Nidal Chalhoub

ZK2WOP  2016. 01. 20.

Database Home Project

Table of Contents

[Introduction 2](#_Toc440415495)

[Table structures 3](#_Toc440415496)

[Visualisation with connections 4](#_Toc440415497)

[ER Diagram 5](#_Toc440415498)

[Decomposition and BCNF 6](#_Toc440415499)

[SQL Queries 0](#_Toc440415500)

[Table Creation and Insert 0](#_Toc440415501)

[A – Simple, single table selects 3](#_Toc440415502)

[B – Simple, single-table GROUP BY 5](#_Toc440415503)

[C – Complex, multi-table selects 7](#_Toc440415504)

[D – Complex sub-query selects 10](#_Toc440415505)

[E - analytical functions (cube, rollup, group by (field1, field2, ..., fieldn), group by grouping sets, row\_number, rank, dense rank) 16](#_Toc440415506)

[F - DDL operations. 2 inserts, 2 updates, 2 deletes 20](#_Toc440415507)

# Introduction

This database represents a set of MCs (rappers), their Labels, their Stage names and the Bands.

The database has four tables: MC, Label, Band, Stagename. Out of the four the Stagename table is the connector table. In the following the structure of these tables, the losslessness of the database, the decomposition of the tables, the ER diagram and a couple of queries will be shown. In the query displaying process the Oracle SQL Developer will be used for convenient purposes.

# Table structures

Labels

* Label\_ID: integer (Primary Key)
* Label\_name: varchar
* Gross: integer
* Founded: date
* SoundCloud\_account: varchar
* HQ\_location: varchar (the label’s registered HQ location)

Bands

* Band\_ID: integer (Primary Key)
* Band\_name: varchar
* Side: varchar (can be East, West, Mid-West)
* Label\_ID: integer (Foreign Key)
* Num\_of\_members: integer
* Years\_active: integer
* Num\_of\_the\_collabs: integer

MC

* MC\_ID: integer (Primary Key)
* Status: varchar (can be Alive, Dead)
* Era: varchar (can be Old school, Golden, New era)
* Birthplace: varchar
* Birthdate: date
* Real\_name: varchar

Stagename (connector table)

* Stagename\_ID: integer (Primary Key)
* Artistname: varchar
* MC\_ID: integer (Foreign Key)
* Label\_ID: integer (Foreign Key)

## Visualisation with connections

|  |
| --- |
| **Label** |
| PK: Label\_ID |
| Label\_name |
| Gross |
| Founded |
| SoundCloud\_account |
| HQ\_location |

|  |
| --- |
| **Band** |
| PK: Band\_ID |
| Band\_name |
| Side |
| Label\_ID (Foreign Key) |
| Num\_of\_members |
| Years\_active |
| Num\_of\_the\_collabs |

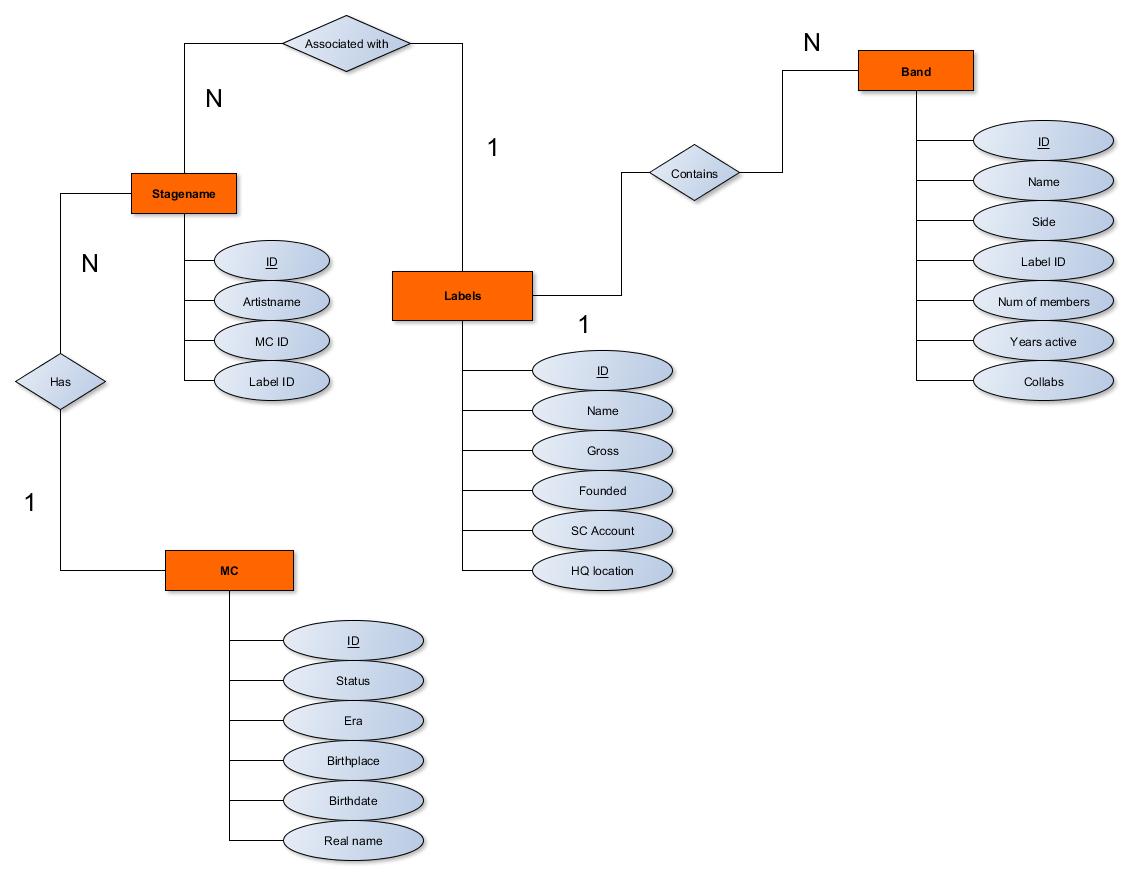
|  |
| --- |
| **Stagename** |
| PK: stagename\_ID |
| Artistname |
| MC\_ID (Foreign Key) |
| Label\_ID (Foreign Key) |

|  |
| --- |
| **MC** |
| PK: MC\_ID |
| Status |
| Era |
| Birthplace |
| Birthdate |
| Real\_name |

## ER Diagram

Relations in the database

* One MC can have multiple Stage names (1 to N)
* Multiple Stage names can be associated with one Label (N to 1), because one MC can belong to one Label (1 to 1), but one MC can have multiple Stage names
* One Label can contain multiple Bands (1 to N)



## Decomposition and BCNF

In the following the correct decomposition and the proof of the losslessness will be shown with the help of BCNF.

Relation R is defined as the following

R = {LabelID, Label\_Name, Gross, Founded, SoundCloud\_Account, HQ\_Location, BandID, Bandname, Side, Num\_Of\_Members, Years\_Active, Collabs, MCID, Status, Era, Birthplace, Birthdate, Realname, StagenameID, Artistname}

Functional dependencies

|  |  |  |  |
| --- | --- | --- | --- |
| * F1Label{LabelID} -> {Label\_Name, Gross, Founded, SoundCloud\_Account, HQ\_Location} | | | |
| * F2Band{BandID} -> {Bandname, Side, Labelid, Num\_Of\_Membes, Years\_Active, Collabs} | | | |
| * F3MC{McID} -> {Status, Era, Birthplace, Birthdate, Realname} |  |  |  |
| * F4Stagename{StagenameID} -> {Artistname, McID, LabelID} |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | labelid | labelname | gross | founded | sc | hq | bandid | bandname | side | #mems | years | collabs | mcid | status | era | pob | dob | realname | stagenameid | artistname |
| Label | B(1,1) | B(1,2) | B(1,3) | B(1,4) | B(1,5) | B(1,6) | B(1,7) | B(1,8) | B(1,9) | B(1,10) | B(1,11) | B(1,12) | B(1,13) | B(1,14) | B(1,15) | B(1,16) | B(1,17) | B(1,18) | B(1,19) | B(1,20) |
| Band | B(2,1) | B(2,2) | B(2,3) | B(2,4) | B(2,5) | B(2,6) | B(2,7) | B(2,8) | B(2,9) | B(2,10) | B(2,11) | B(2,12) | B(2,13) | B(2,14) | B(2,15) | B(2,16) | B(2,17) | B(2,18) | B(2,19) | B(2,20) |
| MC | B(3,1) | B(3,2) | B(3,3) | B(3,4) | B(3,5) | B(3,6) | B(3,7) | B(3,8) | B(3,9) | B(3,10) | B(3,11) | B(3,12) | B(3,13) | B(3,14) | B(3,15) | B(3,16) | B(3,17) | B(3,18) | B(3,19) | B(3,20) |
| Stagename | B(4,1) | B(4,2) | B(4,3) | B(4,4) | B(4,5) | B(4,6) | B(4,7) | B(4,8) | B(4,9) | B(4,10) | B(4,11) | B(4,12) | B(4,13) | B(4,14) | B(4,15) | B(4,16) | B(4,17) | B(4,18) | B(4,19) | B(4,20) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | labelid | labelname | gross | founded | sc | hq | bandid | bandname | side | #mems | years | collabs | mcid | status | era | pob | dob | realname | stagenameid | artistname |
| Label | A1 | A2 | A3 | A4 | A5 | A6 | B(1,7) | B(1,8) | B(1,9) | B(1,10) | B(1,11) | B(1,12) | B(1,13) | B(1,14) | B(1,15) | B(1,16) | B(1,17) | B(1,18) | B(1,19) | B(1,20) |
| Band | B(2,1) | B(2,2) | B(2,3) | B(2,4) | B(2,5) | B(2,6) | B(2,7) | B(2,8) | B(2,9) | B(2,10) | B(2,11) | B(2,12) | B(2,13) | B(2,14) | B(2,15) | B(2,16) | B(2,17) | B(2,18) | B(2,19) | B(2,20) |
| MC | B(3,1) | B(3,2) | B(3,3) | B(3,4) | B(3,5) | B(3,6) | B(3,7) | B(3,8) | B(3,9) | B(3,10) | B(3,11) | B(3,12) | B(3,13) | B(3,14) | B(3,15) | B(3,16) | B(3,17) | B(3,18) | B(3,19) | B(3,20) |
| Stagename | B(4,1) | B(4,2) | B(4,3) | B(4,4) | B(4,5) | B(4,6) | B(4,7) | B(4,8) | B(4,9) | B(4,10) | B(4,11) | B(4,12) | B(4,13) | B(4,14) | B(4,15) | B(4,16) | B(4,17) | B(4,18) | B(4,19) | B(4,20) |

**F1Label{LabelID} -> {Label\_Name, Gross, Founded, SoundCloud\_Account, HQ\_Location}**

**F2Band{BandID} -> {Bandname, Side, Labelid, Num\_Of\_Membes, Years\_Active, Collabs}**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | labelid | labelname | gross | founded | sc | hq | bandid | bandname | side | #mems | years | collabs | mcid | status | era | pob | dob | realname | stagenameid | artistname |
| Label | A1 | A2 | A3 | A4 | A5 | A6 | B(1,7) | B(1,8) | B(1,9) | B(1,10) | B(1,11) | B(1,12) | B(1,13) | B(1,14) | B(1,15) | B(1,16) | B(1,17) | B(1,18) | B(1,19) | B(1,20) |
| Band | A1 | B(2,2) | B(2,3) | B(2,4) | B(2,5) | B(2,6) | A7 | A8 | A9 | A10 | A11 | A12 | B(2,13) | B(2,14) | B(2,15) | B(2,16) | B(2,17) | B(2,18) | B(2,19) | B(2,20) |
| MC | B(3,1) | B(3,2) | B(3,3) | B(3,4) | B(3,5) | B(3,6) | B(3,7) | B(3,8) | B(3,9) | B(3,10) | B(3,11) | B(3,12) | B(3,13) | B(3,14) | B(3,15) | B(3,16) | B(3,17) | B(3,18) | B(3,19) | B(3,20) |
| Stagename | B(4,1) | B(4,2) | B(4,3) | B(4,4) | B(4,5) | B(4,6) | B(4,7) | B(4,8) | B(4,9) | B(4,10) | B(4,11) | B(4,12) | B(4,13) | B(4,14) | B(4,15) | B(4,16) | B(4,17) | B(4,18) | B(4,19) | B(4,20) |

**F3MC{McID} -> {Status, Era, Birthplace, Birthdate, Realname}**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | labelid | labelname | gross | founded | sc | hq | bandid | bandname | side | #mems | years | collabs | mcid | status | era | pob | dob | realname | stagenameid | artistname |
| Label | A1 | A2 | A3 | A4 | A5 | A6 | B(1,7) | B(1,8) | B(1,9) | B(1,10) | B(1,11) | B(1,12) | B(1,13) | B(1,14) | B(1,15) | B(1,16) | B(1,17) | B(1,18) | B(1,19) | B(1,20) |
| Band | A1 | B(2,2) | B(2,3) | B(2,4) | B(2,5) | B(2,6) | A7 | A8 | A9 | A10 | A11 | A12 | B(2,13) | B(2,14) | B(2,15) | B(2,16) | B(2,17) | B(2,18) | B(2,19) | B(2,20) |
| MC | B(3,1) | B(3,2) | B(3,3) | B(3,4) | B(3,5) | B(3,6) | B(3,7) | B(3,8) | B(3,9) | B(3,10) | B(3,11) | B(3,12) | A13 | A14 | A15 | A16 | A17 | A18 | B(3,19) | B(3,20) |
| Stagename | B(4,1) | B(4,2) | B(4,3) | B(4,4) | B(4,5) | B(4,6) | B(4,7) | B(4,8) | B(4,9) | B(4,10) | B(4,11) | B(4,12) | B(4,13) | B(4,14) | B(4,15) | B(4,16) | B(4,17) | B(4,18) | B(4,19) | B(4,20) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | labelid | labelname | gross | founded | sc | hq | bandid | bandname | side | #mems | years | collabs | mcid | status | era | pob | dob | realname | stagenameid | artistname |
| Label | A1 | A2 | A3 | A4 | A5 | A6 | B(1,7) | B(1,8) | B(1,9) | B(1,10) | B(1,11) | B(1,12) | B(1,13) | B(1,14) | B(1,15) | B(1,16) | B(1,17) | B(1,18) | B(1,19) | B(1,20) |
| Band | A1 | B(2,2) | B(2,3) | B(2,4) | B(2,5) | B(2,6) | A7 | A8 | A9 | A10 | A11 | A12 | B(2,13) | B(2,14) | B(2,15) | B(2,16) | B(2,17) | B(2,18) | B(2,19) | B(2,20) |
| MC | B(3,1) | B(3,2) | B(3,3) | B(3,4) | B(3,5) | B(3,6) | B(3,7) | B(3,8) | B(3,9) | B(3,10) | B(3,11) | B(3,12) | A13 | A14 | A15 | A16 | A17 | A18 | B(3,19) | B(3,20) |
| Stagename | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 | A17 | A18 | A19 | A20 |

**F4Stagename{StagenameID} -> {Artistname, McID, LabelID}**

# SQL Queries

## Table Creation and Insert

--creating the tables

DROP TABLE stagename;

DROP TABLE mc;

DROP TABLE band;

DROP TABLE label;

CREATE TABLE label

(

labelid int PRIMARY KEY,

labelname VARCHAR(50),

gross int,

founded DATE NOT NULL,

scaccount VARCHAR(50),

hqlocation VARCHAR(20)

);

CREATE TABLE band

(

bandid int PRIMARY KEY,

bandname VARCHAR(25),

side VARCHAR(10),

labelid int CONSTRAINT band\_label\_id REFERENCES label(labelid),

numofmembers int NOT NULL,

yearsactive int,

collabs int

);

CREATE TABLE mc

(

mcid int PRIMARY KEY,

status VARCHAR(10),

era VARCHAR(20),

birthplace VARCHAR(50),

birthdate DATE,

realname VARCHAR(50)

);

CREATE TABLE stagename

(

stagenameid int PRIMARY KEY,

artistname VARCHAR(20) UNIQUE,

mcid int CONSTRAINT stagename\_mc\_id REFERENCES mc(mcid),

labelid int CONSTRAINT stagename\_label\_id REFERENCES label(labelid)

);

--filling up the tables

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(1,'Stones Throw Records',560000,TO\_DATE('1996-05-12','YYYY-MM-DD'),'soundcloud.com/stonesthrowrecords','Detroit');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(2,'Project Blowed',300000,TO\_DATE('1998-10-05','YYYY-MM-DD'),'soundcloud.com/projectblowed','New York');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(3,'Atlantic Records',450000,TO\_DATE('2003-07-08','YYYY-MM-DD'),'soundcloud.com/atlanticrecords','New York');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(4,'The Breaks',580000,TO\_DATE('1986-12-05','YYYY-MM-DD'),'soundcloud.com/thebreaks','Brooklyn');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(5,'Shady Records',550000,TO\_DATE('1993-02-09','YYYY-MM-DD'),'soundcloud.com/wildpitchrecords','Brooklyn');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(6,'Ruthless Recods',530000,TO\_DATE('1995-11-15','YYYY-MM-DD'),'soundcloud.com/ruthless','Cleveland');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(7,'Rhymesayers Entertainment',550000,TO\_DATE('1993-12-15','YYYY-MM-DD'),'soundcloud.com/rhymesayers','New York');

INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(8,'Hieroglyphics',530000,TO\_DATE('1996-05-19','YYYY-MM-DD'),'soundcloud.com/hierorecords','California');

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(1, 'Bone Thugs N Harmony', 'Mid-West', 6, 4, 15, 80);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(2, 'A Team', 'East', 2, 2, 10, 60);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(3, 'Asap Mob', 'West', 3, 5, 5, 30);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(4, 'Sugarhill Gang', 'East', 4, 5, 30, 70);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(5, 'D12', 'East', 5, 7, 10, 75);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(6, 'Madvillain', 'Mid-West', 1, 2, 8, 75);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(7, 'The Freestyle Fellowship', 'East', 2, 3, 5, 25);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(8, 'DangerMouse', 'Mid-West', 1, 2, 5, 20);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(9, 'NWA', 'Mid-West', 6, 6, 25, 90);

INSERT INTO band(bandid, bandname, side, labelid, numofmembers, yearsactive, collabs) VALUES(10, 'Slum Village', 'Mid-West', 1, 3, 15, 80);

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(1,'Alive','Golden','Detroit',TO\_DATE('1975-05-03','YYYY-MM-DD'),'Marshall Mathers');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(2,'Alive','Oldschool','Brooklyn',TO\_DATE('1963-10-23','YYYY-MM-DD'),'Kurt Walker');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(3,'Dead','Golden','Brooklyn',TO\_DATE('1968-07-23','YYYY-MM-DD'),'Keith Edward Elam');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(4,'Dead','Oldschool','California',TO\_DATE('1972-10-11','YYYY-MM-DD'),'Lesane Perish Crooks');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(5,'Dead','Oldschool','New York',TO\_DATE('1975-12-21','YYYY-MM-DD'),'Christopher Wallace');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(6,'Alive','Oldschool','California',TO\_DATE('1980-02-20','YYYY-MM-DD'),'Calvin Broadus Jr');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(7,'Alive','Golden','Detroit',TO\_DATE('1978-04-06','YYYY-MM-DD'),'Otis Jackson Jr');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(8,'Alive','Golden','London',TO\_DATE('1977-11-12','YYYY-MM-DD'),'Daniel Dumile');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(9,'Alive','Golden','Cleveland',TO\_DATE('1978-11-09','YYYY-MM-DD'),'Bryon Anthony McCane II');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(10,'Alive','Golden','Cleveland',TO\_DATE('1975-03-08','YYYY-MM-DD'),'Anthony Henderson');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(11,'Dead','Oldschool','California',TO\_DATE('1964-09-07','YYYY-MM-DD'),'Eric Lynn Wright');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(12,'Alive','Golden','California',TO\_DATE('1974-07-28','YYYY-MM-DD'),'Edwin M Hayes Jr');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(13,'Alive','Newera','California',TO\_DATE('1984-12-25','YYYY-MM-DD'),'Arian Asllani');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(14,'Alive','Golden','New York',TO\_DATE('1976-06-05','YYYY-MM-DD'),'Ian Matthias Bavitz');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(15,'Alive','Golden','Minneapolis',TO\_DATE('1972-09-07','YYYY-MM-DD'),'Sean Michael Daley');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(16,'Alive','Golden','Oakland',TO\_DATE('1974-07-17','YYYY-MM-DD'),'Paulo Peacock');

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(17,'Dead','Golden','Detroit',TO\_DATE('1974-02-07','YYYY-MM-DD'),'James Dewitt Yancey');

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(1, 'Eminem', 1, 5);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(2, 'Slim Shady', 1, 5);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(3, 'Kurtis Blow', 2, 4);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(4, 'Guru', 3, 4);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(5, '2Pac', 4, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(6, 'Tupac', 4, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(7, 'Notorious BIG', 5, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(8, 'Biggie Smalls', 5, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(9, 'Snoop Dogg', 6, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(10, 'Madlib', 7, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(11, 'Beat Conducta', 7, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(12, 'Quasimoto', 7, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(13, 'MF Doom', 8, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(14, 'King Gedorah', 8, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(15, 'Venomous Villain', 8, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(16, 'Bizzy Bone', 9, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(17, 'Krayzie Bone', 10, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(18, 'Eazy-E', 11, 6);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(19, 'Aceyalone', 12, 2);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(20, 'Action Bronson', 13, 3);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(21, 'Aesop Rock', 14, 7);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(22, 'Slug', 15, 7);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(23, 'Pep Love', 16, 8);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(24, 'J Dilla', 17, 1);

INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(25, 'Jay Dee', 17, 1);

COMMIT;

## A – Simple, single table selects

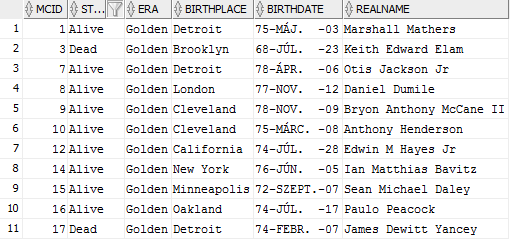
A1. Select all golden era MCs.

SELECT \*

FROM mc

WHERE UPPER(era)='GOLDEN';

Output:



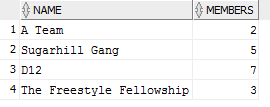
A2. Select all the East-sider band's name, and the number of its members.

SELECT bandname AS name, numofmembers AS members

FROM band

WHERE UPPER(side)='EAST';

Output:



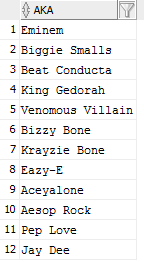
A3. select all artist names whose contains letter "E".

SELECT artistname as aka

FROM stagename

WHERE UPPER(artistname) LIKE '%E%';

Output:



## B – Simple, single-table GROUP BY

B1: list the number of bands grouped by the sides, order by the number of bands, descendingly.

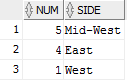
SELECT COUNT(\*) AS NUM, side AS SIDE

FROM band

GROUP BY side

ORDER BY NUM desc;

Output:



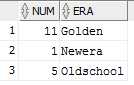
B2: list the number of MCs, grouped by the era.

SELECT COUNT(\*) AS NUM, era AS ERA

FROM mc

GROUP BY ERA;

Output:



B3: list the number MCs per birthplace

SELECT COUNT(\*) AS num, birthplace AS bplace

FROM mc

GROUP BY birthplace;

Output:



## C – Complex, multi-table selects

C1: select ALL artist names with their corresponding real names.

SELECT stagename.artistname as aka, mc.realname as real\_name

FROM stagename, mc

WHERE stagename.mcid=mc.mcid(+);

Output:





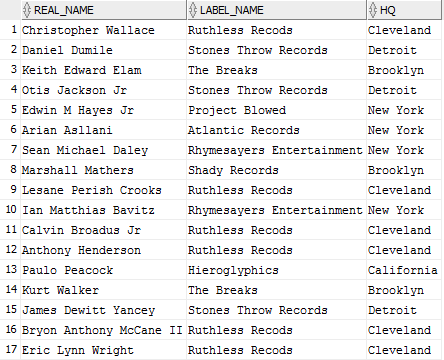
C2: select all real names with their label's name and the corresponding label's location.

SELECT DISTINCT mc.realname AS real\_name, label.labelname AS label\_name, label.hqlocation AS HQ

FROM mc, label, stagename

WHERE mc.mcid=stagename.mcid(+) AND stagename.labelid=label.labelid;

Output:



C3: select all MC real names, their label. List only the ones, whose were found in the 80's.

SELECT DISTINCT mc.realname AS real\_name, label.labelname AS label\_name

FROM mc, label, stagename

WHERE mc.mcid=stagename.mcid(+) AND stagename.labelid=label.labelid AND TO\_CHAR(label.founded,'YYYY')<='1989';

Output:



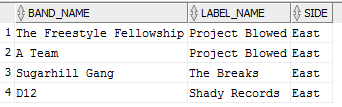
C4: list all bands' names with their label names. List only the East-sider bands.

SELECT band.bandname AS band\_name, label.labelname AS label\_name, band.side AS side

FROM band, label

WHERE band.labelid=label.labelid(+) AND UPPER(side)='EAST';

Output:



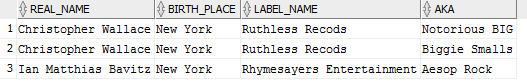
C5: list all the MCs artist names, real name, birth place whose birth place is New York. List the label and all their artistnames too.

SELECT mc.realname AS real\_name, mc.birthplace AS birth\_place, label.labelname AS label\_name, stagename.artistname AS AKA

FROM mc, label, stagename

WHERE stagename.mcid=mc.mcid(+) AND stagename.labelid=label.labelid AND UPPER(mc.birthplace)='NEW YORK';

Output:



## D – Complex sub-query selects

D1: list the real names, how many artist names belonging to them and their labels. order the list by the numer of artist names, descending.

SELECT label.labelname AS labelname, data3.numofnames AS numberofaliases, data3.realnames as realnames

FROM

(SELECT distinct data2.artist\_name AS numofnames, data2.real\_name AS realnames, stagename.labelid AS label\_id

FROM

(SELECT mc.mcid AS mc\_id, data.\*

FROM

(SELECT COUNT(stagename.artistname) AS artist\_name, mc.realname AS real\_name

FROM mc, stagename

WHERE mc.mcid=stagename.mcid(+)

GROUP BY mc.realname

ORDER BY COUNT(stagename.artistname) DESC)data, mc

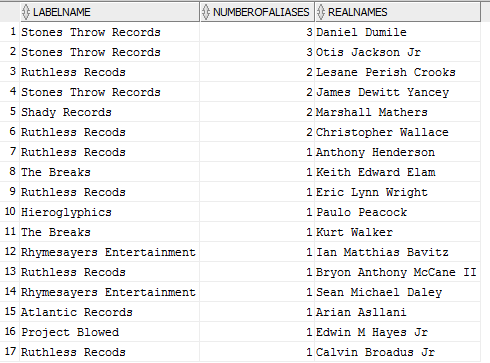
WHERE mc.realname=data.real\_name)data2, stagename

WHERE data2.mc\_id=stagename.mcid)data3, label

WHERE data3.label\_id=label.labelid

ORDER BY numberofaliases DESC;

Output:



D2: list the number of the collabs per labels, order the list by the collaborations, ascending.

SELECT AVG(data.collabs) AS average, data.labelname

FROM

(SELECT band.collabs AS collabs, label.labelname AS labelname

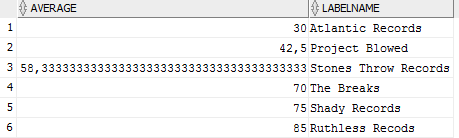
FROM band, label

WHERE band.labelid=label.labelid(+))data

GROUP BY data.labelname

ORDER BY average ASC;

Output:



D3: select those MCs, whose are alive, and their label's HQ location is the same as the MC's birthplace.

SELECT DISTINCT data.\*, label.labelid, label.hqlocation

FROM

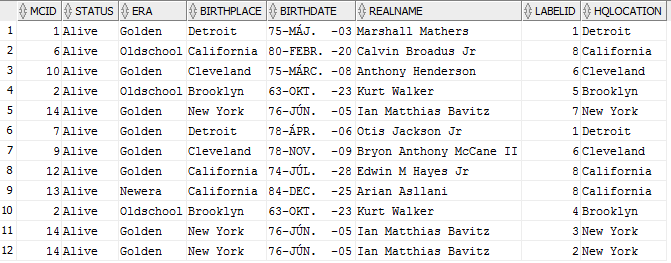
(SELECT \*

FROM mc

WHERE UPPER(status)='ALIVE')data, label, stagename

WHERE data.mcid=stagename.mcid AND UPPER(data.birthplace)=UPPER(label.hqlocation);

Output:



D4: display the label, which has the maximum gross, list out all its MCs with their real names.

SELECT data2.\*, mc.realname AS real\_name

FROM

(SELECT label.\*

FROM

(SELECT MIN(gross) AS mingross

FROM label)data1, label

WHERE gross=data1.mingross)data2, mc, stagename

WHERE stagename.labelid=data2.labelid AND mc.mcid=stagename.mcid;

Output:



D5: display the label, which has less gross than the average gross.

SELECT label.\*

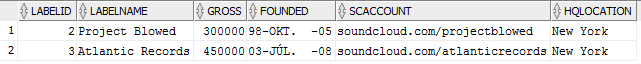
FROM

(SELECT AVG(gross) as avggross

FROM label)data1, label

WHERE label.gross<data1.avggross;

Output:



D6: Which label has the most bands?

SELECT data5.\*

FROM

(SELECT MAX(data3.numofbands) AS maxnum

FROM

(SELECT COUNT(data2.rank) AS numofbands, data2.label\_name

FROM

(SELECT ROW\_NUMBER() OVER(PARTITION BY data1.label\_id ORDER BY data1.label\_id) AS rank, data1.label\_name

FROM

(SELECT band.\*, label.labelid AS label\_id, label.labelname AS label\_name

FROM band, label

WHERE band.labelid=label.labelid(+))data1)data2

GROUP BY data2.label\_name)data3)data4,

(SELECT COUNT(data2.rank) AS numofbands, data2.label\_name

FROM

(SELECT ROW\_NUMBER() OVER(PARTITION BY data1.label\_id ORDER BY data1.label\_id) AS rank, data1.label\_name

FROM

(SELECT band.\*, label.labelid AS label\_id, label.labelname AS label\_name

FROM band, label

WHERE band.labelid=label.labelid(+))data1)data2

GROUP BY data2.label\_name)data5

WHERE data4.maxnum=data5.numofbands;

Output:



D7: Which MC(s) has / have the most aliases?

SELECT mc.\*

FROM

(SELECT basedata.\*

FROM

(SELECT COUNT(stagename.stagenameid) AS aliases, mc.mcid

FROM mc, stagename

WHERE stagename.mcid=mc.mcid(+)

GROUP BY mc.mcid)basedata,

(SELECT MAX(data.aliases) AS num

FROM

(SELECT COUNT(stagename.stagenameid) AS aliases, mc.mcid

FROM mc, stagename

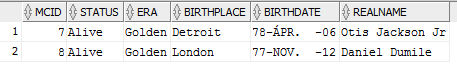
WHERE stagename.mcid=mc.mcid(+)

GROUP BY mc.mcid)data)names

WHERE basedata.aliases=names.num)mcdata, mc

WHERE mcdata.mcid=mc.mcid(+);

Output:



## E - analytical functions (cube, rollup, group by (field1, field2, ..., fieldn), group by grouping sets, row\_number, rank, dense rank)

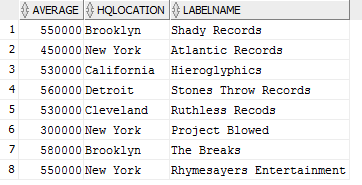
E1: list out the label's average gross, grouped by hq location and label name.

SELECT AVG(gross) AS average, hqlocation, labelname

FROM label

GROUP BY hqlocation, labelname;

Output:



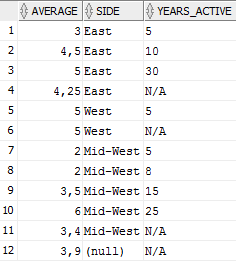
E2: list the bands average number of their members grouped by hierarchically by the side and the years active.

SELECT AVG(numofmembers) AS average, side, NVL(TO\_CHAR(yearsactive), 'N/A') AS years\_active

FROM band

GROUP BY ROLLUP(side, yearsactive);

Output:



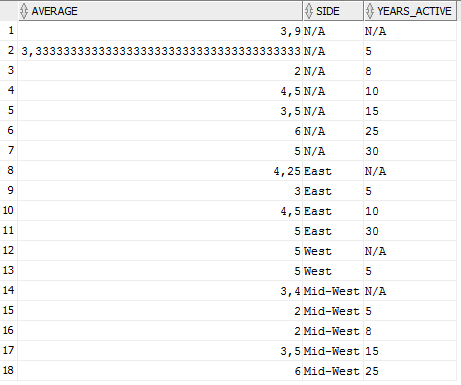
E3: list the bands average number of their members grouped by every possible combination by the side and the years active.

SELECT AVG(numofmembers) AS average, NVL(side, 'N/A') AS side, NVL(TO\_CHAR(yearsactive), 'N/A') as years\_active

FROM band

GROUP BY CUBE(side, yearsactive);

Output:



E4: list out all the MCs who have at least 2 artist names.

CREATE OR REPLACE VIEW data1 AS

SELECT mc.\*, stagename.artistname AS artist\_name

FROM mc, stagename

WHERE mc.mcid=stagename.mcid(+)

CREATE OR REPLACE VIEW aliases AS

SELECT data1.\*, ROW\_NUMBER() OVER(PARTITION BY mcid ORDER BY artist\_name) ranks

FROM data1;

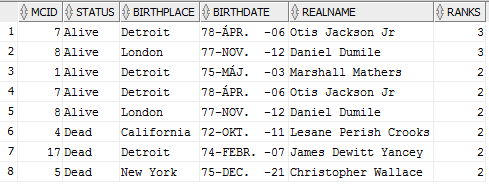
SELECT DISTINCT aliases.mcid, aliases.status, aliases.birthplace, aliases.birthdate, aliases.realname, aliases.ranks

FROM aliases

WHERE ranks>=2

ORDER BY ranks DESC;

Output:



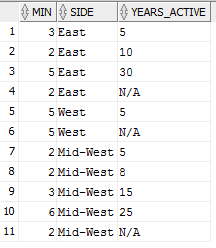
E5: list the bands minimal number of members, grouped by the side and the years active and then by the side only.

SELECT MIN (numofmembers) AS min, side, NVL(TO\_CHAR(yearsactive), 'N/A') AS years\_active

FROM band

GROUP BY GROUPING SETS ((side,yearsactive),(side));

Output:



## F - DDL operations. 2 inserts, 2 updates, 2 deletes

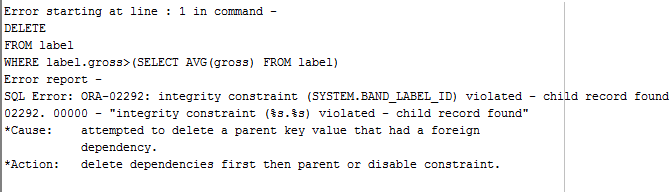
F1: delete those labels in the database, whose gross is higher than the average gross.

DELETE

FROM label

WHERE label.gross>(SELECT AVG(gross) FROM label);

Output (Expected error):



F2: delete all east-side bands.

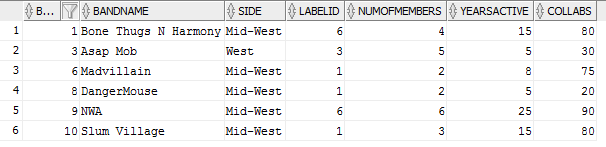
DELETE

FROM band

WHERE UPPER(side)='EAST';

SELECT \* FROM band;

Output:



F3: delete all bands whose have a less number of years active than their label's age.

SELECT \* FROM band;

DELETE

FROM band

WHERE bandid in (SELECT DISTINCT bandid

FROM

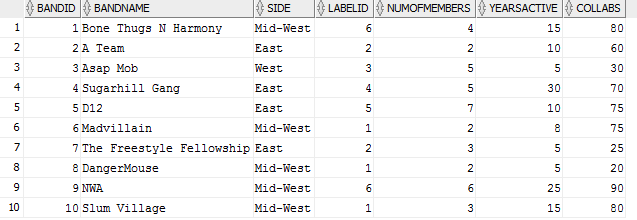
(SELECT TO\_CHAR(SYSDATE,'YYYY')-TO\_CHAR(founded,'YYYY') AS age, labelid

FROM label)data1, band

WHERE band.yearsactive<data1.age);

SELECT \* FROM band;

Output:



After deletion:



F4: update the bands: set collabs to 50 to those bands, whose number of members is less than 3.

SELECT \* FROM band;

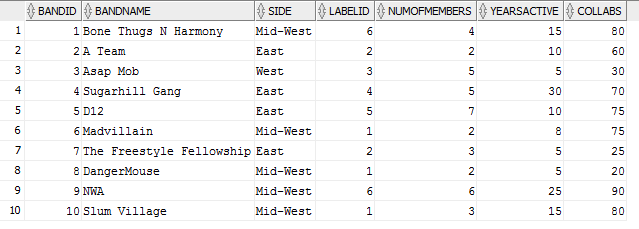
UPDATE band

SET collabs=50

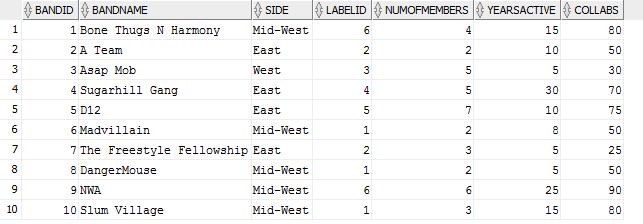
WHERE bandid IN (SELECT bandid FROM band WHERE numofmembers<3);

SELECT \* FROM band;

Output:



After update:



F5: update those MC's era to Newera, whose one of their artist names begins with the letter 'A'.

SELECT \* FROM mc;

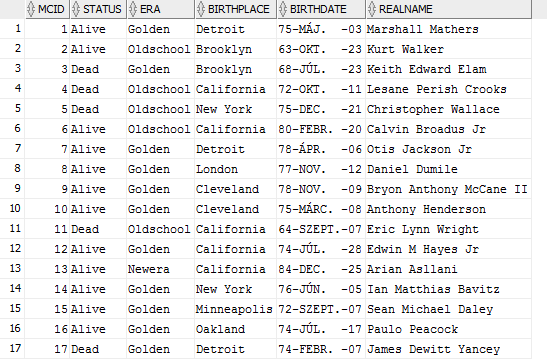
UPDATE mc

SET era='Newera'

WHERE mcid IN (SELECT mc.mcid FROM mc, stagename WHERE mc.mcid=stagename.mcid(+) AND UPPER(stagename.artistname) LIKE 'A%');

SELECT \* FROM mc;

Output:



After update:



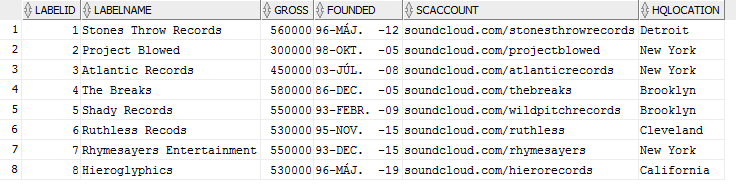
F6: insert a new label.

SELECT \* FROM label;

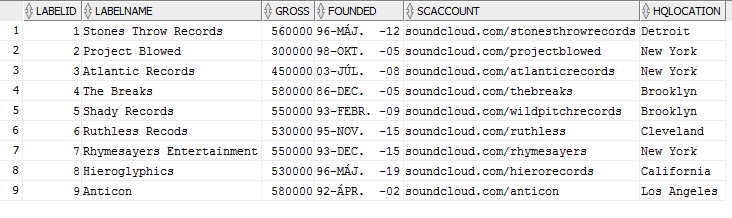
INSERT INTO label(labelid, labelname, gross, founded, scaccount, hqlocation) VALUES(9,'Anticon',580000,TO\_DATE('1992-04-02','YYYY-MM-DD'),'soundcloud.com/anticon','Los Angeles');

SELECT \* FROM label;

Output:



After inserting Anticon:



F7: insert a new MC with artist name too.

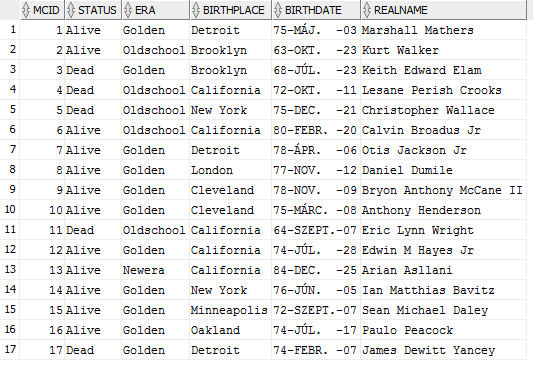
SELECT \* FROM mc;

INSERT INTO mc(mcid, status, era, birthplace, birthdate, realname) VALUES(18,'Alive','Newera','Los Angeles',TO\_DATE('1991-03-06','YYYY-MM-DD'),'Tyler Gregory Okonma');

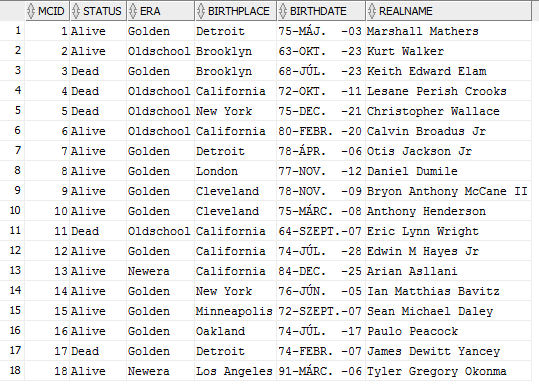
INSERT INTO stagename(stagenameid, artistname, mcid, labelid) VALUES(26, 'Tyler The Creator', 18, 3);

SELECT \* FROM mc;

Output:



After inserting Tyler The Creator:



Tyler Gregory Okonma’s Artistname:

SELECT stagename.\*

FROM stagename, mc

WHERE stagename.mcid=mc.mcid and UPPER(mc.realname)='TYLER GREGORY OKONMA';

