THAPAR UNIVERSITY

UCS310: DATABASE MANAGEMENT SYSTEM

SUBMITTED TO:

DR. SANTOSH SINGH RATHORE

CSED, TU

SUBMITTED BY:

PRAKHAR GUPTA (101610066)

PRATEEK CHHIKARA (101603247)

GROUP-G6


THE MUSIC
LIBRARY

INDEX

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and teachers. We would like to extend our sincere thanks to all of them.

We are highly indebted to Dr. Santosh Singh Rathore for his guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

INTRODUCTION:

The music database is designed to store details of a music collection, including the albums in the collection, the artists who made them, the tracks on the albums.

The Music Database

The music database stores details of a personal music library, and could be used to manage your MP3, CD, or vinyl collection. Because this database is for a personal collection, it's relatively simple and stores only the relationships between artists, albums, and tracks. It ignores the requirements of many music genres, making it most useful for storing popular music and less useful for storing jazz or classical music.

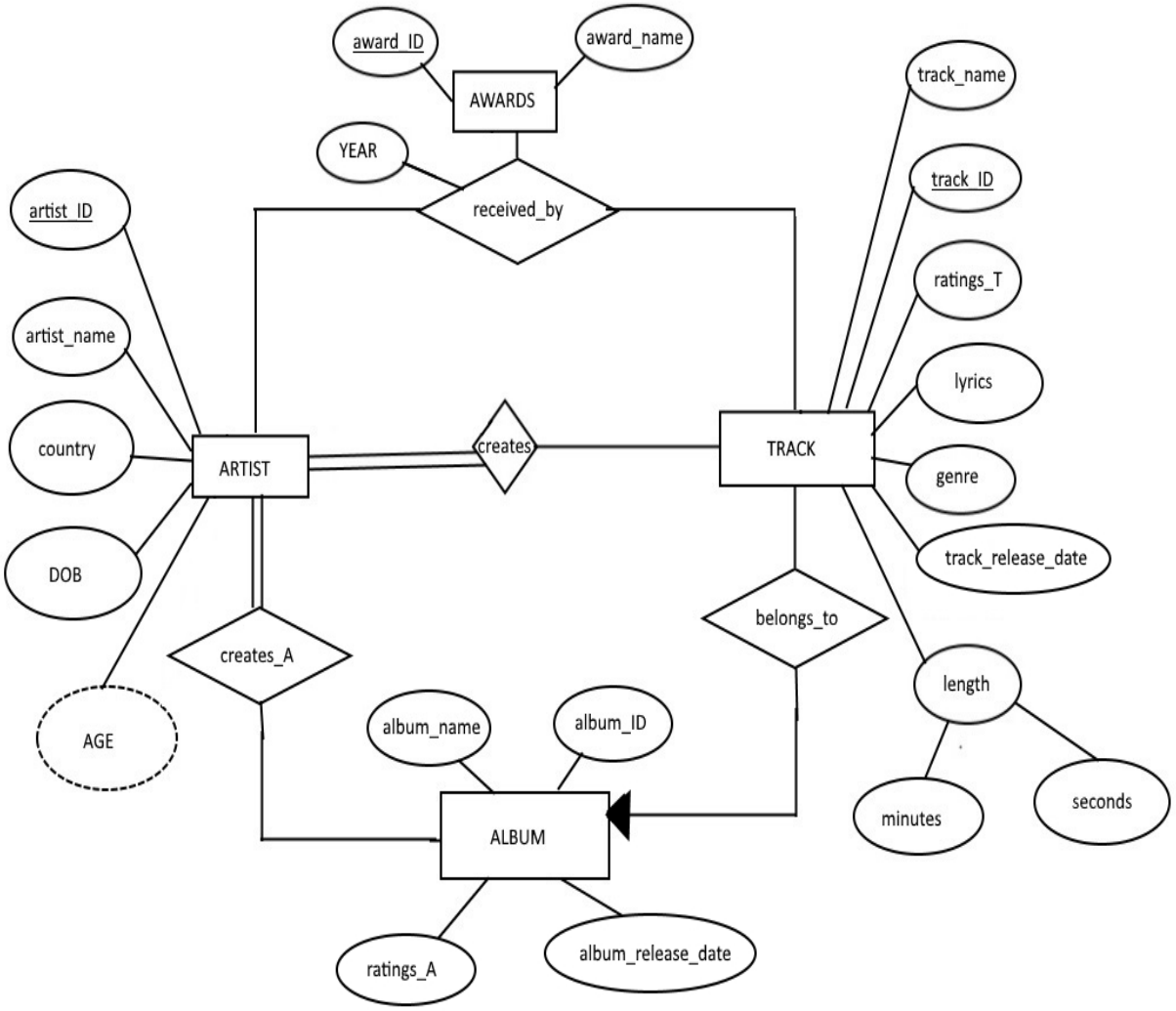
We first draw up a clear list of requirements for our database:

- The collection consists of albums.
- An album is made by exactly one artist.
- An artist makes one or more albums.
- An album contains one or more tracks
- Artists, albums, and tracks each have a name.
- Each track is on exactly one album.
- Each track has a time length, measured in minutes and seconds.

There's no requirement to capture composers, group members or sidemen, recording date or location, the source media, or any other details of artists, albums, or tracks.

The ER diagram derived from our requirements is shown in below figure. The attributes are straightforward: artists, albums, and tracks have names, as well as identifiers to uniquely identify each entity.

ER-Diagram


award_ID
award_name
AWARDS
YEAR
received_by
artist_ID
artist_name
country
DOB
AGE
ARTIST
creates
creates_A
TRACK
track_name
track_ID
ratings_T
lyrics
genre
track_release_date
belongs_to
length
minutes
seconds
album_name
album_ID
ALBUM
ratings_A
album_release_date

What it doesn’t do….?

We’ve kept the music database simple because adding extra features doesn’t help you learn anything new, it just makes the explanations longer. If you wanted to use the music database in practice, then you might consider adding the following features:

1. Support for compilations or various-artists albums, where each track may be by a different artist and may then have its own associated album-like details such as a recording date and time. Under this model, the album would be a strong entity, with many-to-many relationships between artists and albums.
2. Playlists, a user-controlled collection of tracks. For example, you might create a playlist of your favorite tracks from an artist.
3. Source details, such as when you bought an album, what media it came on, how much you paid, and so on.
4. Album details, such as when and where it was recorded, the producer and label, the band members or sidemen who played on the album, and even its artwork.
5. Smarter track management, such as modeling that allows the same track to appear on many albums.

TABLES USED:

ARTIST:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	PRIMARY KEY
ARTIST_NAME	VARCHAR2(30)	
COUNTRY	VARCHAR2(10)	
DOB	DATE	

TRACK:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
TRACK_NAME	VARCHAR2(10)	
TRACK_ID	NUMBER(3)	PRIMARY KEY
RATINGS_T	NUMBER(1)	CHECK
LYRICS	CLOB	
GENRE	VARCHAR(30)	
TRACK_RELEASE_DATE	DATE	
ALBUM_ID	NUMBER(3)	FOREIGN KEY
MINUTES	NUMBER(2)	CHECK
SECONDS	NUMBER(2)	CHECK

ALBUM:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ALBUM_NAME	VARCHAR2(30)	
ALBUM_ID	NUMBER(3)	PRIMARY KEY
RATINGS_A	NUMBER(1)	CHECK
ALBUM_RELEASE_DATE	DATE	

AWARD:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
AWARD_ID	NUMBER(3)	PRIMARY KEY
AWARD_NAME	VARCHAR2(30)	

RECEIVED_BY:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
AWARD_ID	NUMBER(3)	FOREIGN KEY
TRACK_ID	NUMBER(3)	FOREIGN KEY
YEAR	NUMBER(4)	

CREATES_A:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
ALBUM_ID	NUMBER(3)	FOREIGN KEY

CREATES_T:

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ARTIST_ID	NUMBER(3)	FOREIGN KEY
TRACK_ID	NUMBER(3)	FOREIGN KEY

Functional dependency:

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

NORMALIZATION:

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

ARTIST:

ARTIST_ID	ARTIST_NAME	COUNTRY	DOB
1	EMINEM	US	17-10-72
2	TAYLOR SWIFT	US	13-12-89
3	RIHANNA	BARBADOS	20-02-88
4	BRUNO MARS	US	08-10-85
5	MICHAEL JACKSON	US	29-08-58

Functional Dependency:

ARTIST_ID → ARTIST_NAME, COUNTRY, DOB

Candidate Keys: ARTIST_ID
Non-prime attributes: ARTIST_NAME, COUNTRY, DOB

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

TRACK:

TRACK_NAME	TRACK_ID	RATING_T	LYRICS	GENRE	TRACK_RELEASE_DATE	ALBUM_ID	MINUTES	SECONDS
WALK ON WATER	1	5	I walk on waterBut I ain't no JesusI walk on waterBut only when it free...	HIP HOP	15-12-17	1	4	12
NOT AFRAID	2	4	I'm not afraid (I'm not afraid)To take a stand (to take a stand)Everybo...	HIP HOP	14-04-11	2	5	16
SO BAD	3	4	Yeah.Ha ha, you feel that baby?Yeah, I feel it too.Damn.You know, I'm s...	HIP HOP	14-04-11	2	4	45
MEAN	15	4	You, with your words like knivesAnd swords and weapons that you use aga...	COUNTRY MUSIC	25-10-10	9	3	30
BEAUTIFUL	4	5	Lately I've been hard to reach, I've been too long on my ownEverybody h...	HIP HOP	05-03-09	3	6	20
CRACK A BOTTLE	5	4	Ooww ladies and gentlemenThe moment you've all been waiting for..In thi...	HIP HOP	05-03-09	3	4	29
BEAT IT	6	4	They told him, "Don't you ever come around here.Don't wanna see your fa...	POP	25-03-83	4	4	49
THE GIRL IS MINE	7	4	Every night she walks right in my dreamsSince I met her from the startI...	POP	25-03-83	4	4	5
HEAL THE WORLD	8	5	There's A Place InYour HeartAnd I Know That It Is LoveAnd This Place Co...	POP	27-06-93	5	4	15
FULL ATTACK	9	3	Ecstasy... in the airI don't carecan't tell me nothingI'm impairedthe w...	CONTEMPORARY	19-08-12	6	3	10
SHREDDERS	10	4	Work, work, work, work, work, workHe said me have toWork, work, work, w...	CONTEMPORARY	19-08-12	6	3	50
THE LAZY SONG	11	5	Today I don't feel like doing anythingI just wanna lay in my bedDon't f...	FUNK	15-01-11	7	4	32
IT WILL RAIN	12	4	If you ever leave me, baby,Leave some morphine at my door'Cause it woul...	FUNK	15-01-11	7	4	10
TREASURE	13	4	Baby squirrel, you's a sexy motherfuckerGive me your, give me your, giv...	FUNK	22-03-12	8	4	16
MINE	14	4	You were in college working part time waiting tablesLeft a small town, ...	COUNTRY MUSIC	25-10-10	9	4	30
FIFTEEN	16	4	You take a deep breath and you walk through the doorsIt's the morning o...	COUNTRY MUSIC	11-11-08	10	5	20

Functional Dependency:

TRACK_ID → TRACK_NAME, RATING_T, TRACK_RELEASE_DATE, ALBUM_ID, LYRICS, GENRE, MINUTES, SECONDS

Candidate Keys: TRACK_ID
Non-prime attributes: TRACK_NAME, RATING_T, TRACK_RELEASE_DATE, ALBUM_ID, LYRICS, GENRE, MINUTES, SECONDS

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

ALBUM:

ALBUM_ID	ALBUM_NAME	RATING_A	ALBUM_RELEASE_DATE
1	REVIVAL	4.6	15-12-17
2	RECOVERY	4.8	14-04-11
3	RELAPSE	4.3	05-03-09
4	THRILLER	4.5	25-03-83
5	DANGEROUS	4.1	27-06-92
6	BATTLESHIP	3.9	19-08-12
7	DOO-WOPS	2.8	15-01-11
8	UNORTHODOX	3.4	22-03-12
9	SPEAK NOW	4.2	25-10-10
10	FEARLESS	4.4	11-11-08

Functional Dependency:

ALBUM_ID → ALBUM_NAME, RATING_A, ALBUM_RELEASE_DATE

Candidate Keys: ALBUM_ID
Non-prime attributes: ALBUM_NAME, RATING_A, ALBUM_RELEASE_DATE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

AWARD:

AWARD_ID	AWARD_NAME
1	MTV
2	GRAMMY
3	BILLBOARD MUSIC
4	AMERICAN MUSIC
5	FAVOURITE POP/ROCK
6	ARTIST OF THE YEAR
7	BEST ART DIRECTION

Functional Dependency:

AWARD_ID → AWARD_NAME

Candidate Keys: AWARD_ID

Non-prime attributes: AWARD_NAME

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

RECEIVED BY:

ARTIST_ID	AWARD_ID	TRACK_ID	YEAR
1	1	2	2001
1	2	1	2008
1	3	4	2004
2	1	15	2015
2	2	14	2009
2	3	16	2017
3	1	9	2005
3	2	10	2010
3	7	10	2005
4	5	11	2011
4	6	13	2006
5	2	7	2013
5	5	6	2015
5	6	7	2009

Functional Dependency:

AWARD_ID, ARTIST_ID, TRACK_ID → YEAR

Candidate Keys: AWARD_ID, ARTIST_ID, TRACK_ID

Non-prime attributes: YEAR

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

CREATES_A:

ARTIST_ID	ALBUM_ID
1	1
1	2
1	3
2	9
2	10
3	6
4	7
4	8
5	4
5	5

Functional Dependency:

ARTIST_ID, ALBUM_ID → NONE

Candidate Keys: ARTIST_ID, ALBUM_ID

Non-prime attributes: NONE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

CREATES_T:

ARTIST_ID	TRACK_ID
1	1
1	2
1	3
1	4
1	5
2	14
2	15
2	16
3	9
3	10
4	11
4	12
4	13
5	6
5	7
5	8

Functional Dependency:

ARTIST_ID, TRACK_ID → NONE

Candidate Keys: ARTIST_ID, TRACK_ID

Non-prime attributes: NONE

1NF: This table is in first normal form because all the attributes in the relation have atomic domains.

2NF: This table is in second normal form because there is no partial dependency present.

3NF: This table is in third normal form because there is no transitive dependency.

INSERTION:

```
DECLARE
PROCEDURE INSERT_ARTIST (A_ID NUMBER,
 A_NAME VARCHAR2
, A_COUNTRY VARCHAR2
, A_DOB DATE) AS
BEGIN
INSERT INTO ARTIST VALUES(A_ID, A_NAME,A_COUNTRY,A_DOB);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS FOR
ARTIST');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
BEGIN
INSERT_ARTIST (: A_ID,:A_NAME,:A_COUNTRY,:A_DOB);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF ARTIST ADDED ' ||
SQL%ROWCOUNT);
END;
/
```

```
DECLARE
PROCEDURE INSERT_ALBUM (A_ID NUMBER,
 A_NAME VARCHAR2
, A_RATING NUMBER
, A_RELEASE DATE) AS
BEGIN
INSERT INTO ALBUM VALUES(A_ID, A_NAME,A_RATING,A_RELEASE);
```

```
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS FOR ALBUM');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE ('AN ERROR HAS OCCURED WHILE INSERTING!!');
END;
BEGIN
INSERT_ALBUM (: A_ID,:A_NAME,:A_RATING,:A_RELEASE);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF ALBUM ADDED ' || SQL%ROWCOUNT);
END;
/
```

```
DECLARE
PROCEDURE INSERT_TRACK (
  T_NAME VARCHAR2
, T_ID NUMBER
, T_RATING NUMBER
, T_LYRICS CLOB
, T_GENRE VARCHAR2
, T_RELEASE DATE
, A_ID NUMBER
, T_MIN NUMBER
, T_SEC NUMBER) AS
BEGIN
INSERT INTO TRACK VALUES(T_NAME,T_ID,T_RATING,T_LYRICS,T_GENRE,T_RELEASE,A_ID,T_MIN,T_SEC);
```

```
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE('YOU ARE INSERTING DUPLICATE DETAILS FOR
TRACK');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
BEGIN
INSERT_TRACK (:
TRACK_NAME,:TRACK_ID,:TRACK_RATING,:TRACK_LYRICS,:TRACK_GENRE,:
TRACK_RELEASE,:ALBUM_ID,:MINUTES,:SECONDS);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF TRACK ADDED ' ||
SQL%ROWCOUNT);
END;
/


DECLARE
PROCEDURE INSERT_AWARD (
  A_ID NUMBER
, A_NAME VARCHAR2
) AS
BEGIN
INSERT INTO AWARD VALUES (A_ID, A_NAME);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS FOR
AWARD');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE ('AN ERROR HAS OCCURED WHILE
INSERTING!!');
END;
BEGIN
```

```
INSERT_AWARD (: AWARD_ID,:AWARD_NAME);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF AWARDS ADDED ' ||
SQL%ROWCOUNT);
END;

/


DECLARE
PROCEDURE INSERT_RECEIVED_BY (
  A_ID NUMBER
, AW_ID NUMBER
, T_ID NUMBER
, YEAR NUMBER) AS
BEGIN
INSERT INTO RECEIVED_BY VALUES (A_ID, AW_ID,T_ID,YEAR);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
BEGIN
INSERT_RECEIVED_BY (: ARTIST_ID, AWARD_ID,:TRACK_ID,:YEAR);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF ENTRIES ADDED ' ||
SQL%ROWCOUNT);
END;


/
```

```
DECLARE
PROCEDURE INSERT_CREATES_A (
  A_ID NUMBER
, AL_ID NUMBER
) AS
BEGIN
INSERT INTO CREATES_A VALUES (A_ID, AL_ID);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
   DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('SQLERRM');
END;
BEGIN
INSERT_CREATES_A (: ARTIST_ID,:ALBUM_ID);
DBMS_OUTPUT.PUT_LINE ('NUMBER OF ENTRIES ADDED ' ||
SQL%ROWCOUNT);
END;

/



DECLARE
PROCEDURE INSERT_CREATES_T (
  A_ID NUMBER
, T_ID NUMBER
) AS
BEGIN
INSERT INTO CREATES_T VALUES(A_ID,T_ID);
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
```

```
    DBMS_OUTPUT.PUT_LINE ('YOU ARE INSERTING DUPLICATE DETAILS.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
BEGIN
INSERT_CREATES_T(:ARTIST_ID,:T_ID);
DBMS_OUTPUT.PUT_LINE('NUMBER OF ENTRIES ADDED ' ||
SQL%ROWCOUNT);
END;

/
```

DELETION:

```
DECLARE
ERROR_ON_DELETE EXCEPTION;


   PROCEDURE DELETE_DATA (
   A NUMBER
   )
   AS
   BEGIN
   DELETE FROM ARTIST WHERE ARTIST_ID=A;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);


   END;
BEGIN
   DELETE_DATA(:ARTIST_ID);
END;
/
```

```
DECLARE
ERROR_ON_DELETE EXCEPTION;


   PROCEDURE DELETE_DATA(
   A NUMBER
   )
   AS
   BEGIN
   DELETE FROM ALBUM WHERE ALBUM_ID=A;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);


   END;
BEGIN
   DELETE_DATA(:ALBUM_ID);
END;
/
DECLARE
ERROR_ON_DELETE EXCEPTION;
```

```
   PROCEDURE DELETE_DATA(
   A NUMBER
   )
   AS
   BEGIN
   DELETE FROM TRACK WHERE TRACK_ID=A;
   DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
   DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
   DBMS_OUTPUT.PUT_LINE(SQLERRM);

   END;
BEGIN
   DELETE_DATA(:TRACK_ID);
END;
/
DECLARE
ERROR_ON_DELETE EXCEPTION;

   PROCEDURE DELETE_DATA(
     A NUMBER
   , B NUMBER
```

```
    , C NUMBER
    )
    AS
    BEGIN
    DELETE FROM RECEIVED_BY WHERE ARTIST_ID=A AND AWARD_ID=B
AND TRACK_ID=C;
    DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
    DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
    DBMS_OUTPUT.PUT_LINE(SQLERRM);

    END;
BEGIN
    DELETE_DATA(:ARTIST_ID,:AWARD_ID,:TRACK_ID);
END;
/
DECLARE
ERROR_ON_DELETE EXCEPTION;

    PROCEDURE DELETE_DATA(
      A NUMBER
    , B NUMBER
```

```
    )
    AS
    BEGIN
    DELETE FROM CREATES_A WHERE ARTIST_ID=A AND ALBUM_ID=B;
    DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
    DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
    DBMS_OUTPUT.PUT_LINE(SQLERRM);

    END;
BEGIN
    DELETE_DATA(:ARTIST_ID,:ALBUM_ID);
END;
/

DECLARE
ERROR_ON_DELETE EXCEPTION;

    PROCEDURE DELETE_DATA(
      A NUMBER
    , B NUMBER
    )
```

```
    AS
    BEGIN
    DELETE FROM CREATES_T WHERE ARTIST_ID=A AND TRACK_ID=B;
    DBMS_OUTPUT.PUT_LINE('NO OF ENTRIES DELETED:
'||SQL%ROWCOUNT);
IF
SQL%ROWCOUNT=0 THEN
RAISE ERROR_ON_DELETE;
END IF;
EXCEPTION
WHEN ERROR_ON_DELETE THEN
    DBMS_OUTPUT.PUT_LINE('THIS RECORD IS NOT AVAILABLE IN
DATABASE');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('THAT''S AN ERROR');
    DBMS_OUTPUT.PUT_LINE(SQLERRM);


    END;
BEGIN
    DELETE_DATA(:ARTIST_ID,:TRACK_ID);
END;
/
```

CURSOR

```
DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER: =0;
PROCEDURE retrive(id number) as
cursor C1 is select * from TRACK where TRACK_ID = id;
REC C1%rowtype;

begin
open C1;
loop
fetch C1 into REC;
exit when C1%notfound;
FOUND:=1;
dbms_output.put_line('TRACK NAME: ' || REC.TRACK_NAME ||'   RATING:
'||REC.RATING_T ||'   DURATION: '||REC.MINUTES||':'||REC.SECONDS);
dbms_output.put_line('RELEASE DATE: ' || REC.TRACK_RELEASE_DATE||'
GENRE: '||REC.GENRE);
dbms_output.put_line('LYRICS: ');
dbms_output.put_line(REC.LYRICS);
end loop;
close C1;
IF FOUND <> 1 THEN
RAISE NOT_FOUND;
END IF;
EXCEPTION
WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS TRACK ');
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);
```

```
end;

begin
retrive(:TRACK_ID);
END;
/
DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER: =0;

   PROCEDURE retrive(id number) as
   cursor C1 is select * from CREATES_A where  ARTIST_ID = id;
   REC C1%rowtype;
   ALB ALBUM%ROWTYPE;
   A_ID ARTIST.ARTIST_ID%TYPE;
   begin
   SELECT DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;
   dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS CREATED
FOLLOWING ALBUMS' );
   open C1;

   loop
   fetch C1 into REC;
   exit when C1%notfound;
FOUND:=1;

   SELECT * INTO ALB FROM ALBUM WHERE ALBUM_ID=REC.ALBUM_ID;
```

```
    dbms_output.put_line('NAME: '||ALB.ALBUM_NAME||'   RATING:
'||ALB.RATING_A||'  RELEASE DATE: '||ALB.ALBUM_RELEASE_DATE);


    end loop;
    close C1;
IF FOUND <> 1 THEN
RAISE NOT_FOUND;
END IF;
EXCEPTION
WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS ARTIST ');
WHEN OTHERS THEN
dbms_output.put_line(SQLERRM);


    end;


begin
    retrive(:ARTIST_ID);
END;
/


DECLARE
NOT_FOUND EXCEPTION;
FOUND NUMBER: =0;


    PROCEDURE retrive(id number) as
    cursor C1 is select * from CREATES_T where ARTIST_ID = id;
    REC C1%rowtype;
    TRK TRACK%ROWTYPE;
    A_ID ARTIST.ARTIST_ID%TYPE;
    begin
```

```
   SELECT DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;

   dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS CREATED
FOLLOWING TRACKS' );

   open C1;


   loop

   fetch C1 into REC;

   exit when C1%notfound;

FOUND: =1;


   SELECT * INTO TRK FROM TRACK WHERE TRACK_ID=REC.TRACK_ID;

   dbms_output.put_line ('NAME: '||TRK.TRACK_NAME||'   RATING:
'||TRK.RATING_T||'  TRACK DATE: '||TRK.TRACK_ID);


   end loop;

   close C1;

IF FOUND <> 1 THEN

RAISE NOT_FOUND;

END IF;

EXCEPTION

WHEN NOT_FOUND THEN dbms_output.put_line('NO RECORDS FOUND FOR
THIS ARTIST ');

WHEN OTHERS THEN

dbms_output.put_line(SQLERRM);


   end;


begin

   retrive(:ARTIST_ID);

END;
```

```
DECLARE
   NOT_FOUND EXCEPTION;
   FOUND NUMBER: =0;
   PROCEDURE retrive(id number) as
   cursor C1 is select * from RECEIVED_BY where ARTIST_ID = id;
   REC C1%rowtype;
   AWD AWARD%ROWTYPE;
   A_ID ARTIST.ARTIST_ID%TYPE;
TRK TRACK.TRACK_NAME%TYPE;
   begin
   SELECT  DISTINCT ARTIST_ID INTO A_ID FROM RECEIVED_BY WHERE
ARTIST_ID=id;
   dbms_output.put_line('ARTIST WHOSE ID IS '|| A_ID ||' HAS WON FOLLOWING
AWARDS' );
   open C1;

   loop
   fetch C1 into REC;
   exit when C1%notfound;
   FOUND :=1;
   SELECT * INTO AWD FROM AWARD WHERE AWARD_ID=REC.AWARD_ID;
SELECT TRACK_NAME INTO TRK FROM TRACK WHERE
TRACK_ID=REC.TRACK_ID;
   dbms_output.put_line('NAME: '||AWD.AWARD_NAME||'   FOR TRACK:
'||TRK||'  YEAR: '||REC.YEAR);

   end loop;
   close C1;
   IF FOUND <> 1 THEN
   RAISE NOT_FOUND;
   END IF;
   EXCEPTION
   WHEN NOT_FOUND THEN
      dbms_output.put_line('NO RECORDS FOUND FOR THIS ARTIST');
   WHEN OTHERS THEN
      dbms_output.put_line(SQLERRM);
   end;

begin
   retrive(:ARTIST_ID);
END;
/
```

AGE CALCULATION

```
DECLARE
   AGE NUMBER;
   DOB DATE;
   FUNCTION AGE_CALC(DOB IN DATE) RETURN NUMBER IS
   begin
   AGE:=(SYSDATE-DOB)/365;
   RETURN(AGE);
   end;

begin
AGE:=AGE_CALC(:DOB);
DBMS_OUTPUT.PUT_LINE('AGE IS ' || ROUND(AGE,0)||' YEARS
'||ROUND(((AGE-ROUND(AGE,0))*365),0)||' DAYS');
END;
```

TRIGGERS

```
CREATE OR REPLACE TRIGGER AGE
AFTER INSERT OR DELETE OR UPDATE ON ARTIST
FOR EACH ROW
BEGIN
   IF INSERTING THEN
     DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
     DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;

/
CREATE OR REPLACE TRIGGER AWARD
AFTER INSERT OR DELETE OR UPDATE ON AWARD
FOR EACH ROW
BEGIN
   IF INSERTING THEN
     DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
     DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER ALBUM
AFTER INSERT OR DELETE OR UPDATE ON ALBUM
FOR EACH ROW
BEGIN
   IF INSERTING THEN
     DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
     DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
/

CREATE OR REPLACE TRIGGER TRACK
AFTER INSERT OR DELETE OR UPDATE ON TRACK
FOR EACH ROW
BEGIN
   IF INSERTING THEN
     DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
     DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
/

CREATE OR REPLACE TRIGGER R_B
```

```
AFTER INSERT OR DELETE OR UPDATE ON RECEIVED_BY
FOR EACH ROW
BEGIN
   IF INSERTING THEN
     DBMS_OUTPUT.PUT_LINE('INSERTING');
   ELSIF DELETING THEN
     DBMS_OUTPUT.PUT_LINE('DELETING');
   ELSIF UPDATING THEN
      DBMS_OUTPUT.PUT_LINE('UPDATING');
   END IF;
END;
/
```

THE END