TIS1101 Database Fundamentals / TDB2111 Database System

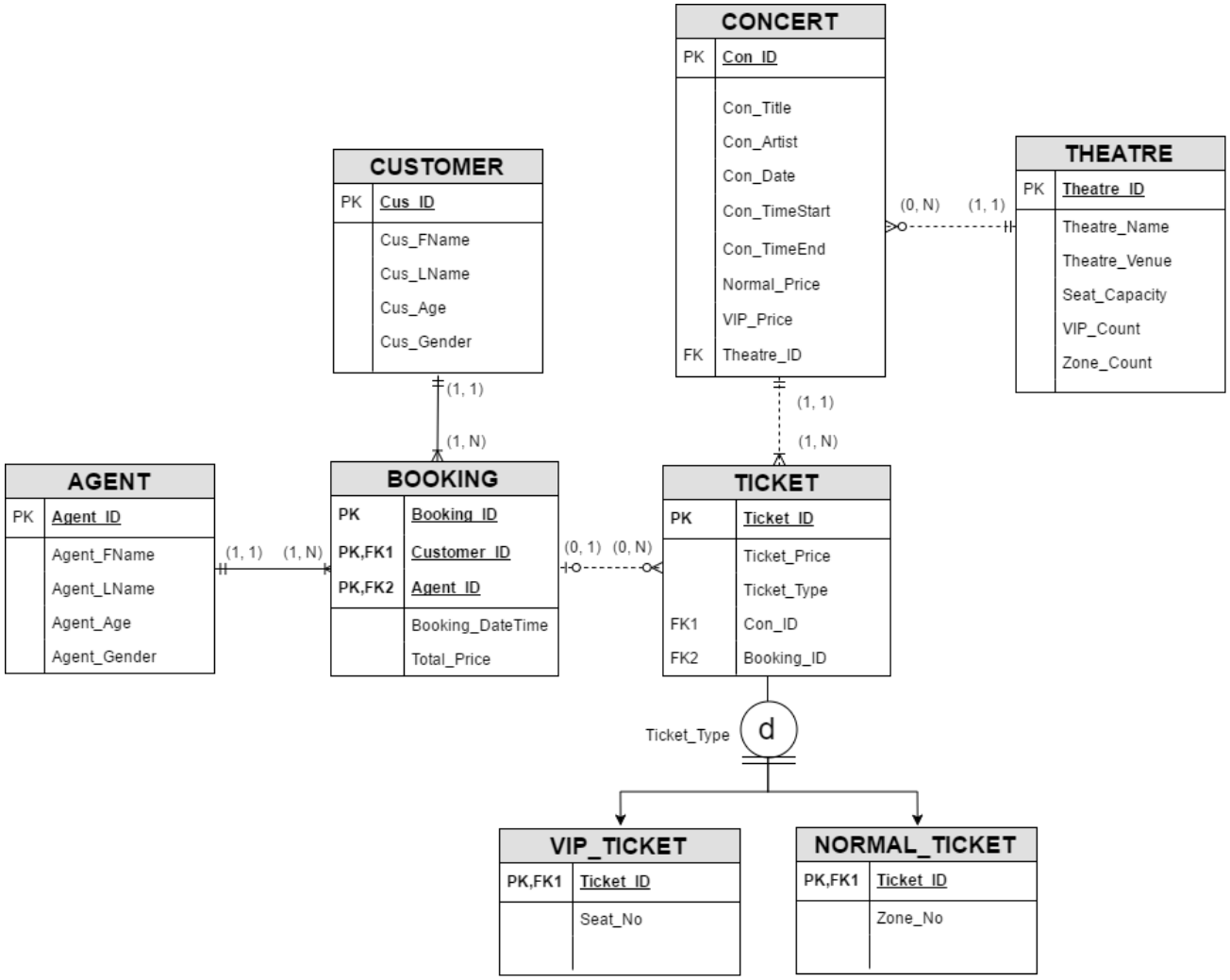
Group Project

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

TIS1101 Database Fundamentals



ER-Diagram

TIS1101 Database Fundamentals

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		

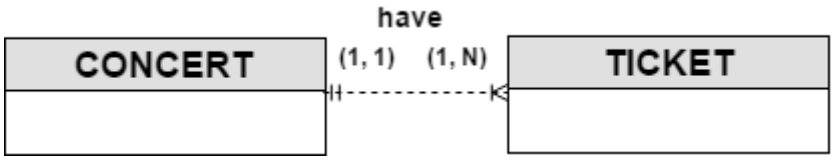
Entities and Business Rules

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

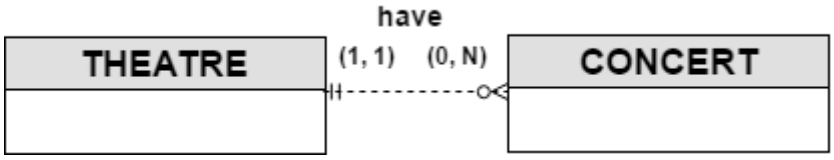
2. Business Rules
    1. One concert has one to many tickets, but one ticket must have only one concert.
    2. One theatre can have many concerts, but one concert must only have one theatre.
    3. Every ticket must be either a normal ticket or a VIP ticket.
    4. Every ticket may have only one booking, but every booking may have many tickets.
    5. One booking must have only one agent and only one customer, but one agent and one customer can have many bookings.
    6. The number of tickets for each concert is depended on the number of seats in the theatre.

Relationships

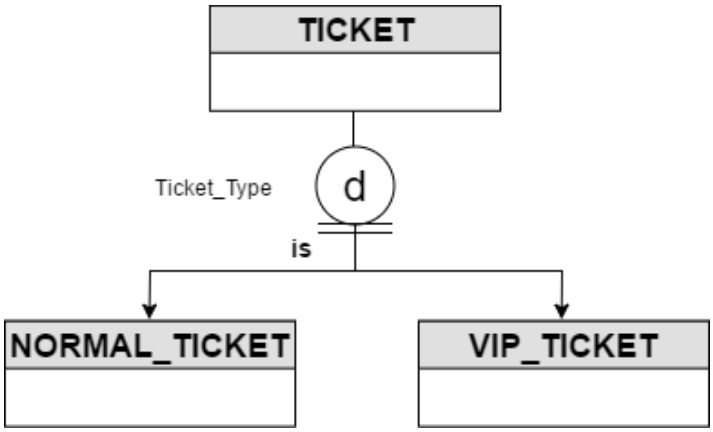
1. One concert has one to many tickets, but one ticket must have only one concert.



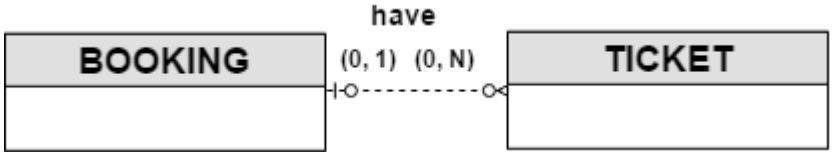
2. One theatre may have many concerts, but one concert must only have one theatre.



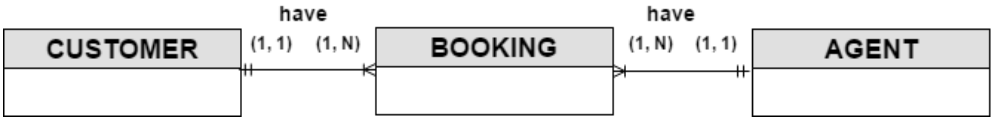
3. Every ticket must be either a normal ticket or a VIP ticket.



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



5. 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)

1. 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@*

THEATRE_ID	THEATRE_NAME	THEATRE_VENUE	SEAT_CAPACITY	VIP_COUNT	ZONE_COUNT
TT0000	Stadium Merdeka	Bukit Jalil	50	10	6
TT0001	Stadium Arena	Genting Highland	100	30	4
TT0002	Grand Hall	MMU Cyberjaya	50	50	6

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@*

CON_ID	CON_TITLE	CON_ARTIST	CON_DATE	CON_TIMESTART	CON_TIMEEND	VIP_PRICE	NORMAL_PRICE	THEATRE_ID
CN0000	The Invincible	Jay Chou	27/01/2017	20:00:00	23:00:00	5000.00	1000.00	TT0000
CN0001	Maroon 5 Genting	Maroon 5	31/12/2017	20:00:00	23:00:00	200.00	100.00	TT0001
CN0002	Party Rock!	-KHUN-	22/05/2017	20:00:00	23:00:00	200.00	100.00	TT0002

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@*

CUS_ID	CUS_FNAME	CUS_LNAME	CUS_AGE	CUS_GENDER
CS0000	GOH	KUN SHUN	19	M
CS0001	CHRISTOPHER	TOO WEI BIN	19	M
CS0002	JOHN	ESCOBIA	19	M
CS0003	NG	JING KEONG	19	M
CS0004	ONG	SHU YU	19	F
CS0005	CHEW	PEI SHAN	19	F
CS0006	TEE	WEI WEI	19	F
CS0007	ABBY	LOW	20	F

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@*

AGENT_ID	AGENT_FNAME	AGENT_LNAME	AGENT_AGE	AGENT_GENDER
AG0000	ELON	MUSK	45	M
AG0001	BILL	GATES	61	M
AG0002	NEIL	TYSON	58	M
AG0003	STEPHEN	HAWKING	75	M
AG0004	WARREN	BUFFET	86	M
AG0005	OPRAH	WINFREY	62	F
AG0006	HEDY	LAMARR	85	F

```
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@
```

BOOKING_ID	CUS_ID	AGENT_ID	BOOKING_TIME	TOTAL_PRICE
BK0000	CS0000	AG0000	2017-02-05-04.51.01.610000	0.00
BK0001	CS0004	AG0006	2017-02-05-04.51.01.632000	0.00
BK0002	CS0005	AG0006	2017-02-05-04.51.01.651000	0.00
BK0003	CS0004	AG0003	2017-02-05-04.51.01.674000	0.00
BK0004	CS0003	AG0005	2017-02-05-04.51.01.692000	0.00
BK0005	CS0007	AG0002	2017-02-05-04.51.01.716000	0.00
BK0006	CS0006	AG0005	2017-02-05-04.51.01.720000	0.00
BK0007	CS0003	AG0004	2017-02-05-04.51.01.728000	0.00
BK0008	CS0002	AG0001	2017-02-05-04.51.01.732000	0.00
BK0009	CS0001	AG0002	2017-02-05-04.51.01.741000	0.00
BK0010	CS0004	AG0006	2017-02-05-04.51.01.745000	0.00

2. Data Update

i. 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'@
```

```
TICKET_ID CON_ID TICKET_PRICE TICKET_TYPE BOOKING_ID
--------- ------ ------------ ----------- ----------
TK0049    CN0000      1000.00 NORMAL      -
TK0050    CN0001       100.00 NORMAL      BK0000
```

ii. 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'@
```

```
TICKET_ID CON_ID TICKET_PRICE TICKET_TYPE BOOKING_ID
--------- ------ ------------ ----------- ----------
TK0029    CN0001       200.00 VIP         BK0001
TK0030    CN0001       100.00 NORMAL      BK0001
```

iii. 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'@
```

```
TICKET_ID CON_ID TICKET_PRICE TICKET_TYPE BOOKING_ID
--------- ------ ------------ ----------- ----------
TK0029    CN0001       200.00 VIP         BK0001
```

iv. 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'@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0010	CN0001	200.00	VIP	BK0003
TK0011	CN0001	200.00	VIP	BK0003
TK0012	CN0001	200.00	VIP	BK0003
TK0013	CN0001	200.00	VIP	BK0003
TK0014	CN0001	200.00	VIP	BK0003
TK0015	CN0001	200.00	VIP	BK0003
TK0016	CN0001	200.00	VIP	BK0003
TK0017	CN0001	200.00	VIP	BK0003
TK0018	CN0001	200.00	VIP	BK0003
TK0019	CN0001	200.00	VIP	BK0003
TK0020	CN0001	200.00	VIP	BK0003
TK0021	CN0001	200.00	VIP	BK0003
TK0022	CN0001	200.00	VIP	BK0003
TK0023	CN0001	200.00	VIP	BK0003
TK0024	CN0001	200.00	VIP	BK0003
TK0025	CN0001	200.00	VIP	BK0003

v. 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' )@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0000	CN0001	200.00	VIP	BK0004

vi. 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%' ) )@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0001	CN0000	5000.00	VIP	-
TK0001	CN0001	200.00	VIP	BK0005

vii. 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'@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0000	CN0002	200.00	VIP	BK0006
TK0001	CN0002	200.00	VIP	BK0006
TK0002	CN0002	200.00	VIP	BK0006
TK0003	CN0002	200.00	VIP	BK0006
TK0004	CN0002	200.00	VIP	BK0006
TK0005	CN0002	200.00	VIP	BK0006

viii. 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'@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0030	CN0001	100.00	NORMAL	BK0001
TK0031	CN0001	100.00	NORMAL	BK0007
TK0032	CN0001	100.00	NORMAL	BK0007
TK0033	CN0001	100.00	NORMAL	BK0007
TK0034	CN0001	100.00	NORMAL	BK0007
TK0035	CN0001	100.00	NORMAL	BK0007

ix. 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@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0025	CN0001	200.00	VIP	BK0004
TK0026	CN0001	200.00	VIP	BK0008
TK0027	CN0001	200.00	VIP	BK0008
TK0028	CN0001	200.00	VIP	BK0008
TK0029	CN0001	200.00	VIP	BK0001

x. 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' )@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0040	CN0001	100.00	NORMAL	BK0009
TK0041	CN0001	100.00	NORMAL	BK0009
TK0042	CN0001	100.00	NORMAL	BK0009
TK0043	CN0001	100.00	NORMAL	BK0009
TK0044	CN0001	100.00	NORMAL	BK0009

Data Manipulation Language (DML):

1. Triggers
    i. **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'@
```

THEATRE_ID	THEATRE_NAME	THEATRE_VENUE	SEAT_CAPACITY	VIP_COUNT	ZONE_COUNT
TT0002	Grand Hall	MMU Cyberjaya	50	50	6

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.

ii. 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'@*

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0000	CN0000	5000.00	VIP	-
TK0001	CN0000	5000.00	VIP	-
TK0002	CN0000	5000.00	VIP	-
TK0003	CN0000	5000.00	VIP	-
TK0004	CN0000	5000.00	VIP	-
TK0005	CN0000	5000.00	VIP	-
TK0006	CN0000	5000.00	VIP	-
TK0007	CN0000	5000.00	VIP	-
TK0008	CN0000	5000.00	VIP	-
TK0009	CN0000	5000.00	VIP	-
TK0010	CN0000	1000.00	NORMAL	-
TK0011	CN0000	1000.00	NORMAL	-
TK0012	CN0000	1000.00	NORMAL	-
TK0013	CN0000	1000.00	NORMAL	-
TK0014	CN0000	1000.00	NORMAL	-
TK0015	CN0000	1000.00	NORMAL	-
TK0016	CN0000	1000.00	NORMAL	-
TK0017	CN0000	1000.00	NORMAL	-
TK0018	CN0000	1000.00	NORMAL	-
TK0019	CN0000	1000.00	NORMAL	-
TK0020	CN0000	1000.00	NORMAL	-
TK0021	CN0000	1000.00	NORMAL	-
TK0022	CN0000	1000.00	NORMAL	-
TK0023	CN0000	1000.00	NORMAL	-
TK0024	CN0000	1000.00	NORMAL	-
TK0025	CN0000	1000.00	NORMAL	-
TK0026	CN0000	1000.00	NORMAL	-
TK0027	CN0000	1000.00	NORMAL	-
TK0028	CN0000	1000.00	NORMAL	-
TK0029	CN0000	1000.00	NORMAL	-
TK0030	CN0000	1000.00	NORMAL	-
TK0031	CN0000	1000.00	NORMAL	-

```
TK0032    CN0000      1000.00 NORMAL      -
TK0033    CN0000      1000.00 NORMAL      -
TK0034    CN0000      1000.00 NORMAL      -
TK0035    CN0000      1000.00 NORMAL      -
TK0036    CN0000      1000.00 NORMAL      -
TK0037    CN0000      1000.00 NORMAL      -
TK0038    CN0000      1000.00 NORMAL      -
TK0039    CN0000      1000.00 NORMAL      -
TK0040    CN0000      1000.00 NORMAL      -
TK0041    CN0000      1000.00 NORMAL      -
TK0042    CN0000      1000.00 NORMAL      -
TK0043    CN0000      1000.00 NORMAL      -
TK0044    CN0000      1000.00 NORMAL      -
TK0045    CN0000      1000.00 NORMAL      -
TK0046    CN0000      1000.00 NORMAL      -
TK0047    CN0000      1000.00 NORMAL      -
TK0048    CN0000      1000.00 NORMAL      -
TK0049    CN0000      1000.00 NORMAL      -

  50 record(s) selected.
```

*SELECT * FROM VIP_TICKET WHERE Con_ID = 'CN0000'@*

```
TICKET_ID CON_ID SEAT_NO
--------- ------ -------
TK0000    CN0000 V0000
TK0001    CN0000 V0001
TK0002    CN0000 V0002
TK0003    CN0000 V0003
TK0004    CN0000 V0004
TK0005    CN0000 V0005
TK0006    CN0000 V0006
TK0007    CN0000 V0007
TK0008    CN0000 V0008
TK0009    CN0000 V0009

  10 record(s) selected.
```

*SELECT * FROM NORMAL_TICKET WHERE Con_ID = 'CN0000'@*

```
TICKET_ID CON_ID ZONE_NO
--------- ------ -------
TK0010    CN0000 A
TK0011    CN0000 A
TK0012    CN0000 A
TK0013    CN0000 A
TK0014    CN0000 A
TK0015    CN0000 A
TK0016    CN0000 A
TK0017    CN0000 B
TK0018    CN0000 B
TK0019    CN0000 B
TK0020    CN0000 B
TK0021    CN0000 B
TK0022    CN0000 B
TK0023    CN0000 B
TK0024    CN0000 C
TK0025    CN0000 C
TK0026    CN0000 C
TK0027    CN0000 C
TK0028    CN0000 C
TK0029    CN0000 C
TK0030    CN0000 C
TK0031    CN0000 D
TK0032    CN0000 D
TK0033    CN0000 D
TK0034    CN0000 D
TK0035    CN0000 D
TK0036    CN0000 D
TK0037    CN0000 E
TK0038    CN0000 E
TK0039    CN0000 E
TK0040    CN0000 E
TK0041    CN0000 E
TK0042    CN0000 E
TK0043    CN0000 E
TK0044    CN0000 F
TK0045    CN0000 F
TK0046    CN0000 F
TK0047    CN0000 F
TK0048    CN0000 F
TK0049    CN0000 F

  40 record(s) selected.
```

iii. 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'@
```

BOOKING_ID	CUS_ID	AGENT_ID	BOOKING_TIME	TOTAL_PRICE
BK0000	CS0000	AG0000	2017-02-05-04.51.01.610000	100.00

iv. 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'@
```

TICKET_ID	CON_ID	TICKET_PRICE	TICKET_TYPE	BOOKING_ID
TK0000	CN0000	5000.00	VIP	-
TK0029	CN0001	200.00	VIP	BK0001

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.

v. 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@
```

BOOKING_ID	CUS_ID	AGENT_ID	BOOKING_TIME	TOTAL_PRICE
BK0000	CS0000	AG0000	2017-02-05-04.51.01.610000	100.00
BK0001	CS0004	AG0006	2017-02-05-04.51.01.632000	300.00
BK0002	CS0005	AG0006	2017-02-05-04.51.01.651000	0.00
BK0003	CS0004	AG0003	2017-02-05-04.51.01.674000	3200.00
BK0004	CS0003	AG0005	2017-02-05-04.51.01.692000	3400.00
BK0005	CS0007	AG0002	2017-02-05-04.51.01.716000	200.00
BK0006	CS0006	AG0005	2017-02-05-04.51.01.720000	1200.00
BK0007	CS0003	AG0004	2017-02-05-04.51.01.728000	500.00
BK0008	CS0002	AG0001	2017-02-05-04.51.01.732000	600.00
BK0009	CS0001	AG0002	2017-02-05-04.51.01.741000	500.00
BK0010	CS0004	AG0006	2017-02-05-04.51.01.745000	0.00

2. Views

i. 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.

AGENT_FNAME	AGENT_LNAME	NORMAL_SOLD	VIP_SOLD	TOTAL_SOLD
BILL	GATES	0	3	3
ELON	MUSK	1	0	1
HEDY	LAMARR	1	1	2
NEIL	TYSON	5	1	6
OPRAH	WINFREY	0	23	23
WARREN	BUFFET	5	0	5

ii. 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 )@
```

CUS_FNAME	CUS_LNAME	NORMAL_BOUGHT	VIP_BOUGHT	TOTAL_TICKET_BOUGHT
ABBY	LOW	0	1	1
CHRISTOPHER	TOO WEI BIN	5	0	5
GOH	KUN SHUN	1	0	1
JOHN	ESCOBIA	0	3	3
NG	JING KEONG	5	17	22
ONG	SHU YU	1	1	2
TEE	WEI WEI	0	6	6

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

iii. 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.

CON_TITLE	NORMAL_AVAILABLE	VIP_AVAILABLE
Maroon 5 Genting	58	8
Party Rock!	0	44
The Invincible	40	10

iv. 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.

CON_TITLE	TICKETS_SOLD	SEAT_CAPACITY	SOLD_PERCENT
Maroon 5 Genting	34	100	34.00
Party Rock!	6	50	12.00

v. 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.

```
CON_TITLE           NORMAL_SOLD NORMAL_CAPACITY SOLD_PERCENT
------------------- ----------- --------------- ------------
Maroon 5 Genting             12              70        17.14
```

vi. 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.

```
CON_TITLE            VIP_SOLD    VIP_CAPACITY SOLD_PERCENT
-------------------- ---------- ------------ ------------
Maroon 5 Genting             22           30        73.33
Party Rock!                   6           50        12.00
```

3. Stored Procedures

   i. 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:

```
db2 => call spTop5Customers( '01/27/2017','12/31/2017' )@


  Result set 1
  --------------

  CUS_FNAME       CUS_LNAME       TICKETS_BOUGHT
  --------------- --------------- --------------
  ONG             SHU YU                      18
  NG              JING KEONG                   6
  TEE             WEI WEI                      6
  CHRISTOPHER     TOO WEI BIN                  5
  JOHN            ESCOBIA                      3

  5 record(s) selected.

  Return Status = 0
```

ii. 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:

```
db2 => call spTop5Agents( '01/27/2017','12/31/2017' )@

  Result set 1
  --------------

  AGENT_FNAME     AGENT_LNAME     TICKETS_SOLD
  --------------- --------------- ------------
  STEPHEN         HAWKING                   16
  OPRAH           WINFREY                    7
  NEIL            TYSON                      6
  WARREN          BUFFET                     5
  BILL            GATES                      3

  5 record(s) selected.

  Return Status = 0
```

iii. 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:

```
db2 => call spTop2Concerts( '01/27/2017','12/31/2017' )@


  Result set 1
  --------------

  CON_TITLE            TICKETS_SOLD
  -------------------- ------------
  Maroon 5 Genting                34
  Party Rock!                      6

  2 record(s) selected.

  Return Status = 0
```

iv. 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:

```
db2 => call spTheatreSchedule( 'TT0000', '01/27/2017'  )@

  Result set 1
  --------------

  CON_TITLE            CON_TIMESTART CON_TIMEEND
  -------------------- ------------- -----------
  The Invincible       20:00:00      23:00:00

  1 record(s) selected.

  Return Status = 0
```

v. spAvailableTickets

Displays the tickets available for a certain concert.

Creation:

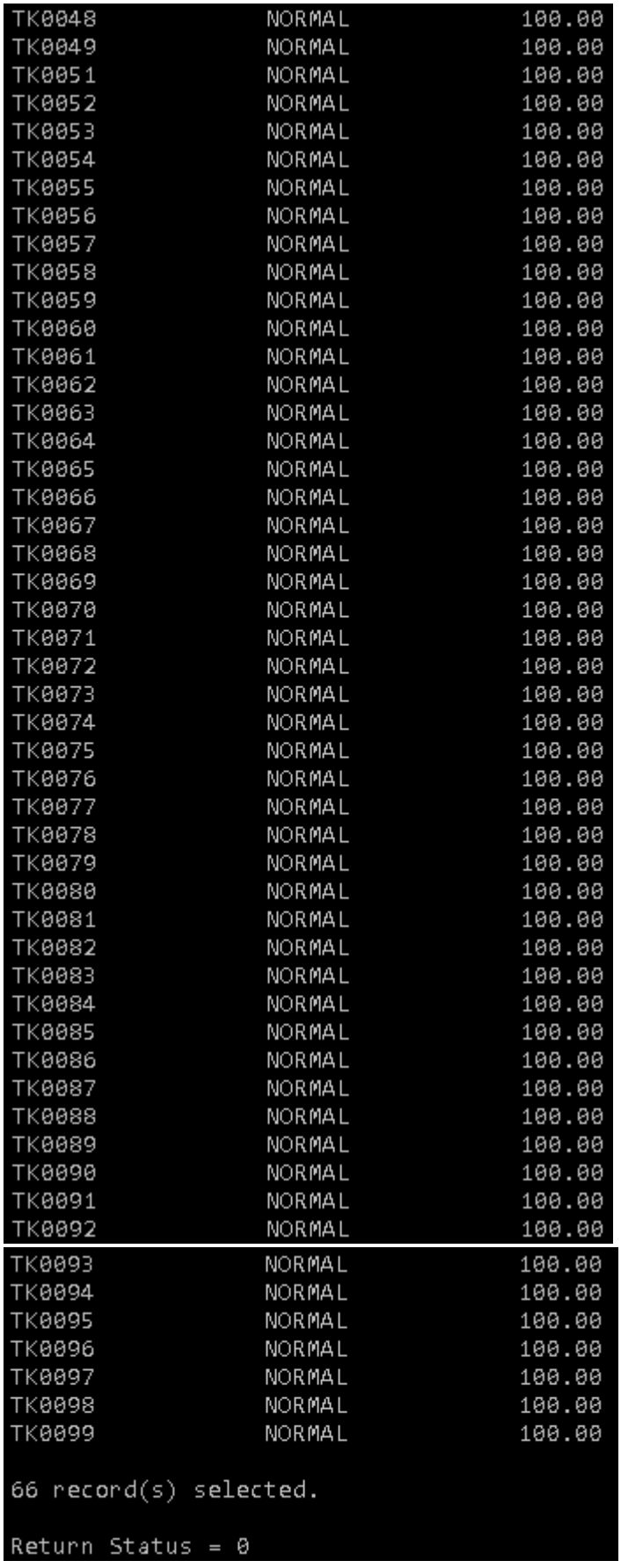
```
CREATE PROCEDURE spAvailableTickets( IN concertID CHAR( 6 ) )
  BEGIN
    DECLARE c cursor with return for
      SELECT DISTINCT TICKET.TICKET_ID AS AVAILABLE_TICKETS,
      TICKET.TICKET_TYPE, TICKET.TICKET_PRICE
      FROM TICKET, BOOKING, CONCERT
      WHERE TICKET.BOOKING_ID IS NULL
      AND concertID = CONCERT.CON_ID
      AND concertID = TICKET.CON_ID
      ORDER BY TICKET.TICKET_ID;
    OPEN c;
  END@
```

Call and output:

```
db2 => call spAvailableTickets( 'CN0001' )@


  Result set 1
  --------------

  AVAILABLE_TICKETS TICKET_TYPE TICKET_PRICE
  ----------------- ----------- ------------
  TK0002            VIP                200.00
  TK0003            VIP                200.00
  TK0004            VIP                200.00
  TK0005            VIP                200.00
  TK0006            VIP                200.00
  TK0007            VIP                200.00
  TK0008            VIP                200.00
  TK0009            VIP                200.00
  TK0036            NORMAL             100.00
  TK0037            NORMAL             100.00
  TK0038            NORMAL             100.00
  TK0039            NORMAL             100.00
  TK0045            NORMAL             100.00
  TK0046            NORMAL             100.00
  TK0047            NORMAL             100.00
```



```
TK0048          NORMAL          100.00
TK0049          NORMAL          100.00
TK0051          NORMAL          100.00
TK0052          NORMAL          100.00
TK0053          NORMAL          100.00
TK0054          NORMAL          100.00
TK0055          NORMAL          100.00
TK0056          NORMAL          100.00
TK0057          NORMAL          100.00
TK0058          NORMAL          100.00
TK0059          NORMAL          100.00
TK0060          NORMAL          100.00
TK0061          NORMAL          100.00
TK0062          NORMAL          100.00
TK0063          NORMAL          100.00
TK0064          NORMAL          100.00
TK0065          NORMAL          100.00
TK0066          NORMAL          100.00
TK0067          NORMAL          100.00
TK0068          NORMAL          100.00
TK0069          NORMAL          100.00
TK0070          NORMAL          100.00
TK0071          NORMAL          100.00
TK0072          NORMAL          100.00
TK0073          NORMAL          100.00
TK0074          NORMAL          100.00
TK0075          NORMAL          100.00
TK0076          NORMAL          100.00
TK0077          NORMAL          100.00
TK0078          NORMAL          100.00
TK0079          NORMAL          100.00
TK0080          NORMAL          100.00
TK0081          NORMAL          100.00
TK0082          NORMAL          100.00
TK0083          NORMAL          100.00
TK0084          NORMAL          100.00
TK0085          NORMAL          100.00
TK0086          NORMAL          100.00
TK0087          NORMAL          100.00
TK0088          NORMAL          100.00
TK0089          NORMAL          100.00
TK0090          NORMAL          100.00
TK0091          NORMAL          100.00
TK0092          NORMAL          100.00
TK0093          NORMAL          100.00
TK0094          NORMAL          100.00
TK0095          NORMAL          100.00
TK0096          NORMAL          100.00
TK0097          NORMAL          100.00
TK0098          NORMAL          100.00
TK0099          NORMAL          100.00

66 record(s) selected.

Return Status = 0
```

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