

**TIS1101 Database Fundamentals / TDB2111 Database System**

**Group Project**

**Title: Concert Ticketing System**

**(Concert Ticketing System)**

**Members:**

|  |  |  |
| --- | --- | --- |
| GOH KUN SHUN | 1151101980 | kunshun225@gmail.com |
| JOHN CHRISTIAN | 1132701350 |  |
| CHRISTOPHER TOO WEI BIN | 1151101473 |  |
| NG JING KEONG | 1151100169 |  |

# 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.

## 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 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)

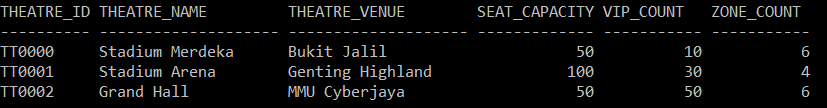
## Data Insertion

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@

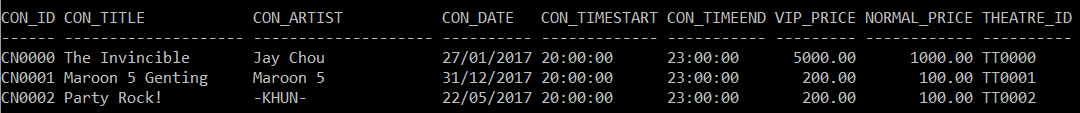


INSERT INTO CONCERT VALUES ( 'CN0000', 'The Invincible', 'Jay Chou', '2017-01-27', '20:00:00', '23:00:00', 5000.00, 1000.0, 'TT0000' )@

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' )@

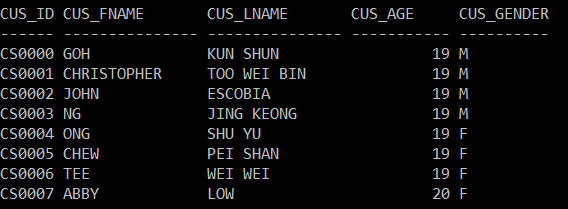
INSERT INTO CUSTOMER VALUES ( 'CS0004', 'ONG', 'SHU YU', 19, 'F' )@

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' )@

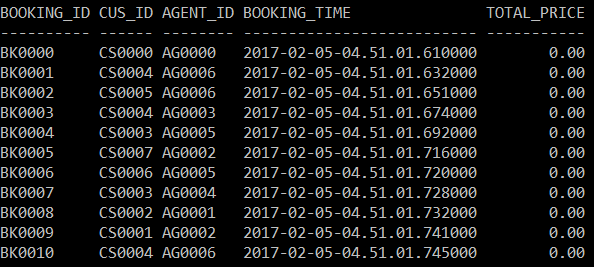
INSERT INTO BOOKING ( Booking\_ID, Cus\_ID, Agent\_ID ) VALUES ( 'BK0007', 'CS0003', 'AG0004' )@

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'

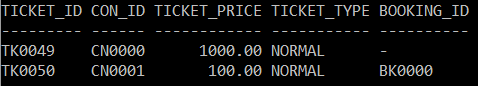
WHERE Ticket\_ID = 'TK0049' AND Con\_ID = 'CN0000' OR -- Expired concert

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'

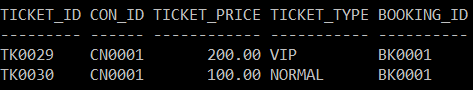
WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR -- VIP ticket

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'

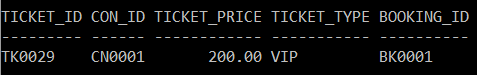
WHERE Ticket\_ID = 'TK0029' AND Con\_ID = 'CN0001' OR -- Booked ticket

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'

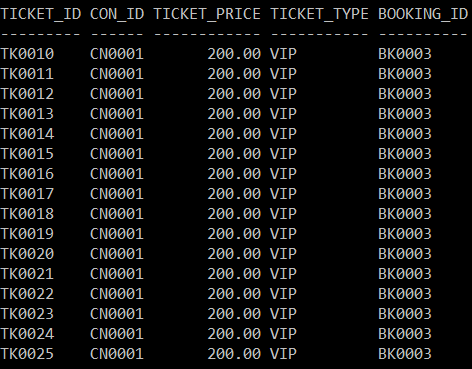
WHERE Con\_ID = 'CN0001' AND

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

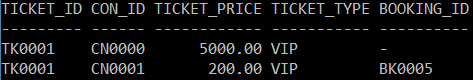
FROM CONCERT

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

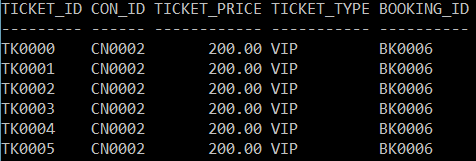
Ticket\_Type = 'VIP'@

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

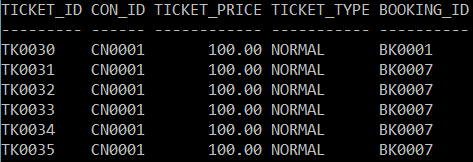
TICKET\_TYPE = 'NORMAL'@

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

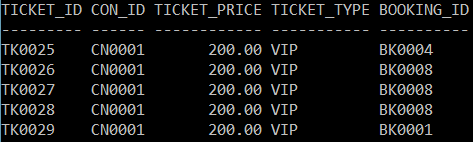
Ticket\_Price > 100.0@

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

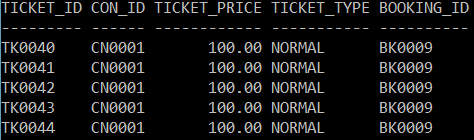
( Ticket\_ID BETWEEN 'TK0040' AND 'TK0044' ) AND

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@

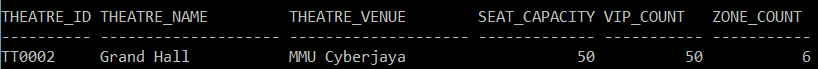
The trigger above will check if user try to input VIP\_Count with a value greater than Seat\_Capacity of a THEATRE. If it is detected, the trigger will set the VIP\_Count to the Seat\_Capacity, which means the CONCERTs held on this THEATRE will not have NORMAL\_TICKETs.

Inserting this line of code:

INSERT INTO THEATRE VALUES ( 'TT0002', 'Grand Hall', 'MMU Cyberjaya', 50, 55,

6 )@

The output:



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

### 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

## 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

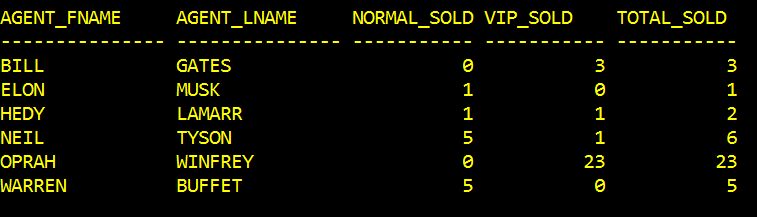
LEFT OUTER JOIN AGENT AG

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

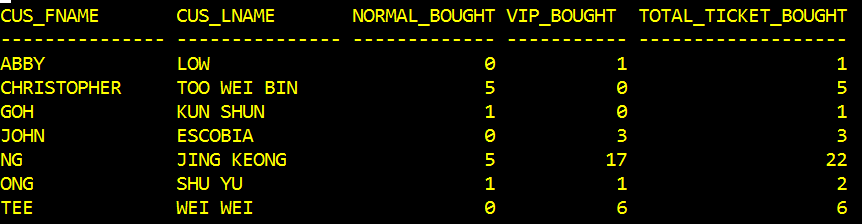
ON BK.BOOKING\_ID = TK.BOOKING\_ID

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

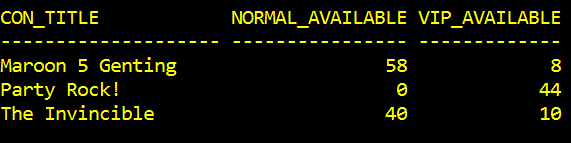
ON T.TICKET\_ID = VT.TICKET\_ID

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

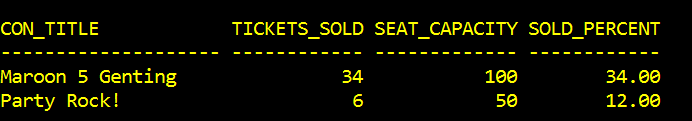
LEFT OUTER JOIN THEATRE TT

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” wont be listed because the concert did not sold 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

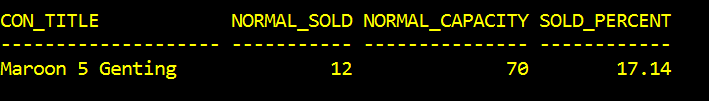
ON NT.TICKET\_ID = T.TICKET\_ID

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 sold 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 sold any ticket.

