

MULTIMEDIA UNIVERSITY ®

TIS1101 Database Fundamentals

Group Assignment

Title: 5. Online Music Store

Prepared by:  
Leader: 1142701684 NG CHIN ANN [chinann6213@gmail.com](mailto:chinann6213@gmail.com)

Member: 1142701883 KHOR KIA KIN [0jetpistol0@gmail.com](mailto:0jetpistol0@gmail.com)

Member: 1142701444 TEY MENG KIAT [manjorney@gmail.com](mailto:manjorney@gmail.com)

Date: 3 FEBRUARY 2016

Table of Contents

[Business Rules 1](#_Toc441957651)

[Entity-Relationship Diagram](file:///C:\Users\NC%20Ann\Desktop\TIS1101%20Database%20Fundamentals.docx#_Toc441957652) 2

[Data Dictionary 3](#_Toc441957653)

[Data Definition Command (DDL) 5](#_Toc441957654)

[Date Insertion 7](#_Toc441957655)

[Data Manipulation Command (DML) 16](#_Toc441957656)

[(a) Aggregate Funtions 16](#_Toc441957657)

[(b) “Group by” with “having” clause. 17](#_Toc441957658)

[(c) Nested Query/ Sub Query 18](#_Toc441957659)

[(d) Views 19](#_Toc441957660)

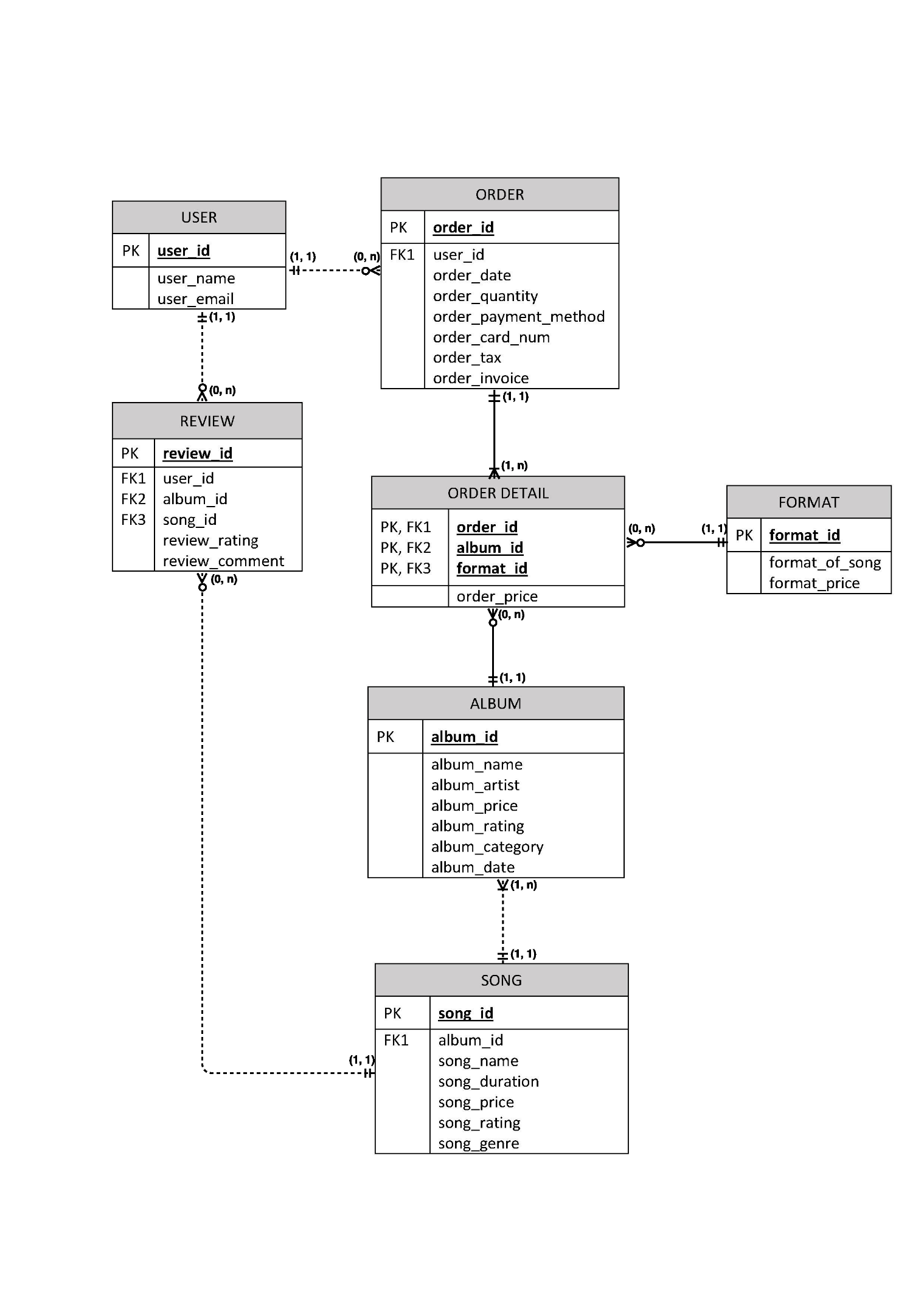
[(e) Trigger 20](#_Toc441957661)

[(f) Stored procedure 26](#_Toc441957662)

[(g) Additional query component 28](#_Toc441957663)

# Business Rules

1. A unique username and a valid email address are needed to register a user account.
2. An email address can only be used to make one account.
3. An account is needed to purchase an album.
4. An account can have zero or more than one purchase.
5. User can use many payment methods.
6. An order must be associated with at least one album.
7. An album must consist of its album name, artist name, release date, album format and price.
8. A song must consists of its song name, artist name, duration and genre.
9. Every album must be categorized to new, trending or null.
10. Rating for a song or an album is based on user.



# Entity-Relationship Diagram

# Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table Name | Attribute Name | Contents/Description | Data Type | Format | Range | PK/ FK | FK Referenced Table |
| USER | user\_id  user\_name  user\_email | User ID  User Name  User Email | INT  VARCHAR(20)  VARCHAR(30) | #  Xxxxx  Xxxxxx | 1 – n | PK |  |
| ALBUM | album\_id  album\_name  album\_artist  album\_price  album\_rating  album\_category  album\_date | Album ID  Album Name  Album Artist  Album Price  Album Rating  Album Category  Album Date | INT  VARCHAR(30)  VARCHAR(30)  DECIMAL(5,2)  DECIMAL(2,1)  VARCHAR(20)  DATE | #  Xxxxxx  Xxxxxx  ###.##  #.#  Xxxxx  YYYY-MM-DD | 1 – n | PK |  |
| SONG | song\_id  song\_name  song\_duration  song\_price  song\_rating  song\_genre  album\_id | Song ID  Song Name  Song Duration  Song Price  Song Rating  Song Genre  Album ID | INT  VARCHAR(30)  CHAR(5)  DECIMAL(4,2)  DECIMAL(2,1)  VARCHAR(10)  INT | #  Xxxxxxx  xxxxx  ##.##  #.#  Xxxx  # | 1 – n | PK  FK1 | ALBUM |
| FORMAT | format\_id  format\_of\_song  format\_price | Format ID  Format of Song  Format Price | INT  CHAR(3)  DECIMAL(5,2) | #  Xxx  ###.## | 1 – 3 | PK |  |
| REVIEW | review\_id  user\_id  album\_id  song\_id  review\_rating  review\_comment | Review ID  User ID  Album ID  Song ID  Review Rating  Review Comment | INT  INT  INT  INT  DECIMAL(2,1)  VARCHAR(50) | #  #  #  #  #.#  Xxxxxxx | 1 – n | PK  FK1  FK2  FK3 | USER  ALBUM  SONG |
| ORDER | order\_id  order\_date  order\_quantity  order\_payment\_method  order\_card\_num  order\_tax  order\_invoice  user\_id | Order ID  Order Date  Order Quantity  Order Payment Method  Order Card Number  Order Tax Charged  Order Invoice  User ID | INT  DATE  INT  VARCHAR(20)  CHAR(16)  DECIMAL(3,2)  DECIMAL(7,2)  INT | #  YYYY-MM-DD  ##  Xxxxx  xxxxx  ##.##  #####.##  # | 1 – n | PK  FK1 | USER |
| ORDER\_DETAIL | order\_id  album\_id  format\_id  order\_price | Order ID  Album ID  Format ID  Order Price(Single album) | INT  INT  INT  DECIMAL(5,2) | #  #  #  ###.## | 1 – n | PK, FK1  PK, FK2  PK, FK3 | ORDER  ALBUM  FORMAT |

# Data Definition Command (DDL)

CREATE DATABASE msicstor;

CONNECT TO msicstor;

CREATE TABLE user

(

user\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

user\_name VARCHAR(20),

user\_email VARCHAR(30)

);

CREATE TABLE album

(

album\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

album\_name VARCHAR(30),

album\_artist VARCHAR(30),

album\_price DECIMAL(5,2) WITH DEFAULT 0,

album\_rating DECIMAL(2,1) WITH DEFAULT 0.0,

album\_category VARCHAR(20),

album\_date DATE,

CHECK (album\_rating BETWEEN 0.0 AND 5.0),

CHECK (album\_category IN ('New', 'Trending'))

);

CREATE TABLE song

(

song\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

song\_name VARCHAR(30),

song\_duration CHAR(5),

song\_price DECIMAL(4,2),

song\_rating DECIMAL(2,1) WITH DEFAULT 0.0,

song\_genre VARCHAR(10),

album\_id INT,

FOREIGN KEY (album\_id) REFERENCES album,

CHECK (song\_rating BETWEEN 0.0 AND 5.0),

CHECK (song\_genre IN ('Pop', 'Rock', 'Classical', 'Electric', 'Hip-Hop', 'Jazz', 'Karaoke', 'RnB'))

);

CREATE TABLE format

(

format\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

format\_of\_song CHAR(3),

format\_price DECIMAL(5,2),

CHECK (format\_of\_song IN ('mp3', 'wma', 'wav'))

);

CREATE TABLE review

(

review\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

user\_id INT,

album\_id INT,

song\_id INT,

review\_rating DECIMAL(2,1),

review\_comment VARCHAR(50),

FOREIGN KEY (user\_id) REFERENCES user,

FOREIGN KEY (album\_id) REFERENCES album,

FOREIGN KEY (song\_id) REFERENCES song,

CHECK (review\_rating BETWEEN 1.0 AND 5.0)

);

CREATE TABLE order

(

order\_id INT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

order\_date DATE,

order\_quantity INT,

order\_payment\_method VARCHAR(20),

order\_card\_num CHAR(16),

order\_tax DECIMAL(3,2),

order\_invoice DECIMAL(7,2),

user\_id INT,

FOREIGN KEY (user\_id) REFERENCES user,

CHECK (order\_payment\_method IN ('Visa', 'Master', 'Paypal', 'Maybank2u', 'CIMB Clicks'))

);

CREATE TABLE order\_detail

(

order\_id INT NOT NULL,

album\_id INT NOT NULL,

format\_id INT NOT NULL,

order\_price DECIMAL(5,2),

FOREIGN KEY (order\_id) REFERENCES order,

FOREIGN KEY (album\_id) REFERENCES album,

FOREIGN KEY (format\_id) REFERENCES format,

PRIMARY KEY (order\_id, album\_id, format\_id)

);

# Date Insertion

INSERT INTO user

(user\_name, user\_email)

VALUES

('John', 'john@gmail.com'),

('Constantine', 'constantine@hotmail.com'),

('Taylor', 'taylor@yahoomail.com'),

('Lucy', 'lucy@gmail.com'),

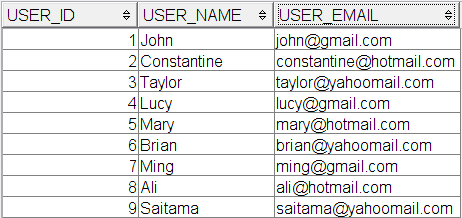
('Mary', 'mary@hotmail.com'),

('Brian', 'brian@yahoomail.com'),

('Ming', 'ming@gmail.com'),

('Ali', 'ali@hotmail.com'),

('Saitama', 'saitama@yahoomail.com');

SELECT \* FROM user;

INSERT INTO album

(album\_name, album\_artist, album\_category, album\_date)

VALUES

('25', 'Adele', 'New', '2015-11-20'),

('Purpose', 'Justin Bieber', 'New', '2015-11-13'),

('The Piano Guys', 'The Piano Guys', 'Trending', '2012-10-02'),

('The Marshall Mathers LP 2', 'Eminem', NULL, '2013-11-05'),

('Nothing But the Best', 'Frank Sinatra', 'Trending', '2008-05-13'),

('Thriller', 'Michael Jackson', 'Trending', '1983-07-29'),

('Confident', 'Justin Bieber', NULL, '2013-09-08'),

('Blackstar', 'David Bowie', 'New', '2016-01-08'),

('Girl On Fire', 'Alicia Keys', NULL, '2012-11-23'),

('Under The Mistletoe', 'Justin Bieber', 'Trending', '2011-04-18'),

('Love, Marriage & Divorce', 'Toni Braxton & Babyface', NULL, '2014-01-01'),

('Night Visions', 'Imagine Dragons', 'Trending', '2012-01-01'),

('Does This Look Infected?', 'Sum 41', 'Trending', '2002-01-01'),

('My World 2.0', 'Justin Bieber', NULL, '2010-05-13'),

('Believe', 'Justin Bieber', 'Trending', '2012-02-14');

SELECT \* FROM album;



INSERT INTO song

(album\_id, song\_name, song\_duration, song\_price, song\_genre)

VALUES

-- 25

(1, 'Hello', '04:55', 3.50, 'Pop'),

(1, 'When We Were Young', '04:50', 3.50, 'Pop'),

(1, 'Million Years Ago', '03:47', 3.50, 'Jazz'),

(1, 'All I Ask', '04:32', 3.50, 'Classical'),

-- Purpose

(2, 'I Will Show You', '03:25', 3.00, 'Pop'),

(2, 'What Do You Mean?', '03:25', 3.00, 'Hip-Hop'),

(2, 'Sorry', '03:20', 3.00, 'RnB'),

(2, 'Love Yourself', '03:53', 3.00, 'Pop'),

-- The Piano Guys

(3, 'Over the Rainbow, Simple Gifts', '03:44', 4.00, 'Classical'),

(3, 'Beethoven 5 Secrets', '05:09', 4.00, 'Classical'),

(3, 'A Thousand Years', '04:36', 4.00, 'Classical'),

(3, 'Peponi', '04:10', 4.00, 'Classical'),

-- The Marshall Mathers LP 2

(4, 'The Monster', '04:10', 3.00, 'Hip-Hop'),

(4, 'Rap God', '06:03', 3.00, 'Electric'),

(4, 'Berzerk', '03:58', 3.00, 'Rock'),

-- Nothing But the Best

(5, 'Fly Me To the Moon', '02:27', 3.50, 'Jazz'),

(5, 'My Way', '04:36', 3.50, 'Jazz'),

(5, 'Strangers In the Night', '02:44', 3.50, 'Jazz'),

-- Thriller

(6, 'Thriller', '05:57', 4.00, 'RnB'),

(6, 'Beat It', '04:18', 4.00, 'Hip-Hop'),

(6, 'Billie Jean', '04:54', 4.00, 'Jazz'),

-- Confident

(7, 'Confident', '04:22', 4.00, 'Hip-Hop'),

-- Blackstar

(8, 'Blackstar', '09:57', 3.50, 'Rock'),

(8, 'Lazarus', '06:22', 3.50, 'Electric'),

(8, 'Dollar Days', '04:44', 3.50, 'RnB'),

(8, 'I Cannot Give Everything Away', '05:47', 3.50, 'Hip-Hop'),

-- Girl On Fire

(9, 'Girl On Fire', '04:30', 4.00, 'RnB'),

(9, 'Tears Always Win', '03:59', 4.00, 'RnB'),

(9, 'Brand New Me', '03:53', 4.00, 'RnB'),

-- Under The Mistletoe

(10, 'Fa La La', '03:55', 3.00, 'RnB'),

-- Love, Marriage & Divorce

(11, 'Hurt You', '04:10', 3.00, 'RnB'),

(11, 'Where Did We Go Wrong', '03:37', 3.00, 'Rock'),

(11, 'Reunited', '03:18', 3.00, 'Electric'),

(11, 'I Wish', '03:03', 3.00, 'RnB'),

-- Night Visions

(12, 'Radioactive', '03:06', 3.50, 'Rock'),

(12, 'Demons', '02:57', 3.50, 'Rock'),

(12, 'Its Time', '04:00', 3.50, 'Classical'),

-- Does This Look Infected?

(13, 'The Hell Song', '03:18', 4.00, 'Rock'),

(13, 'Over My Head, Better Off Dead', '02:29', 4.00, 'Rock'),

(13, 'Still Waiting', '02:38', 4.00, 'Rock'),

(13, 'Thanks for Nothing', '03:04', 4.00, 'Rock'),

-- My World 2.0

(14, 'Where Are You Now', '03:45', 3.00, 'RnB'),

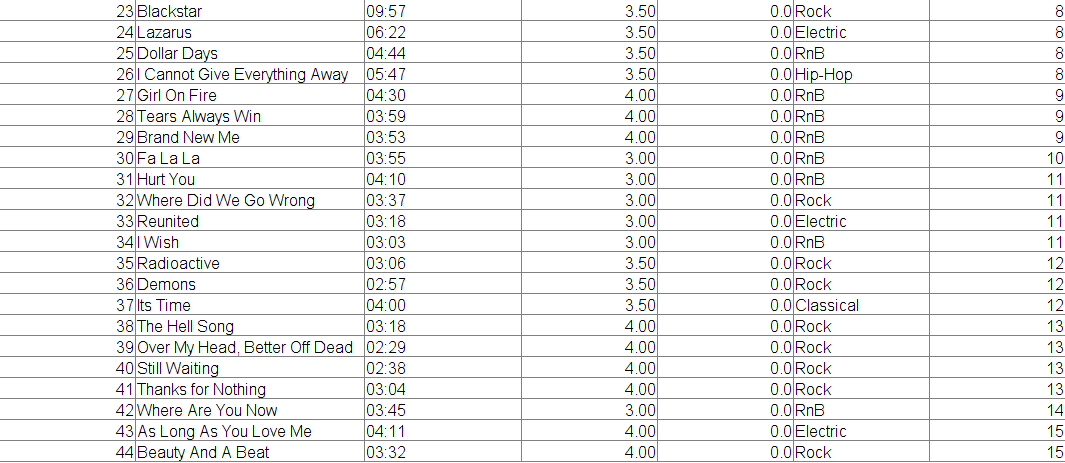
-- Believe

(15, 'As Long As You Love Me', '04:11', 4.00, 'Electric'),

(15, 'Beauty And A Beat', '03:32', 4.00, 'Rock');

SELECT \* FROM song;





INSERT INTO format

(format\_of\_song, format\_price)

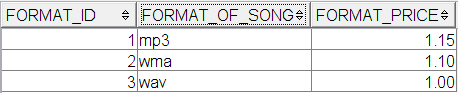
VALUES

('mp3', 1.15),

('wma', 1.10),

('wav', 1.00);

SELECT \* FROM format;



-- album\_rating

INSERT INTO review

(user\_id, album\_id, review\_rating, review\_comment)

VALUES

(1, 12, 4.0, 'Great album, love it.'),

(7, 9, 5.0, 'One of the best album!'),

(4, 1, 1.0, 'Worst album ever!'),

(6, 11, 4.0, 'Awesome~~'),

(7, 9, 3.0, 'So so..'),

(1, 2, 4.0, 'Like it!'),

(2, 2, 4.5, 'Well done!'),

(5, 3, 2.5, 'A waste of money!'),

(8, 11, 3.0, 'OK...'),

(7, 2, 4.2, 'One of the best albums!'),

(4, 7, 3.8, 'Not bad.'),

(1, 4, 4.1, 'Just like it.'),

(2, 5, 5.0, 'Worth it!'),

(4, 5, 4.8, 'Nice album for collection.'),

(6, 9, 4.1, 'What an album!'),

(2, 6, 3.6, 'Still has space to improve.');

-- song\_rating

INSERT INTO review

(user\_id, song\_id, review\_rating, review\_comment)

VALUES

(3, 36, 3.0, 'This song is epic.'),

(6, 23, 4.0, 'Addictive song.'),

(6, 9, 5.0, 'This song releases my soul!'),

(2, 5, 2.0, 'Lame song.'),

(7, 18, 5.0, 'It gives me goosebumps!'),

(2, 3, 4.7, 'My favourite song!'),

(9, 19, 5.0, 'A must listen song this year!'),

(5, 18, 4.5, 'Addicted to this song.'),

(4, 11, 3.0, 'So so...'),

(1, 4, 2.0, 'Waste of money...'),

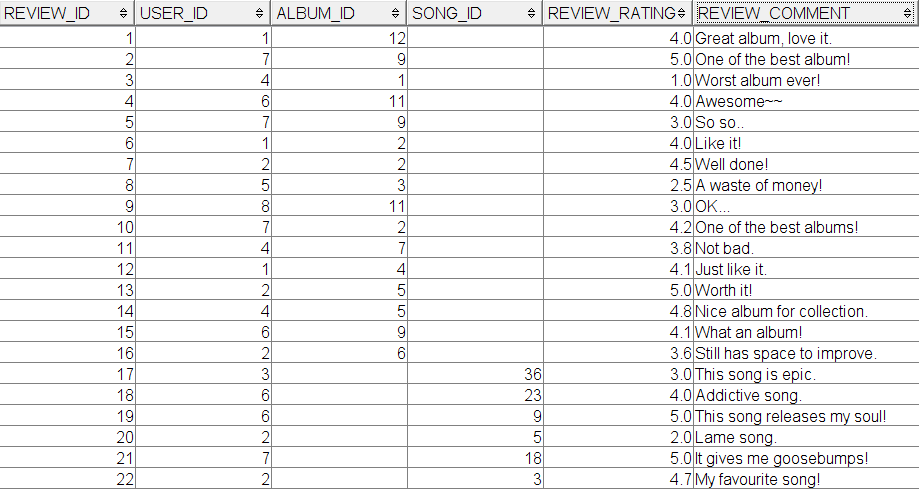
(1, 9, 4.2, 'Like the chorus part!'),

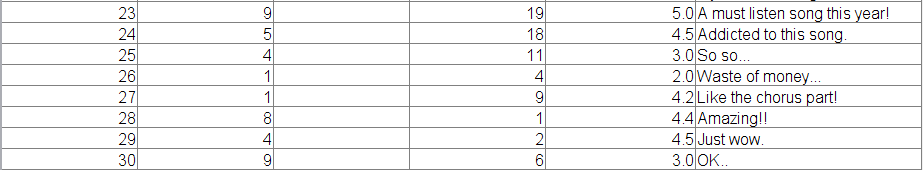
(8, 1, 4.4, 'Amazing!!'),

(4, 2, 4.5, 'Just wow.'),

(9, 6, 3.0, 'OK..');

SELECT \* FROM review;





INSERT INTO order

(order\_date, order\_quantity, order\_payment\_method, order\_card\_num, order\_tax, user\_id)

VALUES

('2012-02-04', 4, 'Master', '5571200428576225', 1.06, 9),

('2015-01-29', 3, 'Master', '1568692387368551', 1.06, 5),

('2014-02-01', 1, 'Master', '4388997619380775', 1.06, 9),

('2015-05-21', 2, 'Visa', '5912626632056525', 1.06, 4),

('2011-09-20', 5, 'CIMB Clicks', '0391325125792567', 1.06, 8),

('2013-06-02', 3, 'CIMB Clicks', '2108903411664564', 1.06, 2),

('2012-06-04', 3, 'Maybank2u', '1665777678804642', 1.06, 6),

('2015-09-20', 2, 'Maybank2u', '1384840526804208', 1.06, 1),

('2016-01-08', 3, 'Visa', '5869674929411283', 1.06, 3),

('2014-07-01', 5, 'Paypal', '7593688480304381', 1.06, 6),

('2015-05-22', 3, 'Master', '5571200428576225', 1.06, 9),

('2015-09-09', 4, 'Visa', '5912626632056525', 1.06, 4),

('2015-12-12', 2, 'CIMB Clicks', '0391325125792567', 1.06, 8),

('2016-01-10', 3, 'Maybank2u', '1665777678804642', 1.06, 6),

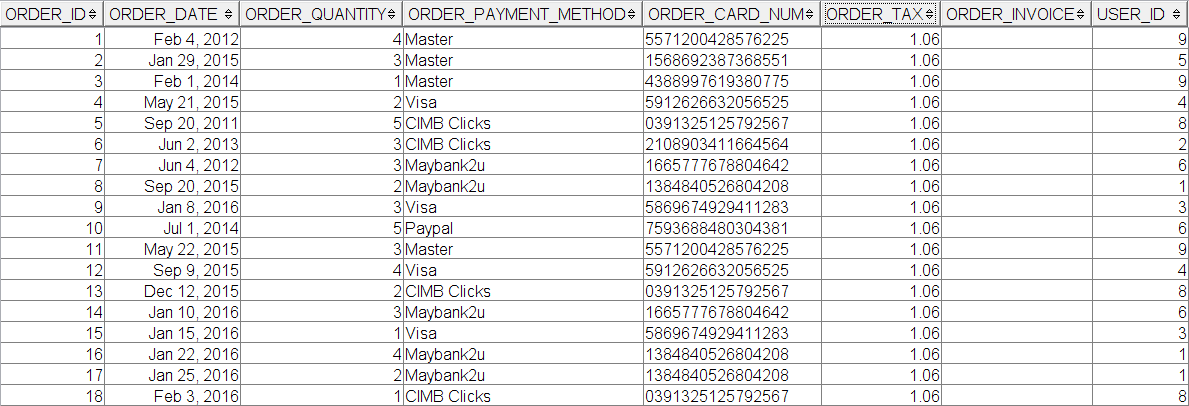
('2016-01-15', 1, 'Visa', '5869674929411283', 1.06, 3),

('2016-01-22', 4, 'Maybank2u', '1384840526804208', 1.06, 1),

('2016-01-25', 2, 'Maybank2u', '1384840526804208', 1.06, 1),

('2016-02-03', 1, 'CIMB Clicks', '0391325125792567', 1.06, 8);

SELECT \* FROM order;



INSERT INTO order\_detail

(order\_id, album\_id, format\_id)

VALUES

-- order\_id = 1

(1, 3, 1),

(1, 6, 2),

(1, 7, 1),

(1, 9, 1),

-- order\_id = 2

(2, 4, 2),

(2, 6, 1),

(2, 5, 2),

-- order\_id = 3

(3, 2, 2),

-- order\_id = 4

(4, 2, 2),

(4, 10, 3),

-- order\_id = 5

(5, 1, 1),

(5, 2, 1),

(5, 4, 3),

(5, 9, 2),

(5, 11, 2),

-- order\_id = 6

(6, 5, 3),

(6, 6, 1),

(6, 8, 2),

-- order\_id = 7

(7, 3, 1),

(7, 6, 2),

(7, 7, 2),

-- order\_id = 8

(8, 6, 2),

(8, 9, 2),

-- order\_id = 9

(9, 2, 1),

(9, 3, 1),

(9, 5, 2),

-- order\_id = 10

(10, 1, 1),

(10, 5, 1),

(10, 6, 1),

(10, 9, 3),

(10, 10, 2),

-- order\_id = 11

(11, 5, 1),

(11, 6, 1),

(11, 2, 2),

-- order\_id = 12

(12, 3, 3),

(12, 8, 2),

(12, 7, 2),

(12, 7, 1),

-- order\_id = 13

(13, 3, 1),

(13, 8, 3),

-- order\_id = 14

(14, 5, 2),

(14, 1, 3),

(14, 3, 2),

-- order\_id = 15

(15, 8, 1),

-- order\_id = 16

(16, 7, 2),

(16, 4, 3),

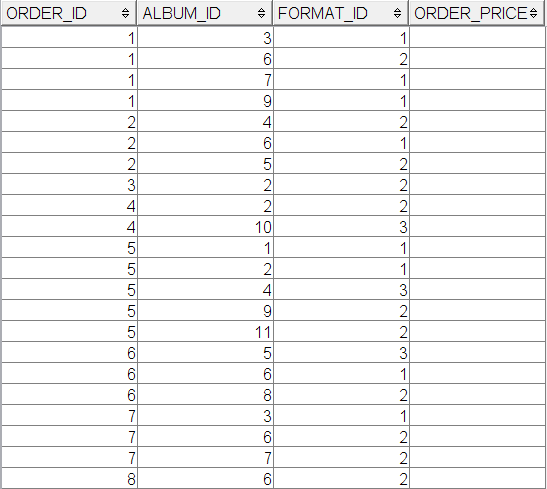
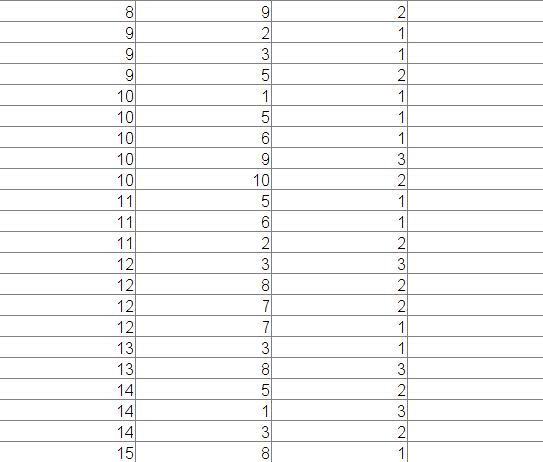
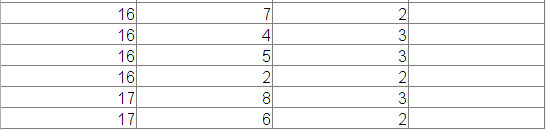
(16, 5, 3),

(16, 2, 2),

-- order\_id = 17

(17, 8, 3),

(17, 6, 2);

SELECT \* FROM order\_detail;

# Data Manipulation Command (DML)

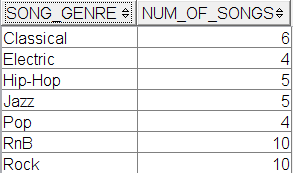
## (a) Aggregate Funtions

(i) List every song genre and number of songs associated with each.

SELECT song\_genre, COUNT(song\_id) AS num\_of\_songs

FROM song

GROUP BY song\_genre;



(ii) List albums' popularity by purchase rate.

SELECT album\_name, (CAST(((COUNT(order\_detail.album\_id))/

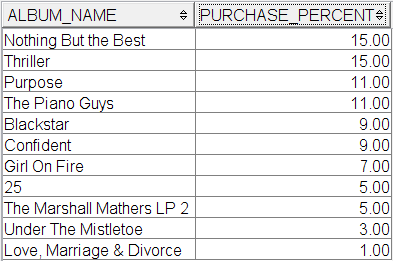
(SELECT CAST(COUNT(album\_id) AS DECIMAL(4,2)) FROM order\_detail)) AS DECIMAL(4,2))\* 100) AS purchase\_percent

FROM order\_detail, album

WHERE order\_detail.album\_id = album.album\_id

GROUP BY album\_name

ORDER BY purchase\_percent DESC;



## (b) “Group by” with “having” clause.

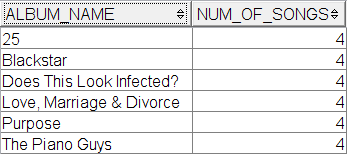
(i) List albums with four songs.

SELECT album\_name, COUNT(song\_id) AS num\_of\_songs

FROM album, song

WHERE album.album\_id = song.album\_id

GROUP BY album\_name

 HAVING COUNT(song\_id) = 4;

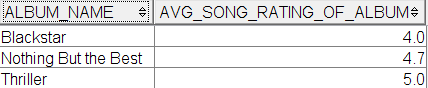
(ii) List albums with its average song rating higher than or equal to four.

SELECT album\_name, CAST(ROUND(AVG(song\_rating), 2) AS DECIMAL(2,1)) AS AVG\_song\_rating\_of\_album

FROM album, song

WHERE album.album\_id = song.album\_id

GROUP BY album\_name

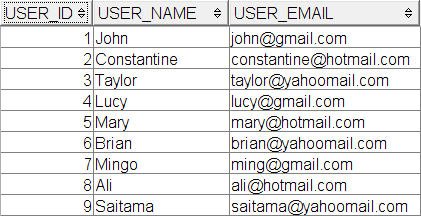
 HAVING AVG(song\_rating) >= 4;

## (c) Nested Query/ Sub Query

(i) Update username.

UPDATE user

SET user\_name = 'Mingo'

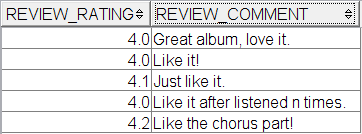
WHERE user\_id = (SELECT user\_id FROM user WHERE user\_name = 'Ming');

(ii) Update review rating and comment.

UPDATE review

SET review\_rating = 4.0, review\_comment = 'Like it after listened n times.'

WHERE user\_id = (SELECT user\_id FROM user WHERE user\_name = 'John')

AND song\_id = (SELECT song\_id FROM review WHERE song\_id =

## (d) Views

(i) List songs with rating higher than or equal to four.

CREATE VIEW top\_rate\_songs AS

SELECT song\_name, album\_name, song\_rating

FROM song, album

WHERE song.album\_id = album.album\_id

AND song\_rating >= 4;

SELECT \* FROM top\_rate\_songs;



(ii) List users with their total invoice.

CREATE VIEW transaction\_history AS

SELECT user\_name AS name, COUNT(album\_id) AS total\_album\_purchased, SUM(order\_invoice) AS total\_price

FROM order, user, order\_detail

WHERE user.user\_id = order.user\_id

AND order\_detail.order\_id = order.order\_id

GROUP BY user\_name;

SELECT \* FROM transaction\_history;



## (e) Trigger

(i) Update album\_price

CREATE TRIGGER update\_album\_price

AFTER INSERT ON song

FOR EACH ROW MODE DB2SQL

UPDATE album

SET album\_price =

(SELECT DISTINCT song\_price FROM song WHERE album.album\_id = song.album\_id) \*

(SELECT COUNT(song\_id) FROM song WHERE album.album\_id = song.album\_id);

CREATE TRIGGER update\_album\_price2

AFTER UPDATE OF song\_price ON song

FOR EACH ROW MODE DB2SQL

UPDATE album

SET album\_price =

(SELECT DISTINCT song\_price FROM song WHERE album.album\_id = song.album\_id) \*

(SELECT COUNT(song\_id) FROM song WHERE album.album\_id = song.album\_id);

UPDATE song

SET song\_price = 3.80

WHERE song.album\_id = 1;

SELECT \* FROM album;



(ii) Update song\_rating

CREATE TRIGGER update\_song\_rating

AFTER INSERT ON review

FOR EACH ROW MODE DB2SQL

UPDATE song

SET song\_rating =

(SELECT AVG(review\_rating) FROM review WHERE song.song\_id = review.song\_id);

CREATE TRIGGER update\_song\_rating2

AFTER UPDATE OF review\_rating ON review

FOR EACH ROW MODE DB2SQL

UPDATE song

SET song\_rating =

(SELECT AVG(review\_rating) FROM review WHERE song.song\_id = review.song\_id);

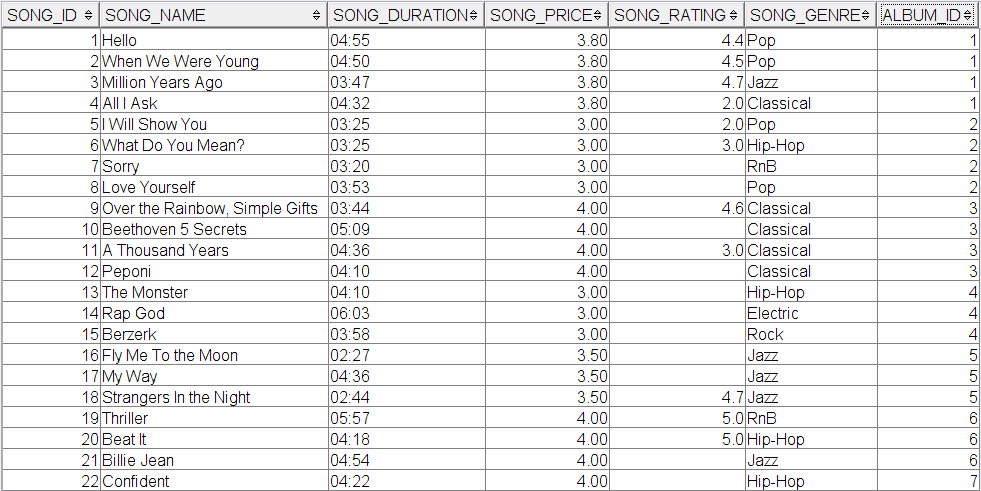
INSERT INTO review

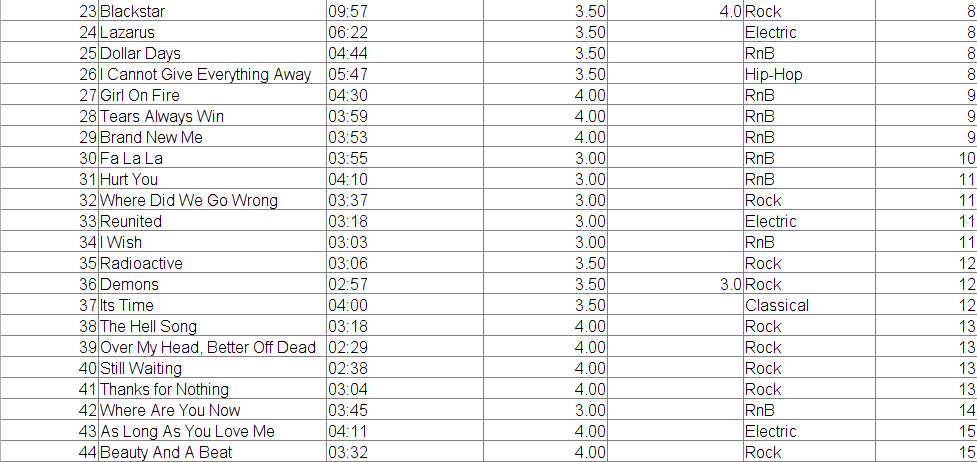
(user\_id, song\_id, review\_rating, review\_comment)

VALUES

(9, 20, 5.0, 'Legendary song!');

SELECT \* FROM song;





(iii) Update album\_rating

CREATE TRIGGER update\_album\_rating

AFTER INSERT ON review

FOR EACH ROW MODE DB2SQL

UPDATE album

SET album\_rating =

(SELECT AVG(review\_rating) FROM review WHERE album.album\_id = review.album\_id);

CREATE TRIGGER update\_album\_rating2

AFTER UPDATE OF review\_rating ON review

FOR EACH ROW MODE DB2SQL

UPDATE album

SET album\_rating =

(SELECT AVG(review\_rating) FROM review WHERE album.album\_id = review.album\_id);

INSERT INTO review

(user\_id, album\_id, review\_rating, review\_comment)

VALUES

(9, 6, 5.0, 'My favourite album!');

SELECT \* FROM album;



(iv) Update order\_price

CREATE TRIGGER update\_order\_price

AFTER INSERT ON order\_detail

FOR EACH ROW MODE DB2SQL

UPDATE order\_detail

SET order\_price =

(SELECT album\_price FROM album WHERE album.album\_id = order\_detail.album\_id) \*

(SELECT format\_price FROM format WHERE format.format\_id = order\_detail.format\_id);

(v) Update order\_invoice

CREATE TRIGGER update\_order\_invoice

AFTER INSERT ON order\_detail

FOR EACH ROW MODE DB2SQL

UPDATE order

SET order\_invoice =

order\_tax \*

(SELECT SUM(order\_price) FROM order\_detail WHERE order.order\_id = order\_detail.order\_id);

-- Note: Used in conjuction with procedure #1 discount

CREATE TRIGGER update\_order\_invoice2

AFTER UPDATE OF order\_tax ON order

FOR EACH ROW MODE DB2SQL

UPDATE order

SET order\_invoice =

order\_tax \*

(SELECT SUM(order\_price) FROM order\_detail WHERE order.order\_id = order\_detail.order\_id);

INSERT INTO order\_detail

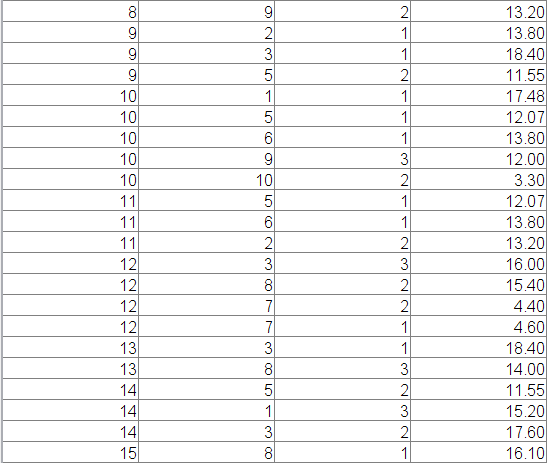
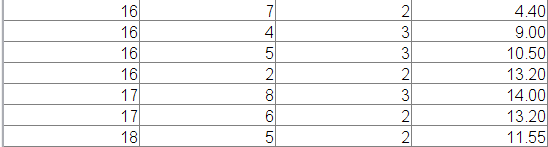
(order\_id, album\_id, format\_id)

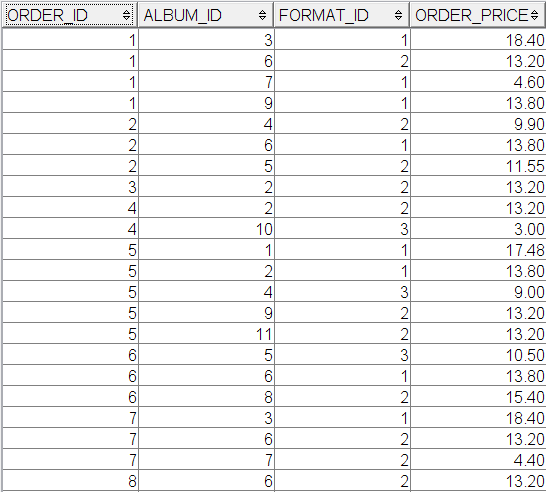
VALUES

-- order\_id = 18

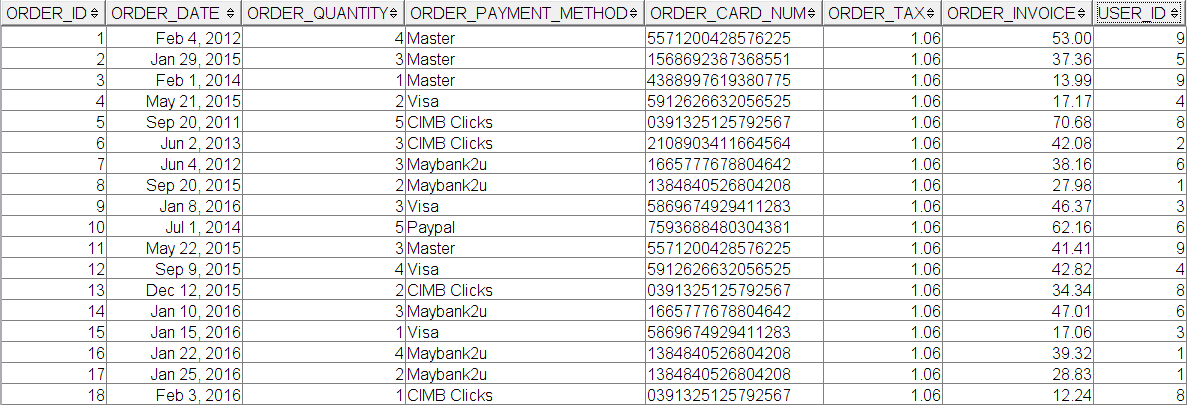
(18, 5, 2);

SELECT \* FROM order\_detail; -- after trigger #4





SELECT \* FROM order;



## (f) Stored procedure

(i) Update tax as a discount for customers that purchased more than or equal to three albums. Note: Used in conjunction with trigger #6 update\_order\_invoice2

CREATE PROCEDURE discount(IN discount\_index DECIMAL(3,2))

LANGUAGE SQL

BEGIN

UPDATE order

SET order\_tax = order\_tax - discount\_index

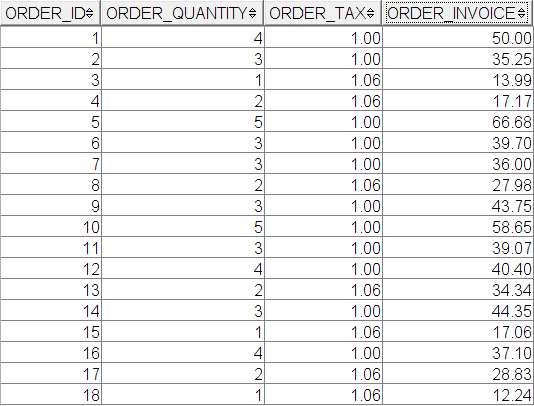
WHERE order\_quantity >= 3;

END@

CALL discount(0.06);

SELECT order\_id, order\_quantity, order\_tax, order\_invoice

FROM order;



(ii) Update an album's category.

CREATE PROCEDURE update\_category(IN category VARCHAR(10), albumID INT)

BEGIN

IF(category = 'New') THEN

UPDATE album

SET album\_category = 'New'

WHERE album\_id = albumID;

ELSE

UPDATE album

SET album\_category = 'Trending'

WHERE album\_id = albumID;

END IF;

END@

CALL update\_category('Trending', 4);

SELECT \* FROM album;



## (g) Additional query component

(i) List the most recent five orders from John.

SELECT order\_date, album\_name AS album\_purchased, CAST(ROUND((order\_price \* order\_tax),2) AS DECIMAL(5,2)) AS final\_price

FROM order, album, order\_detail

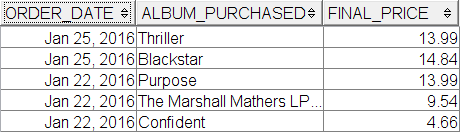
WHERE order\_detail.album\_id = album.album\_id

AND order\_detail.order\_id = order.order\_id

AND user\_id = (SELECT user\_id from user WHERE user\_name = 'John' AND user.user\_id = order.user\_id)

ORDER BY order\_date DESC

FETCH FIRST 5 ROWS ONLY;



(ii) List songs rated and ratings given by Brian.

SELECT song\_name AS rated\_song, review\_rating AS rating\_given, user\_name AS rated\_by

FROM song, review, user U

WHERE U.user\_id = review.user\_id

AND review.song\_id = song.song\_id

AND U.user\_id = (SELECT user\_id FROM user WHERE user\_name = 'Brian');

