ) | Xx9999 | CN0000 - CN9999 | Y | PK, FK2 | CONCERT |
|  | Seat\_No | Seat number | CHAR( 5 ) | X9999 | V0000 - V9999 | Y |  |  |
| NORMAL\_TICKET | Ticket\_ID | Ticket identification code | CHAR( 6 ) | Xx9999 | TK0000 - TK9999 | Y | PK, FK1 | TICKET |
|  | Con\_ID | Concert identification code | CHAR( 6 ) | Xx9999 | CN0000 - CN9999 | Y | PK, FK2 | CONCERT |
|  | Zone\_No | Zone number | CHAR( 1 ) | X | A - F | Y |  |  |

# Entities and Business Rules

## Entities

1. Agent
2. Booking
3. Theatre
4. Customer
5. Ticket
6. VIP Ticket
7. Normal Ticket
8. Concert

## Business Rules

1. One concert has one to many tickets, but one ticket must have only one concert.
2. One theatre can have many concerts, but one concert must only have one

## theatre.

## Every ticket must be either a normal ticket or a VIP ticket.

## Every ticket may have only one booking, but every booking may have many tickets.

## One booking must have only one agent and only one customer, but one agent and one customer can have many bookings.

## The number of tickets for each concert is depended on the number of seats in the theatre.

# Relationships

## One concert has one to many tickets, but one ticket must have only one concert.



## One theatre may have many concerts, but one concert must only have one

## theatre.



## Every ticket must be either a normal ticket or a VIP ticket.



## Every ticket may have only one booking, but every booking may have many tickets.



## One booking must have only one agent and only one customer, but one agent and one customer can have many bookings.

## 

# Data Definition Command (DDL)

\*We have initialized our IBM DB2 with ‘@’ as the termination character.

## Table Creation

*CREATE TABLE THEATRE*

*(*

*Theatre\_ID CHAR( 6 ) NOT NULL CHECK ( Theatre\_ID BETWEEN 'TT0000' AND*

*'TT9999' ),*

*Theatre\_Name VARCHAR( 20 ) NOT NULL,*

*Theatre\_Venue VARCHAR( 20 ) NOT NULL,*

*Seat\_Capacity INTEGER NOT NULL CHECK ( Seat\_Capacity BETWEEN 50*

*AND 125000 ),*

*VIP\_Count INTEGER NOT NULL,*

*Zone\_Count INTEGER NOT NULL CHECK ( Zone\_Count BETWEEN 1 AND*

*6 ),*

*PRIMARY KEY ( Theatre\_ID )*

*)@*

*CREATE TABLE CONCERT*

*(*

*Con\_ID CHAR( 6 ) NOT NULL CHECK ( Con\_ID BETWEEN 'CN0000'*

*AND 'CN9999' ),*

*Con\_Title VARCHAR( 20 ) NOT NULL,*

*Con\_Artist VARCHAR( 20 ),*

*Con\_Date DATE NOT NULL,*

*Con\_TimeStart TIME NOT NULL,*

*Con\_TimeEnd TIME NOT NULL,*

*VIP\_Price DECIMAL( 6, 2 ) NOT NULL,*

*Normal\_Price DECIMAL( 6, 2 ) NOT NULL,*

*Theatre\_ID CHAR( 6 ) NOT NULL,*

*PRIMARY KEY ( Con\_ID ),*

*FOREIGN KEY ( Theatre\_ID ) REFERENCES THEATRE( Theatre\_ID ) ON DELETE CASCADE*

*)@*

*CREATE TABLE CUSTOMER*

*(*

*Cus\_ID CHAR( 6 ) NOT NULL CHECK ( Cus\_ID BETWEEN 'CS0000' AND*

*'CS9999' ),*

*Cus\_FName VARCHAR( 15 ) NOT NULL,*

*Cus\_LName VARCHAR( 15 ) NOT NULL,*

*Cus\_Age INTEGER,*

*Cus\_Gender CHAR( 1 ) NOT NULL,*

*PRIMARY KEY ( Cus\_ID )*

*)@*

*CREATE TABLE AGENT*

*(*

*Agent\_ID CHAR( 6 ) NOT NULL CHECK ( Agent\_ID BETWEEN 'AG0000'*

*AND 'AG9999' ),*

*Agent\_FName VARCHAR( 15 ) NOT NULL,*

*Agent\_LName VARCHAR( 15 ) NOT NULL,*

*Agent\_Age INTEGER,*

*Agent\_Gender CHAR( 1 ) NOT NULL,*

*PRIMARY KEY ( Agent\_ID )*

*)@*

*CREATE TABLE BOOKING*

*(*

*Booking\_ID CHAR( 6 ) NOT NULL CHECK ( Booking\_ID BETWEEN 'BK0000'*

*AND 'BK9999' ),*

*Cus\_ID CHAR( 6 ) NOT NULL,*

*Agent\_ID CHAR( 6 ) NOT NULL,*

*Booking\_Time TIMESTAMP NOT NULL,*

*Total\_Price DECIMAL( 8, 2 ) NOT NULL DEFAULT 0.0,*

*PRIMARY KEY ( Booking\_ID ),*

*FOREIGN KEY ( Cus\_ID ) REFERENCES CUSTOMER( Cus\_ID ),*

*FOREIGN KEY ( Agent\_ID ) REFERENCES AGENT( Agent\_ID )*

*)@*

*CREATE TABLE TICKET*

*(*

*Ticket\_ID CHAR( 6 ) NOT NULL CHECK ( Ticket\_ID BETWEEN 'TK0000'*

*AND 'TK9999' ),*

*Con\_ID CHAR( 6 ) NOT NULL,*

*Ticket\_Price DECIMAL( 6, 2 ) NOT NULL CHECK ( Ticket\_Price BETWEEN 0 AND*

*9999.99 ),*

*Ticket\_Type VARCHAR( 6 ) NOT NULL,*

*Booking\_ID CHAR( 6 ),*

*PRIMARY KEY ( Ticket\_ID, Con\_ID ),*

*FOREIGN KEY ( Con\_ID ) REFERENCES CONCERT ( Con\_ID ) ON DELETE CASCADE,*

*FOREIGN KEY ( Booking\_ID ) REFERENCES BOOKING ( Booking\_ID ) ON DELETE CASCADE*

*)@*

*CREATE TABLE VIP\_TICKET*

*(*

*Ticket\_ID CHAR( 6 ) NOT NULL CHECK ( Ticket\_ID BETWEEN 'TK0000'*

*AND 'TK9999' ),*

*Con\_ID CHAR( 6 ) NOT NULL CHECK ( Con\_ID BETWEEN 'CN0000'*

*AND 'CN9999' ),*

*Seat\_No CHAR( 5 ) NOT NULL CHECK ( Seat\_No BETWEEN 'V0000'*

*AND 'V9999' ),*

*PRIMARY KEY ( Ticket\_ID, Con\_ID ),*

*FOREIGN KEY ( Ticket\_ID, Con\_ID ) REFERENCES TICKET ( Ticket\_ID, Con\_ID ) ON DELETE CASCADE*

*)@*

*CREATE TABLE NORMAL\_TICKET*

*(*

*Ticket\_ID CHAR( 6 ) NOT NULL CHECK ( Ticket\_ID BETWEEN 'TK0000'*

*AND 'TK9999' ),*

*Con\_ID CHAR( 6 ) NOT NULL CHECK ( Con\_ID BETWEEN 'CN0000'*

*AND 'CN9999' ),*

*Zone\_No CHAR( 1 ) NOT NULL CHECK ( Zone\_No BETWEEN 'A' AND*

*'F' ),*

*PRIMARY KEY ( Ticket\_ID, Con\_ID ),*

*FOREIGN KEY ( Ticket\_ID, Con\_ID ) REFERENCES TICKET ( Ticket\_ID, Con\_ID ) ON DELETE*

*CASCADE*

*)@*

# Data Definition Command (DDL)

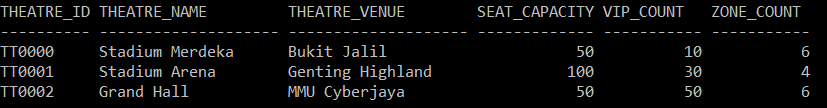
## Data Insertion

*INSERT INTO THEATRE VALUES ( 'TT0000','Stadium Merdeka', 'Bukit Jalil', 50, 10, 6 )@*

*INSERT INTO THEATRE VALUES ( 'TT0001','Stadium Arena', 'Genting Highland', 100, 30, 4 )@*

*INSERT INTO THEATRE VALUES ( 'TT0002','Grand Hall', 'MMU Cyberjaya', 50, 55, 6 )@*

*SELECT \* FROM THEATRE@*

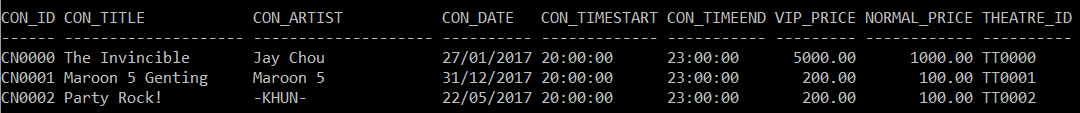


*INSERT INTO CONCERT VALUES ( 'CN0000', 'The Invincible', 'Jay Chou', '2017-01-27', '20:00:00', '23:00:00', 5000.00, 1000.0, 'TT0000' )@*

*INSERT INTO CONCERT VALUES ( 'CN0001', 'Maroon 5 Genting', 'Maroon 5', '2017-12-31', '20:00:00', '23:00:00', 200.00, 100.0, 'TT0001' )@*

*INSERT INTO CONCERT VALUES ( 'CN0002', 'Party Rock!', '-KHUN-', '2017-05-22', '20:00:00', '23:00:00', 200.00, 100.0, 'TT0002' )@*

*SELECT \* FROM CONCERT@*



*INSERT INTO CUSTOMER VALUES ( 'CS0000', 'GOH', 'KUN SHUN', 19, 'M' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0001', 'CHRISTOPHER', 'TOO WEI BIN', 19, 'M' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0002', 'JOHN', 'ESCOBIA', 19, 'M' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0003', 'NG', 'JING KEONG', 19, 'M' )@*

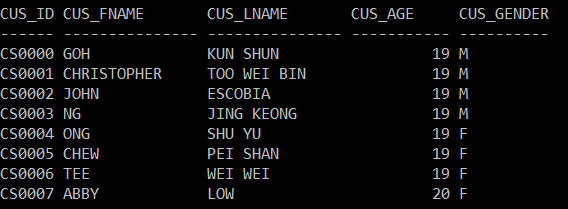
*INSERT INTO CUSTOMER VALUES ( 'CS0004', 'ONG', 'SHU YU', 19, 'F' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0005', 'CHEW', 'PEI SHAN', 19, 'F' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0006', 'TEE', 'WEI WEI', 19, 'F' )@*

*INSERT INTO CUSTOMER VALUES ( 'CS0007', 'ABBY', 'LOW', 20, 'F' )@*

*SELECT \* FROM CUSTOMER@*



*INSERT INTO AGENT VALUES ( 'AG0000', 'ELON', 'MUSK', 45, 'M' )@*

*INSERT INTO AGENT VALUES ( 'AG0001', 'BILL', 'GATES', 61, 'M' )@*

*INSERT INTO AGENT VALUES ( 'AG0002', 'NEIL', 'TYSON', 58, 'M' )@*

*INSERT INTO AGENT VALUES ( 'AG0003', 'STEPHEN', 'HAWKING', 75, 'M' )@*

*INSERT INTO AGENT VALUES ( 'AG0004', 'WARREN', 'BUFFET', 86, 'M' )@*

*INSERT INTO AGENT VALUES ( 'AG0005', 'OPRAH', 'WINFREY', 62, 'F' )@*

*INSERT INTO AGENT VALUES ( 'AG0006', 'HEDY', 'LAMARR', 85, 'F' )@*

*SELECT \* FROM AGENT@*



*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0000', 'CS0000', 'AG0000' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0001', 'CS0004', 'AG0006' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0002', 'CS0005', 'AG0006' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0003', 'CS0004', 'AG0003' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0004', 'CS0003', 'AG0005' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0005', 'CS0007', 'AG0002' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0006', 'CS0006', 'AG0005' )@*

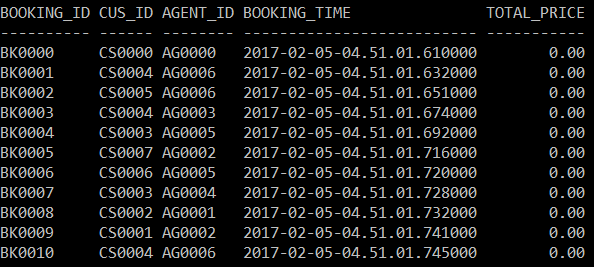
*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0007', 'CS0003', 'AG0004' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0008', 'CS0002', 'AG0001' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0009', 'CS0001', 'AG0002' )@*

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0010', 'CS0004', 'AG0006' )@*

*SELECT \* FROM BOOKING@*



## Data Update

### Trying to book one expired concert’s ticket, and one available ticket

*UPDATE TICKET*

*SET Booking\_ID = 'BK0000'*

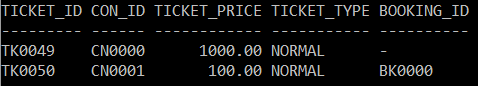
*WHERE Ticket\_ID = 'TK0049' AND Con\_ID = 'CN0000' OR -- Expired concert*

*Ticket\_ID = 'TK0050' AND Con\_ID = 'CN0001'@ -- Okay concert*

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0049' AND Con\_ID = 'CN0000' OR*

*Ticket\_ID = 'TK0050' AND Con\_ID = 'CN0001'@*



### Trying to book different ticket types

*UPDATE TICKET*

*SET Booking\_ID = 'BK0001'*

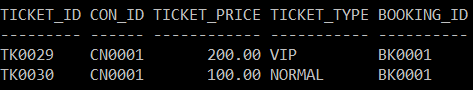
*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR -- VIP ticket*

*Ticket\_ID = 'TK0030' AND Con\_ID = 'CN0001'@ -- Normal ticket*

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR*

*Ticket\_ID = 'TK0030' AND Con\_ID = 'CN0001'@*



### Trying to book 2 invalid tickets

*UPDATE TICKET*

*SET Booking\_ID = 'BK0002'*

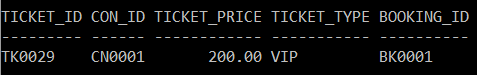
*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR -- Booked ticket*

*Ticket\_ID = 'TK9999' AND Con\_ID = 'CN0001'@ -- Invalid ticket ID*

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR*

*Ticket\_ID = 'TK9999' AND Con\_ID = 'CN0001'@*



### Booking multiple tickets with BETWEEN

*UPDATE TICKET*

*SET Booking\_ID = 'BK0003'*

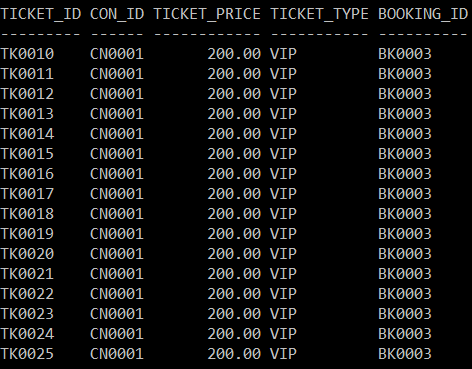
*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0010' AND 'TK0025'@*

*SELECT \* FROM TICKET*

*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0010' AND 'TK0025'@*



### Booking using subquery

*UPDATE TICKET*

*SET Booking\_ID = 'BK0004'*

*WHERE Ticket\_ID = 'TK0000' AND*

*Con\_ID IN ( SELECT Con\_ID*

*FROM CONCERT*

*WHERE Con\_Date > '2017-06-01' )@*

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0000' AND*

*Con\_ID IN ( SELECT Con\_ID*

*FROM CONCERT*

*WHERE Con\_Date > '2017-06-01' )@*



### Booking using multi-level subqueries and LIKE

*UPDATE TICKET*

*SET Booking\_ID = 'BK0005'*

*WHERE Ticket\_ID = 'TK0001' AND*

*Con\_ID IN ( SELECT Con\_ID*

*FROM CONCERT*

*WHERE Theatre\_ID IN ( SELECT Theatre\_ID*

*FROM THEATRE*

*WHERE Theatre\_Name LIKE*

*'Stadium%' ) )@*

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0001' AND*

*Con\_ID IN ( SELECT Con\_ID*

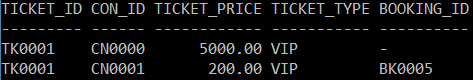
*FROM CONCERT*

*WHERE Theatre\_ID IN ( SELECT Theatre\_ID*

*FROM THEATRE*

*WHERE Theatre\_Name LIKE*

*'Stadium%' ) )@*



### Book all VIP within a range

*UPDATE TICKET*

*SET Booking\_ID = 'BK0006'*

*WHERE Con\_ID = 'CN0002' AND*

*Ticket\_ID BETWEEN 'TK0000' AND 'TK0005' AND*

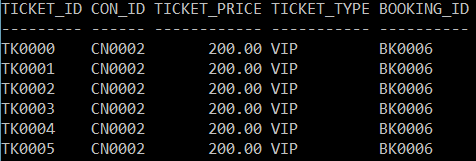
*Ticket\_Type = 'VIP'@*

*SELECT \* FROM TICKET*

*WHERE Con\_ID = 'CN0002' AND*

*( Ticket\_ID BETWEEN 'TK0000' AND 'TK0005' ) AND*

*Ticket\_Type = 'VIP'@*



### Book normal ticket only

*UPDATE TICKET*

*SET Booking\_ID = 'BK0007'*

*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0025' AND 'TK0035' AND*

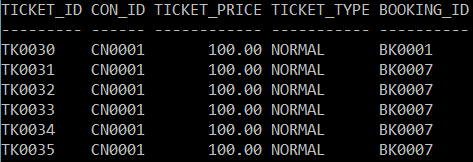
*TICKET\_TYPE = 'NORMAL'@*

*SELECT \* FROM TICKET*

*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0025' AND 'TK0035' AND*

*TICKET\_TYPE = 'NORMAL'@*



### Booking ticket based on ticket’s price

*UPDATE TICKET*

*SET Booking\_ID = 'BK0008'*

*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0025' AND 'TK0035' AND*

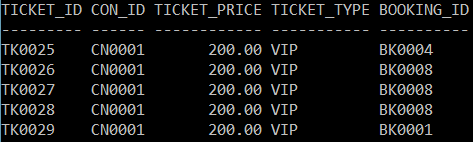
*Ticket\_Price > 100.0@*

*SELECT \* FROM TICKET*

*WHERE Con\_ID = 'CN0001' AND*

*Ticket\_ID BETWEEN 'TK0025' AND 'TK0035' AND*

*Ticket\_Price > 100.0@*



### Booking ticket based on ticket’s price with aggregate function

*UPDATE TICKET*

*SET Booking\_ID = 'BK0009'*

*WHERE ( Con\_ID = 'CN0001' OR Con\_ID = 'CN0002' ) AND*

*( Ticket\_ID BETWEEN 'TK0040' AND 'TK0044' ) AND*

*Ticket\_Price < ( SELECT AVG( Ticket\_Price )*

*FROM TICKET*

*GROUP BY Con\_ID*

*HAVING Con\_ID = 'CN0002' )@*

*SELECT \* FROM TICKET*

*WHERE ( Con\_ID = 'CN0001' OR Con\_ID = 'CN0002' ) AND*

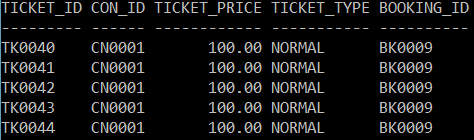
*( Ticket\_ID BETWEEN 'TK0040' AND 'TK0044' ) AND*

*Ticket\_Price < ( SELECT AVG( Ticket\_Price )*

*FROM TICKET*

*GROUP BY Con\_ID*

*HAVING Con\_ID = 'CN0002' )@*



# Data Manipulation Language (DML):

## Triggers

### TRG\_CHECK\_SEATCAP

*CREATE TRIGGER TRG\_CHECK\_SEATCAP*

*BEFORE INSERT ON THEATRE*

*REFERENCING NEW AS N*

*FOR EACH ROW MODE DB2SQL*

*BEGIN*

*IF N.VIP\_Count > N.Seat\_Capacity*

*THEN*

*SET N.VIP\_Count = N.Seat\_Capacity;*

*END IF;*

*END@*

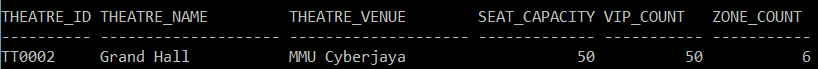
*The trigger above will check if user try to input VIP\_Count with a value greater than Seat\_Capacity of a THEATRE. If it is detected, the trigger will set the VIP\_Count to the Seat\_Capacity, which means the CONCERTs held on this THEATRE will not have NORMAL\_TICKETs.*

*Inserting this line of code:*

*INSERT INTO THEATRE VALUES ( 'TT0002', 'Grand Hall', 'MMU Cyberjaya', 50, 55, 6 )@*

*The output:*

*SELECT \* FROM THEATRE WHERE Theatre\_ID = 'TT0002'@*



Notice how the VIP\_Count is set to 50 instead of 55 because the trigger

detected the invalid input and automatically set it to Seat\_Capacity’s value.

### TRG\_GEN\_CONCERT\_TICKET

*CREATE TRIGGER TRG\_GEN\_CONCERT\_TICKET*

*AFTER INSERT ON CONCERT*

*REFERENCING NEW AS N*

*FOR EACH ROW MODE DB2SQL*

*BEGIN*

*-- Declare variables*

*DECLARE I INTEGER; -- Counter = 0*

*DECLARE ID CHAR( 6 ); -- Ticket ID*

*DECLARE VIP\_ID CHAR( 5 ); -- VIP ticket ID*

*DECLARE T\_TYPE CHAR( 6 ); -- Ticket type*

*DECLARE T\_ZONE CHAR( 1 ); -- Ticket zone*

*DECLARE NO\_SEATS INTEGER; -- Number of seats*

*DECLARE NO\_VIP INTEGER; -- Number of VIP\_TICKETS*

*DECLARE NO\_NORM INTEGER; -- Number of NORMAL\_TICKETS*

*DECLARE VIP\_P DECIMAL( 6, 2 ); -- VIP ticket price*

*DECLARE NORM\_P DECIMAL( 6, 2 ); -- Normal ticket price*

*DECLARE NO\_ZONE INTEGER; -- Number of zones*

*-- Initialize vairables*

*SET I = 0;*

*SET VIP\_P = ( SELECT VIP\_PRICE FROM CONCERT WHERE N.CON\_ID =*

*CONCERT.CON\_ID );*

*SET NORM\_P = ( SELECT NORMAL\_PRICE FROM CONCERT WHERE N.CON\_ID =*

*CONCERT.CON\_ID );*

*SET NO\_SEATS = ( SELECT SEAT\_CAPACITY FROM THEATRE WHERE N.THEATRE\_ID =*

*THEATRE.THEATRE\_ID );*

*SET NO\_VIP = ( SELECT VIP\_COUNT FROM THEATRE WHERE N.THEATRE\_ID =*

*THEATRE.THEATRE\_ID );*

*SET NO\_NORM = NO\_SEATS - NO\_VIP;*

*SET NO\_ZONE = ( SELECT ZONE\_COUNT FROM THEATRE WHERE N.THEATRE\_ID =*

*THEATRE.THEATRE\_ID );*

*WHILE I < NO\_SEATS*

*DO*

*-- Determine TICKET\_ID*

*SET ID = ( 'TK' || LPAD( I, 4, 0 ) );*

*SET VIP\_ID = ( 'V' || LPAD( I, 4, 0 ) );*

*-- Determine TICKET\_TYPE*

*IF I < NO\_VIP*

*THEN*

*SET T\_TYPE = 'VIP';*

*-- Insert the generated VIP\_TICKET into TICKET*

*INSERT INTO TICKET VALUES ( ID, N.CON\_ID, VIP\_P,*

*T\_TYPE, NULL );*

*-- Insert the generated VIP\_TICKET into VIP\_TICKET*

*INSERT INTO VIP\_TICKET VALUES ( ID, N.CON\_ID, VIP\_ID );*

*ELSE*

*SET T\_TYPE = 'NORMAL';*

*IF I <= ( NO\_VIP + ( NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'A';*

*ELSEIF I <= ( NO\_VIP + ( 2 \* NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'B';*

*ELSEIF I <= ( NO\_VIP + ( 3 \* NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'C';*

*ELSEIF I <= ( NO\_VIP + ( 4 \* NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'D';*

*ELSEIF I <= ( NO\_VIP + ( 5 \* NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'E';*

*ELSEIF I <= ( NO\_VIP + ( 6 \* NO\_NORM / NO\_ZONE ) )*

*THEN*

*SET T\_ZONE = 'F';*

*END IF;*

*-- Insert the generated NORMAL\_TICKET into TICKET*

*INSERT INTO TICKET VALUES ( ID, N.CON\_ID, NORM\_P, T\_TYPE,*

*NULL );*

*-- Insert the generated NORMAL\_TICKET into NORMAL\_TICKET*

*INSERT INTO NORMAL\_TICKET VALUES ( ID, N.CON\_ID, T\_ZONE );*

*END IF;*

*-- Increase counter by 1*

*SET I = I + 1;*

*END WHILE;*

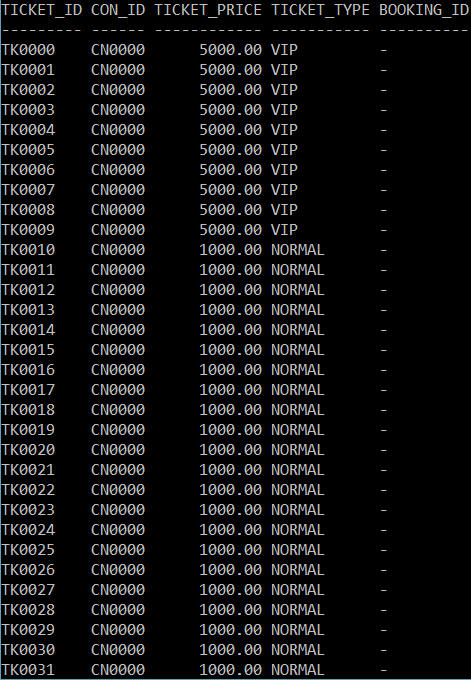
*END@*

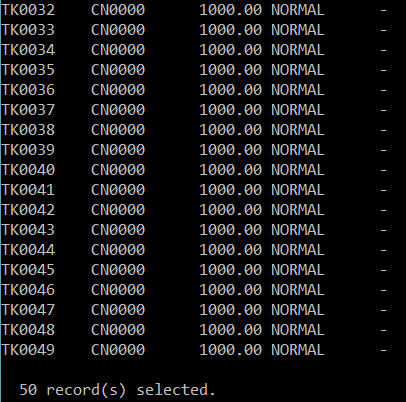
The above trigger will be triggered when user insert a CONCERT. The trigger will automatically generate the TICKETs based on the THEATRE the CONCERT will be held on. For example, this trigger will generate VIP\_TICKETs based on the VIP\_Count. The rest of the non-VIP\_TCIKETs will be generated as NORMAL\_TICKETs. For the NORMAL\_TICKETs, the Zone\_No is also automatically generated based on Zone\_Count of the THEATRE, and they will be divided equally.

Inserting this line of code:

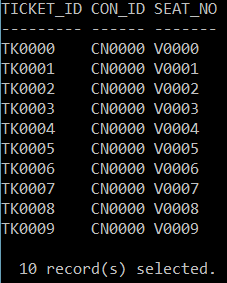
*INSERT INTO CONCERT VALUES ( 'CN0000', 'The Invincible', 'Jay Chou', '2017-01-27', '20:00:00', '23:00:00', 5000.00, 1000.0, 'TT0000' )@*

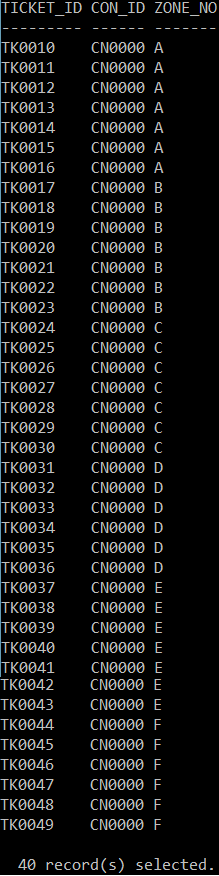
The output:

*SELECT \* FROM TICKET WHERE Con\_ID = ‘CN0000’@*

**

*SELECT \* FROM VIP\_TICKET WHERE Con\_ID = ‘CN0000’@*

**

*SELECT \* FROM NORMAL\_TICKET WHERE Con\_ID = ‘CN0000’@*

### TRG\_BOOKING

*CREATE TRIGGER TRG\_BOOKING*

*BEFORE INSERT ON BOOKING*

*REFERENCING NEW AS N*

*FOR EACH ROW MODE DB2SQL*

*BEGIN*

*SET N.BOOKING\_TIME = CURRENT TIMESTAMP;*

*END@*

The above trigger will be triggered when user create a BOOKING. It will automatically set the Booking\_Time to the current time using the CURRENT TIMESTAMP keyword in IBM DB2.

Inserting this line of code:

*INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0000', 'CS0000', 'AG0000' )@*

The output:

*SELECT \* FROM BOOKING WHERE Booking\_ID = ‘BK0000’@*



### TRG\_BOOK\_TICKET

*CREATE TRIGGER TRG\_BOOK\_TICKET*

*BEFORE UPDATE OF BOOKING\_ID ON TICKET*

*REFERENCING OLD AS O NEW AS N*

*FOR EACH ROW MODE DB2SQL*

*BEGIN*

*DECLARE CONCERT\_BEGIN TIMESTAMP; -- Concert begin time*

*DECLARE BOOKTIME TIMESTAMP; -- Booking time*

*-- Convert date to timestamp*

*SET CONCERT\_BEGIN = ( SELECT TIMESTAMP\_ISO( DATE( Con\_Date ) )*

*FROM CONCERT C*

*WHERE N.CON\_ID = C.CON\_ID );*

*SET BOOKTIME = ( SELECT Booking\_Time*

*FROM BOOKING B*

*WHERE N.BOOKING\_ID = B.BOOKING\_ID );*

*IF ( O.BOOKING\_ID IS NOT NULL ) OR ( BOOKTIME > CONCERT\_BEGIN )*

*THEN*

*SET N.BOOKING\_ID = O.BOOKING\_ID;*

*END IF;*

*END@*

The above trigger will be triggered when user create a BOOKING. It will automatically set the Booking\_Time to the current time using the CURRENT TIMESTAMP keyword in IBM DB2.

Inserting this line of code:

*UPDATE TICKET*

*SET Booking\_ID = 'BK0002'*

*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR -- Booked ticket*

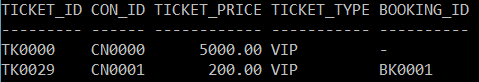
*Ticket\_ID = 'TK0000' AND Con\_ID = 'CN0000'@ -- Invalid ticket ID*

The output:

*SELECT \* FROM TICKET*

*WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR*

*Ticket\_ID = 'TK0000' AND Con\_ID = 'CN0000'@*



Notice how the and Booking\_ID of TK0000 is still null because the CONCERT it is associated is already expired ( refer to the Data Insertion section ) and Booking\_ID of TK0029 is not updated because it is previously booked by BK0001.

### TRG\_CALC\_BOOKING\_PRICE

*CREATE TRIGGER TRG\_CALC\_BOOKING\_PRICE*

*AFTER UPDATE OF BOOKING\_ID ON TICKET*

*REFERENCING NEW AS N*

*FOR EACH ROW MODE DB2SQL*

*UPDATE BOOKING B*

*SET TOTAL\_PRICE = ( SELECT SUM( TICKET\_PRICE )*

*FROM TICKET*

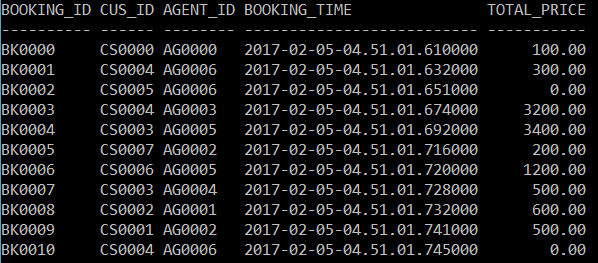
*WHERE TICKET.BOOKING\_ID = N.BOOKING\_ID )*

*WHERE N.BOOKING\_ID = B.BOOKING\_ID@*

The above trigger will be triggered when user update the Booking\_ID of a TICKET. It will automatically calculate the Total\_Price of a booking by getting the sum of all the Ticket\_Price that have the same Booking\_ID .

The output:

*SELECT \* FROM BOOKING@*



## Views

### AGENT\_SOLD

*CREATE VIEW AGENT\_SOLD AS*

*SELECT AGENT\_FNAME,*

*AGENT\_LNAME,*

*COUNT(NT.TICKET\_ID) AS NORMAL\_SOLD,*

*COUNT(VT.TICKET\_ID) AS VIP\_SOLD,*

*COUNT(NT.TICKET\_ID) + COUNT(VT.TICKET\_ID) AS TOTAL\_SOLD*

*FROM TICKET TK*

*LEFT OUTER JOIN NORMAL\_TICKET NT*

*ON TK.TICKET\_ID = NT.TICKET\_ID*

*AND TK.CON\_ID = NT.CON\_ID*

*LEFT OUTER JOIN VIP\_TICKET VT*

*ON TK.TICKET\_ID = VT.TICKET\_ID*

*AND TK.CON\_ID = VT.CON\_ID*

*LEFT OUTER JOIN BOOKING BK*

*ON BK.BOOKING\_ID = TK.BOOKING\_ID*

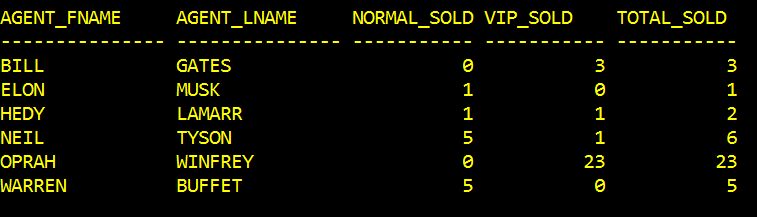
*LEFT OUTER JOIN AGENT AG*

*ON BK.AGENT\_ID = AG.AGENT\_ID*

*WHERE TK.BOOKING\_ID IS NOT NULL*

*GROUP BY ( AGENT\_FNAME, AGENT\_LNAME )@*

AGENT\_SOLD view is to show the number VIP ticket, normal ticket and total amount of ticket sold by an agent.



### CUSTOMER\_PURCHASE

*CREATE VIEW CUSTOMER\_PURCHASE AS*

*SELECT CUS\_FNAME,*

*CUS\_LNAME,*

*COUNT(NT.TICKET\_ID) AS NORMAL\_BOUGHT,*

*COUNT(VT.TICKET\_ID) AS VIP\_BOUGHT,*

*COUNT(NT.TICKET\_ID) + COUNT(VT.TICKET\_ID) AS TOTAL\_TICKET\_BOUGHT*

*FROM TICKET TK*

*LEFT OUTER JOIN NORMAL\_TICKET NT*

*ON TK.TICKET\_ID = NT.TICKET\_ID*

*AND TK.CON\_ID = NT.CON\_ID*

*LEFT OUTER JOIN VIP\_TICKET VT*

*ON TK.TICKET\_ID = VT.TICKET\_ID*

*AND TK.CON\_ID = VT.CON\_ID*

*LEFT OUTER JOIN BOOKING BK*

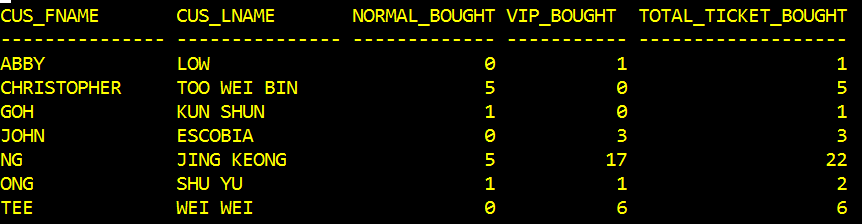
*ON BK.BOOKING\_ID = TK.BOOKING\_ID*

*LEFT OUTER JOIN CUSTOMER CS*

*ON BK.CUS\_ID = CS.CUS\_ID*

*WHERE TK.BOOKING\_ID IS NOT NULL*

*GROUP BY ( CUS\_FNAME, CUS\_LNAME )@*



CUSTOMER\_PURCHASE view is to show the amount of VIP ticket, normal ticket and total amount of ticket purchase by a customer.

### SEAT\_AVAILABLE

*CREATE VIEW SEAT\_AVAILABLE AS*

*SELECT CON\_TITLE,*

*COUNT(NT.TICKET\_ID) AS NORMAL\_AVAILABLE,*

*COUNT(VT.TICKET\_ID) AS VIP\_AVAILABLE*

*FROM TICKET T*

*LEFT OUTER JOIN CONCERT C*

*ON C.CON\_ID = T.CON\_ID*

*LEFT OUTER JOIN NORMAL\_TICKET NT*

*ON T.TICKET\_ID = NT.TICKET\_ID*

*AND T.CON\_ID = NT.CON\_ID*

*LEFT OUTER JOIN VIP\_TICKET VT*

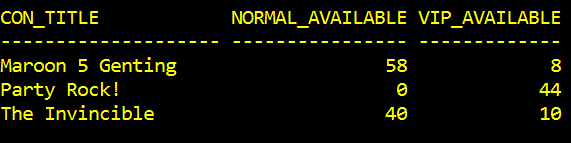
*ON T.TICKET\_ID = VT.TICKET\_ID*

*AND T.CON\_ID = VT.CON\_ID*

*WHERE T.BOOKING\_ID IS NULL*

*GROUP BY ( CON\_TITLE )@*

SEAT\_AVAILABLE view is to show the number of VIP ticket and normal ticket available for a concert.



### N0\_TICKET\_SOLD

*CREATE VIEW N0\_TICKET\_SOLD AS*

*SELECT CON\_TITLE,*

*COUNT( TICKET\_ID ) AS TICKETS\_SOLD,*

*SEAT\_CAPACITY,*

*( CAST( ( ( ( CAST( COUNT( TICKET\_ID ) AS DECIMAL ( 7, 3 ) ) ) / SEAT\_CAPACITY ) \* 100) AS DECIMAL ( 5, 2 ) ) ) AS SOLD\_PERCENT*

*FROM TICKET T*

*LEFT OUTER JOIN CONCERT C*

*ON C.CON\_ID = T.CON\_ID*

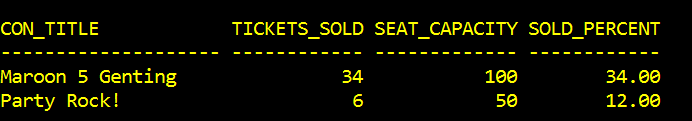
*LEFT OUTER JOIN THEATRE TT*

*ON C.THEATRE\_ID = TT.THEATRE\_ID*

*WHERE T.BOOKING\_ID IS NOT NULL*

*GROUP BY ( CON\_TITLE, SEAT\_CAPACITY )@*

N0\_TICKET\_SOLD view is to show the amount of ticket sold, seat capacity and the percentage of ticket sold for a concert, this view only shows ticket that are sold, therefore “The Invincible” won’t be listed because the concert did not sell any ticket.



### NORM\_TICKET\_SOLD

*CREATE VIEW NORM\_TICKET\_SOLD AS*

*SELECT CON\_TITLE,*

*COUNT( NT.TICKET\_ID ) AS NORMAL\_SOLD,*

*SEAT\_CAPACITY - VIP\_COUNT AS NORMAL\_CAPACITY,*

*( CAST( ( ( ( CAST( COUNT( NT.TICKET\_ID ) AS DECIMAL ( 7, 3 ) ) ) / (SEAT\_CAPACITY - VIP\_COUNT) ) \* 100) AS DECIMAL ( 5, 2 ) ) ) AS SOLD\_PERCENT*

*FROM NORMAL\_TICKET NT*

*LEFT OUTER JOIN CONCERT C*

*ON C.CON\_ID = NT.CON\_ID*

*LEFT OUTER JOIN THEATRE TT*

*ON C.THEATRE\_ID = TT.THEATRE\_ID*

*LEFT OUTER JOIN TICKET T*

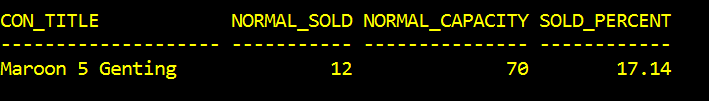
*ON NT.TICKET\_ID = T.TICKET\_ID*

*AND NT.CON\_ID = T.CON\_ID*

*WHERE T.BOOKING\_ID IS NOT NULL*

*GROUP BY ( CON\_TITLE, SEAT\_CAPACITY, VIP\_COUNT )@*

NORM\_TICKET\_SOLD view show the normal ticket sold by a concert, with their percentage and capacity for a concert. “Party Rock!” is not listed here because it does not have normal ticket allocated, while “The Invincible” is not listed because it did not sell any ticket.



### VIP\_TICKET\_SOLD

*CREATE VIEW VIP\_TICKET\_SOLD AS*

*SELECT CON\_TITLE,*

*COUNT( VT.TICKET\_ID ) AS VIP\_SOLD,*

*VIP\_COUNT AS VIP\_CAPACITY,*

*( CAST( ( ( ( CAST( COUNT( VT.TICKET\_ID ) AS DECIMAL ( 7, 3 ) ) ) / (VIP\_COUNT) ) \* 100) AS DECIMAL ( 5, 2 ) ) ) AS SOLD\_PERCENT*

*FROM VIP\_TICKET VT*

*LEFT OUTER JOIN CONCERT C*

*ON C.CON\_ID = VT.CON\_ID*

*LEFT OUTER JOIN THEATRE TT*

*ON C.THEATRE\_ID = TT.THEATRE\_ID*

*LEFT OUTER JOIN TICKET T*

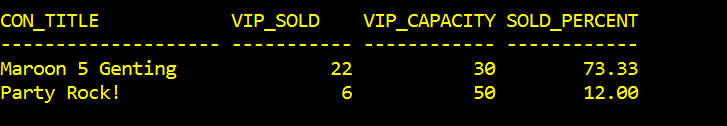
*ON VT.TICKET\_ID = T.TICKET\_ID*

*AND VT.CON\_ID = T.CON\_ID*

*WHERE T.BOOKING\_ID IS NOT NULL*

*GROUP BY ( CON\_TITLE, VIP\_COUNT )@*

VIP\_TICKET\_SOLD view show the VIP ticket sold by a concert, with their percentage and capacity for a concert. “The Invincible” is not listed because it did not sell any ticket.



## Stored Procedures

### spTop5Customers

Displays five customers with the most purchased tickets within two concert dates – parameters.

Creation:

*CREATE PROCEDURE spTop5Customers( IN startDate DATE, endDate DATE )*

*BEGIN*

*DECLARE c cursor with return for*

*SELECT CUSTOMER.CUS\_FNAME, CUSTOMER.CUS\_LNAME,*

*COUNT(TICKET.TICKET\_ID) AS TICKETS\_BOUGHT*

*FROM CUSTOMER, TICKET, BOOKING, CONCERT*

*WHERE CONCERT.CON\_DATE BETWEEN startDate AND endDate*

*AND BOOKING.BOOKING\_ID = TICKET.BOOKING\_ID*

*AND CUSTOMER.CUS\_ID = BOOKING.CUS\_ID*

*AND CONCERT.CON\_ID = TICKET.CON\_ID*

*GROUP BY CUSTOMER.CUS\_FNAME, CUSTOMER.CUS\_LNAME*

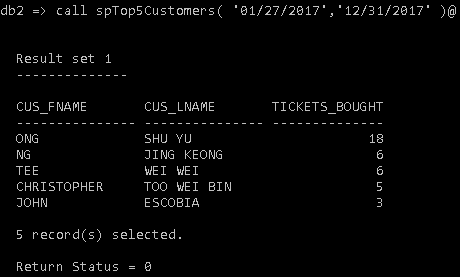
*ORDER BY COUNT(TICKET.TICKET\_ID) DESC*

*LIMIT 5;*

*OPEN c;*

*END@*

Call and output:



### spTop5Agents

Displays five agents who sold most tickets within two concert dates – parameters.

Creation:

*CREATE PROCEDURE spTop5Agents( IN startDate DATE, endDate DATE )*

*BEGIN*

*DECLARE c cursor with return for*

*SELECT AGENT.AGENT\_FNAME, AGENT.AGENT\_LNAME,*

*COUNT(TICKET.TICKET\_ID) AS TICKETS\_SOLD*

*FROM AGENT, TICKET, BOOKING, CONCERT*

*WHERE CONCERT.CON\_DATE BETWEEN startDate AND endDate*

*AND BOOKING.BOOKING\_ID = TICKET.BOOKING\_ID*

*AND AGENT.AGENT\_ID = BOOKING.AGENT\_ID*

*AND CONCERT.CON\_ID = TICKET.CON\_ID*

*GROUP BY AGENT.AGENT\_FNAME, AGENT.AGENT\_LNAME*

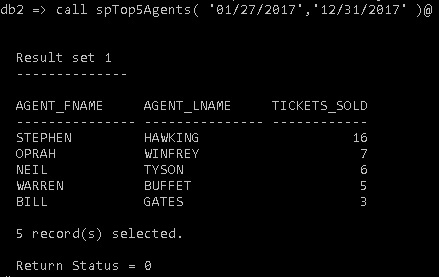
*ORDER BY COUNT(TICKET.TICKET\_ID) DESC*

*LIMIT 5;*

*OPEN c;*

*END@*

Call and output:



### spTop5Concerts

Displays five concerts with the most tickets sold within two concert dates – parameters.

Creation:

*CREATE PROCEDURE spTop2Concerts( IN startDate DATE, endDate DATE )*

*BEGIN*

*DECLARE c cursor with return for*

*SELECT CONCERT.CON\_TITLE, COUNT(TICKET.TICKET\_ID) AS*

*TICKETS\_SOLD*

*FROM CONCERT, TICKET, BOOKING*

*WHERE CONCERT.CON\_DATE BETWEEN startDate AND endDate*

*AND BOOKING.BOOKING\_ID = TICKET.BOOKING\_ID*

*AND CONCERT.CON\_ID = TICKET.CON\_ID*

*GROUP BY CONCERT.CON\_TITLE*

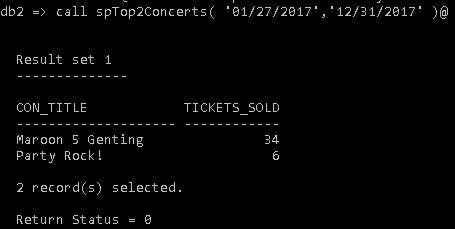
*ORDER BY COUNT(TICKET.TICKET\_ID) DESC*

*LIMIT 2;*

*OPEN c;*

*END@*

Call and output:



### spTheatreSchedule

Displays the concert(s) detail(s) – title, time start and time end for a specific theatre and date.

Creation:

*CREATE PROCEDURE spTheatreSchedule( IN theatreID CHAR( 6 ),*

*concertDate DATE )*

*BEGIN*

*DECLARE c cursor with return for*

*SELECT DISTINCT CONCERT.CON\_TITLE, CONCERT.CON\_TIMESTART,*

*CONCERT.CON\_TIMEEND*

*FROM CONCERT, THEATRE*

*WHERE theatreID = THEATRE.THEATRE\_ID*

*AND theatreID = CONCERT.THEATRE\_ID*

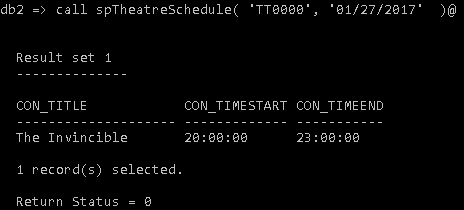
*AND concertDate = CONCERT.CON\_DATE*

*ORDER BY CONCERT.CON\_TIMESTART;*

*OPEN c;*

*END@*

Call and output:



### spAvailableTickets

Displays the tickets available for a certain concert.

Creation:

*CREATE PROCEDURE spAvailableTickets( IN concertID CHAR( 6 ) )*

*BEGIN*

*DECLARE c cursor with return for*

*SELECT DISTINCT TICKET.TICKET\_ID AS AVAILABLE\_TICKETS,*

*TICKET.TICKET\_TYPE, TICKET.TICKET\_PRICE*

*FROM TICKET, BOOKING, CONCERT*

*WHERE TICKET.BOOKING\_ID IS NULL*

*AND concertID = CONCERT.CON\_ID*

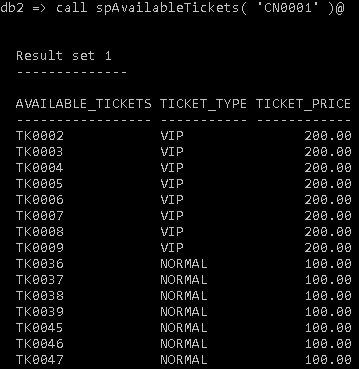
*AND concertID = TICKET.CON\_ID*

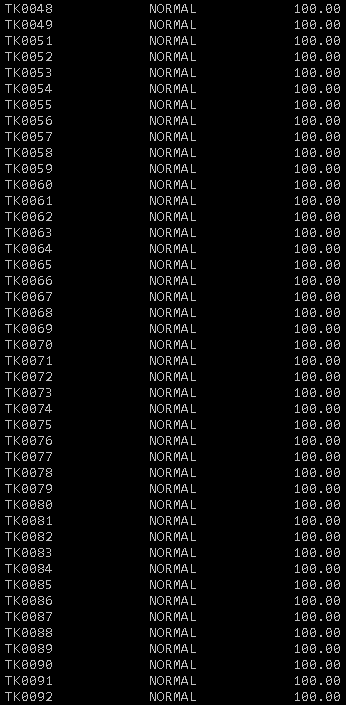
*ORDER BY TICKET.TICKET\_ID;*

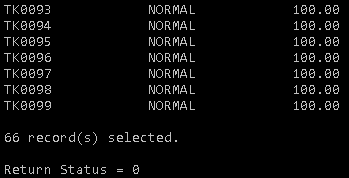
*OPEN c;*

*END@*

Call and output:







“66 record(s)” indicates there are 66 tickets available for concert CN0001.