# Artists Music Database

By: Jason Hodge
Dr. Alan Labouseur
CMPT 308 - Fall 2020

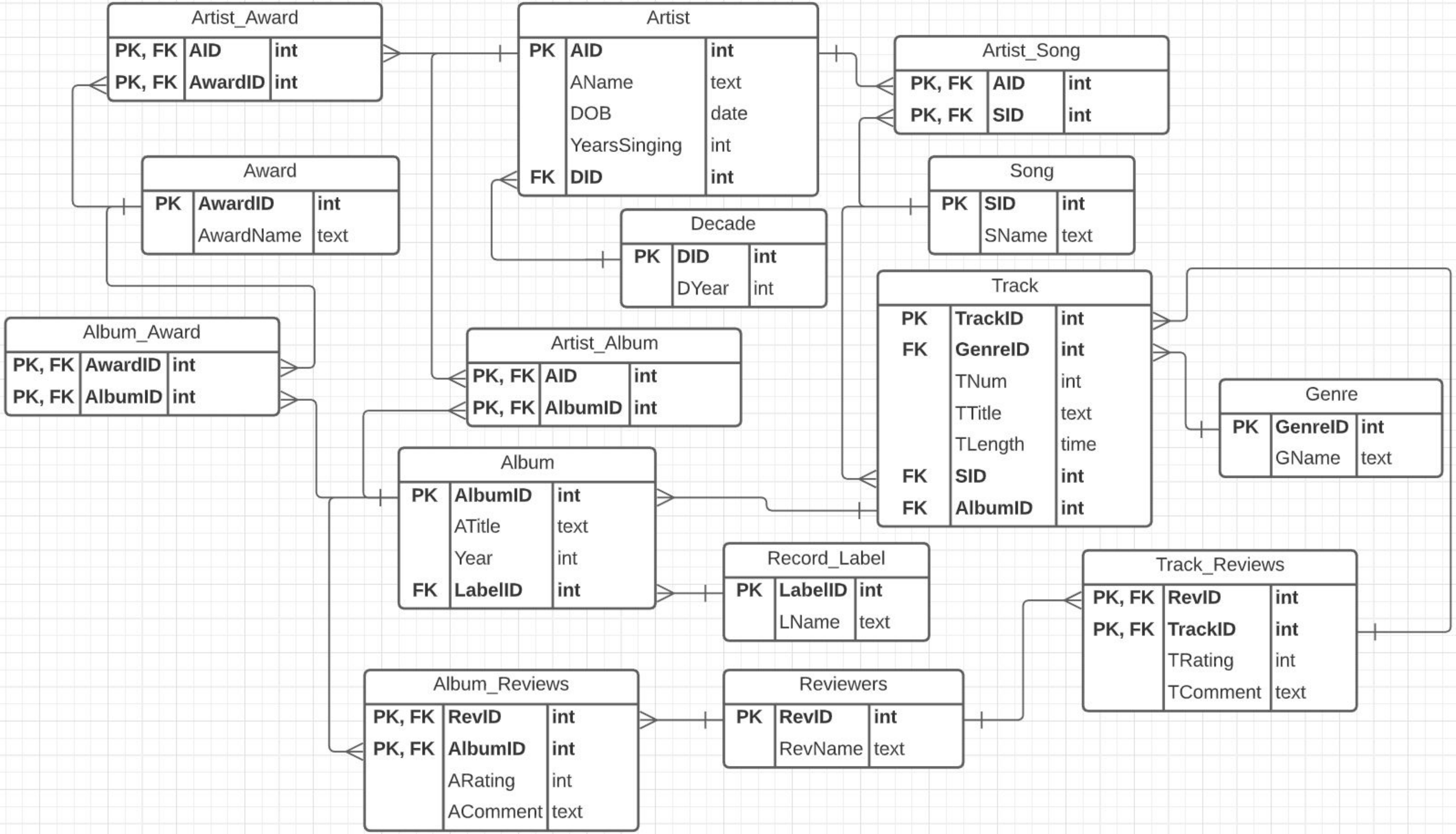
## Table of Contents

## Executive Summary

This database can be used to organize Artists music of all kinds, from all decades and styles of music.

This database shows and keeps track of Artists Music, specifically their songs in albums, reviews made by reviewers on tracks and albums, the decade an Artist is from, the genre of music tracks in an album are, awards an artist and their albums won, as well as the record labels that produced Artists albums. This is done through the use of tables represented in the Entity-Relationship diagram and create table statements. With this information and corresponding data, wanted information can be found through the use of queries, views, triggers, and stored procedures. Examples of these are presented.

Also included is a very unique artist with an excellent hit song!

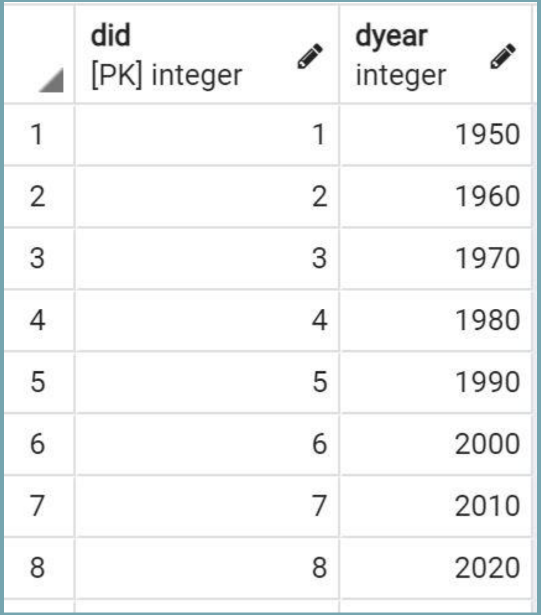

Artist_Award
PK, FK AID int
PK, FK AwardID int
Artist
PK AID int
AName text
DOB date
YearsSinging int
FK DID int
Artist_Song
PK, FK AID int
PK, FK SID int
Award
PK AwardID int
AwardName text
Decade
PK DID int
DYear int
Song
PK SID int
SName text
Track
PK TrackID int
FK GenreID int
TNum int
TTitle text
TLength time
FK SID int
FK AlbumID int
Album_Award
PK, FK AwardID int
PK, FK AlbumID int
Artist_Album
PK, FK AID int
PK, FK AlbumID int
Genre
PK GenreID int
GName text
Album
PK AlbumID int
ATitle text
Year int
FK LabelID int
Record_Label
PK LabelID int
LName text
Track_Reviews
PK, FK RevID int
PK, FK TrackID int
TRating int
TComment text
Album_Reviews
PK, FK RevID int
PK, FK AlbumID int
ARating int
AComment text
Reviewers
PK RevID int
RevName text

## Create Table Statements

This table stores the decade year for which an Artist can belong to.

```
CREATE TABLE Decade (
      DID           Int not null,
      DYear          Int not null,
Primary key (DID)
);
```

Functional Dependencies: DID -> DYear



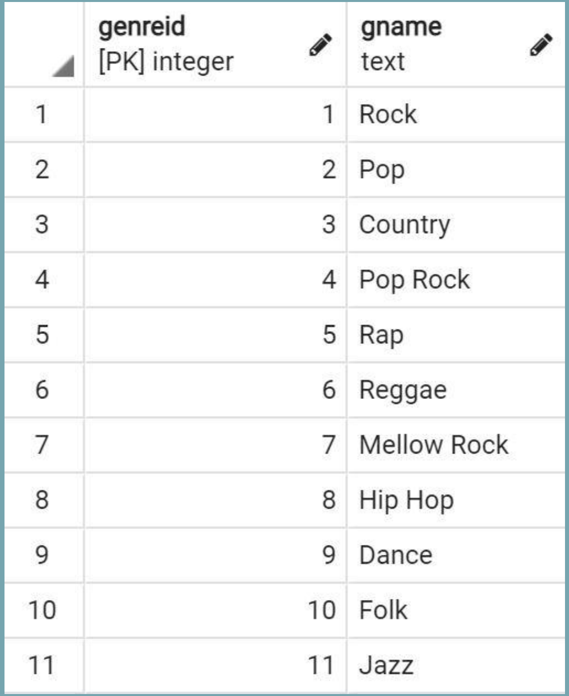
| | did [PK] integer | dyear integer |
|---|---|---|
| 1 | 1 | 1950 |
| 2 | 2 | 1960 |
| 3 | 3 | 1970 |
| 4 | 4 | 1980 |
| 5 | 5 | 1990 |
| 6 | 6 | 2000 |
| 7 | 7 | 2010 |
| 8 | 8 | 2020 |

## Create Table Statements

This table stores the genre names a track, song, and its album can be characterized under.

```
CREATE TABLE Genre (
	GenreID		Int not null,
	GName		Text not null,
Primary key (GenreID)
);
```

Functional Dependencies: GenreID -> GName



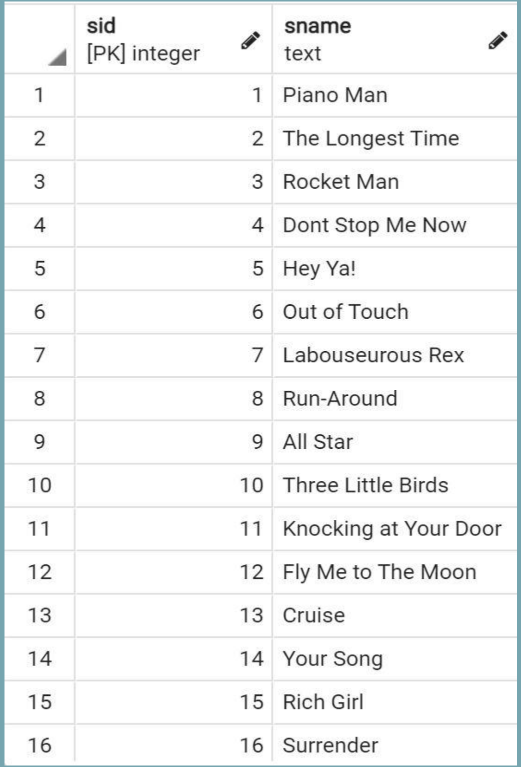
| | genreid [PK] integer | gname text |
|---|---|---|
| 1 | 1 | Rock |
| 2 | 2 | Pop |
| 3 | 3 | Country |
| 4 | 4 | Pop Rock |
| 5 | 5 | Rap |
| 6 | 6 | Reggae |
| 7 | 7 | Mellow Rock |
| 8 | 8 | Hip Hop |
| 9 | 9 | Dance |
| 10 | 10 | Folk |
| 11 | 11 | Jazz |

## Create Table Statements

This table stores the song names by artists, where a song is also associated with a track number in an album.

```
CREATE TABLE Song (
      SID          Int not null,
      SName          Text not null,
Primary key (SID)
);
```

Functional Dependencies: SID -> SName



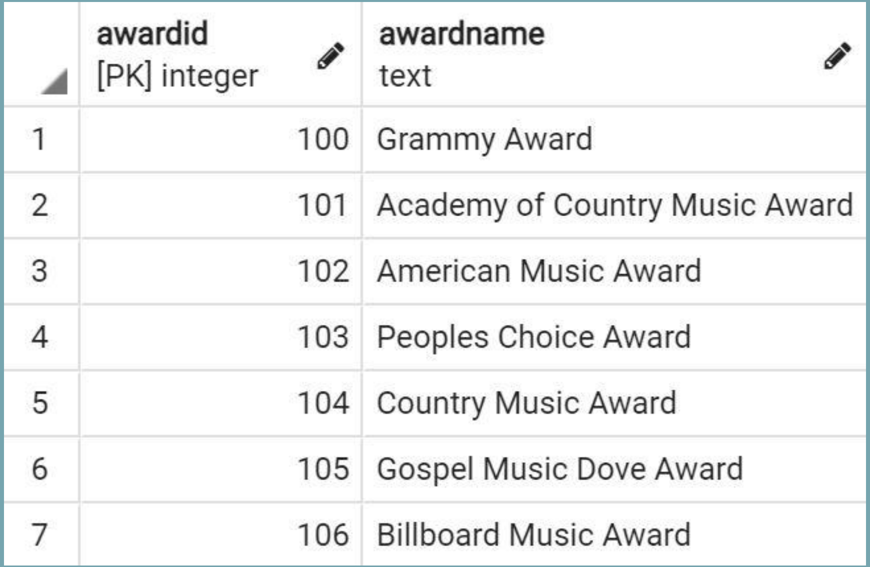
| | sid [PK] integer | sname text |
|---|---|---|
| 1 | 1 | Piano Man |
| 2 | 2 | The Longest Time |
| 3 | 3 | Rocket Man |
| 4 | 4 | Dont Stop Me Now |
| 5 | 5 | Hey Ya! |
| 6 | 6 | Out of Touch |
| 7 | 7 | Labouseurous Rex |
| 8 | 8 | Run-Around |
| 9 | 9 | All Star |
| 10 | 10 | Three Little Birds |
| 11 | 11 | Knocking at Your Door |
| 12 | 12 | Fly Me to The Moon |
| 13 | 13 | Cruise |
| 14 | 14 | Your Song |
| 15 | 15 | Rich Girl |
| 16 | 16 | Surrender |

## Create Table Statements

This table stores the award names for which an Artist and or an album can receive.

```
CREATE TABLE Award (
	AwardID		Int not null,
	AwardName	Text not null,
Primary key (AwardID)
);
```

Functional Dependencies: AwardID -> AwardName



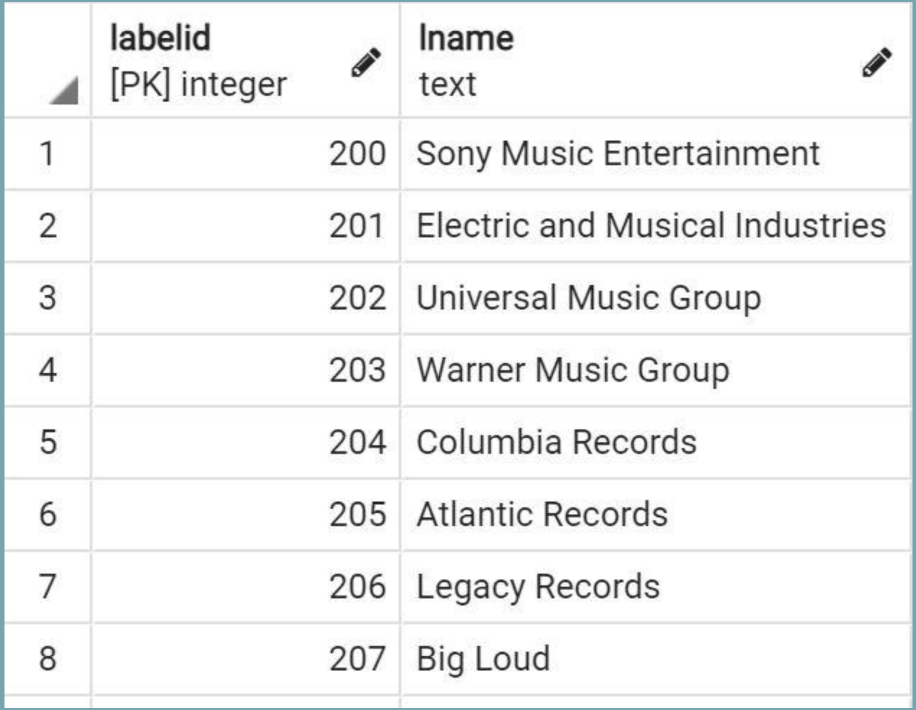
| | awardid [PK] integer | awardname text |
|---|---|---|
| 1 | 100 | Grammy Award |
| 2 | 101 | Academy of Country Music Award |
| 3 | 102 | American Music Award |
| 4 | 103 | Peoples Choice Award |
| 5 | 104 | Country Music Award |
| 6 | 105 | Gospel Music Dove Award |
| 7 | 106 | Billboard Music Award |

## Create Table Statements

This table stores the label name for which an album can be produced by.

```
CREATE TABLE Record_Label (
        LabelID         Int not null,
        LName           Text not null,
Primary key (LabelID)
);
```

Functional Dependencies: LabelID -> LName



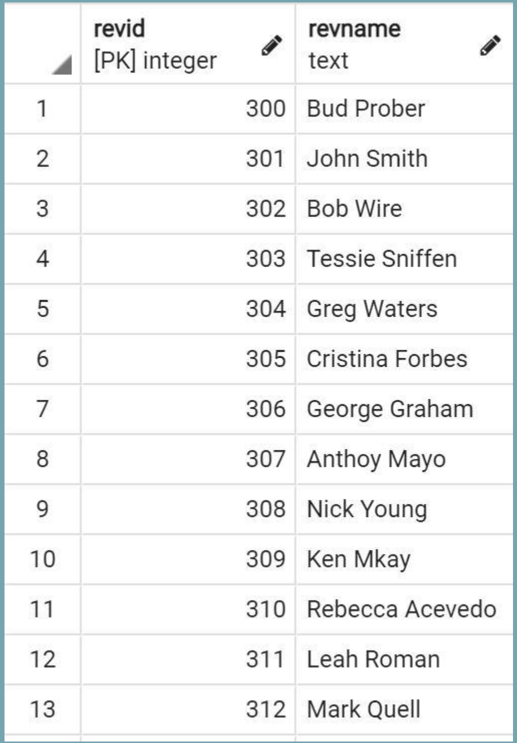
| | labelid [PK] integer | lname text |
|---|---|---|
| 1 | 200 | Sony Music Entertainment |
| 2 | 201 | Electric and Musical Industries |
| 3 | 202 | Universal Music Group |
| 4 | 203 | Warner Music Group |
| 5 | 204 | Columbia Records |
| 6 | 205 | Atlantic Records |
| 7 | 206 | Legacy Records |
| 8 | 207 | Big Loud |

## Create Table Statements

This table stores the names of reviewers, where reviewers can give reviews on both albums and individual tracks.

```
CREATE TABLE Reviewers (
	RevID		Int not null,
	RevName		Text not null,
Primary key (RevID)
);
```

Functional Dependencies: RevID -> RevName



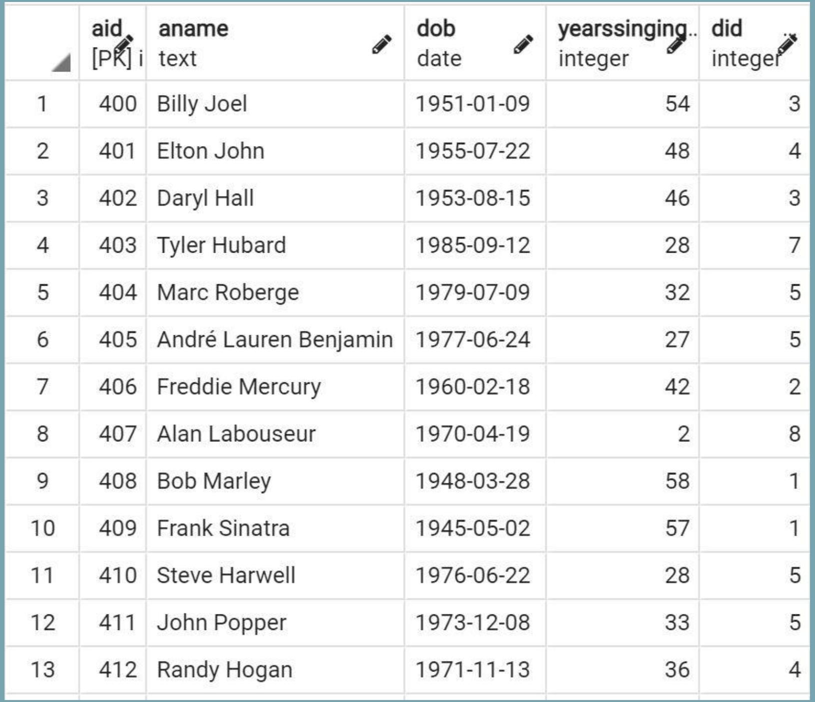
| | revid [PK] integer | revname text |
|---|---|---|
| 1 | 300 | Bud Prober |
| 2 | 301 | John Smith |
| 3 | 302 | Bob Wire |
| 4 | 303 | Tessie Sniffen |
| 5 | 304 | Greg Waters |
| 6 | 305 | Cristina Forbes |
| 7 | 306 | George Graham |
| 8 | 307 | Anthoy Mayo |
| 9 | 308 | Nick Young |
| 10 | 309 | Ken Mkay |
| 11 | 310 | Rebecca Acevedo |
| 12 | 311 | Leah Roman |
| 13 | 312 | Mark Quell |

# Create Table Statements

This table stores the names of Artists, their date of birth, and years singing. It is connected to the table decade each Artist may be apart of.

```
CREATE TABLE Artist (
        AID           Int not null,
        AName          Text not null,
        DOB           Int not null,
        YearsSinging  Int not null,
        DID           Int not null,
Primary key (AID),
Foreign key (DID) references Decade (DID)
);
```

Functional Dependencies: AID -> AName, DOB, YearsSinging, DID



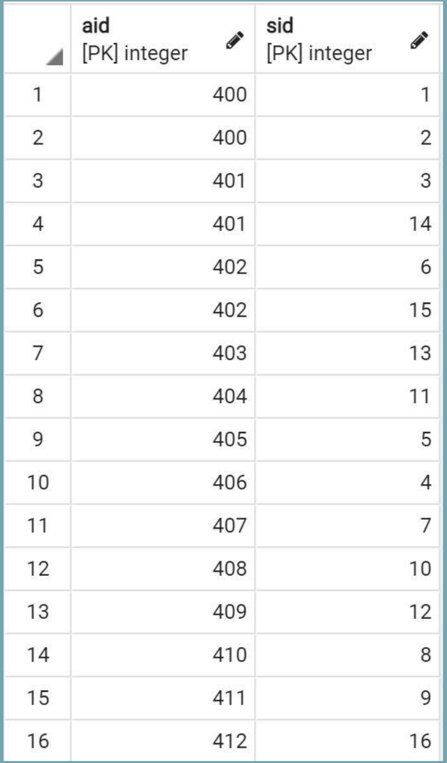
| | aid [PK] i | aname text | dob date | yearssinging.. integer | did integer |
|---|---|---|---|---|---|
| 1 | 400 | Billy Joel | 1951-01-09 | 54 | 3 |
| 2 | 401 | Elton John | 1955-07-22 | 48 | 4 |
| 3 | 402 | Daryl Hall | 1953-08-15 | 46 | 3 |
| 4 | 403 | Tyler Hubard | 1985-09-12 | 28 | 7 |
| 5 | 404 | Marc Roberge | 1979-07-09 | 32 | 5 |
| 6 | 405 | André Lauren Benjamin | 1977-06-24 | 27 | 5 |
| 7 | 406 | Freddie Mercury | 1960-02-18 | 42 | 2 |
| 8 | 407 | Alan Labouseur | 1970-04-19 | 2 | 8 |
| 9 | 408 | Bob Marley | 1948-03-28 | 58 | 1 |
| 10 | 409 | Frank Sinatra | 1945-05-02 | 57 | 1 |
| 11 | 410 | Steve Harwell | 1976-06-22 | 28 | 5 |
| 12 | 411 | John Popper | 1973-12-08 | 33 | 5 |
| 13 | 412 | Randy Hogan | 1971-11-13 | 36 | 4 |

## Create Table Statements

This table references Artists and their songs and serves as a connecting table between the two.

```
CREATE TABLE Artist_Song (
        AID          Int not null,
        SID          Int not null,
Primary key (AID, SID),
Foreign key (AID) references Artist (AID),
Foreign key (SID) references Song (SID)
);
```

Functional Dependencies: None



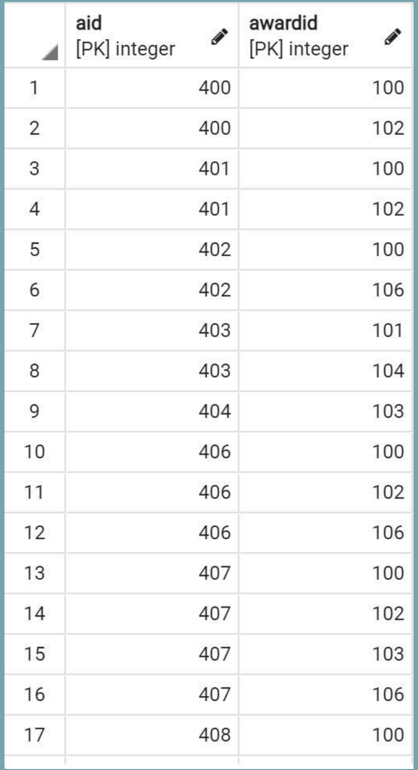
|  | aid [PK] integer | sid [PK] integer |
|---|---|---|
| 1 | 400 | 1 |
| 2 | 400 | 2 |
| 3 | 401 | 3 |
| 4 | 401 | 14 |
| 5 | 402 | 6 |
| 6 | 402 | 15 |
| 7 | 403 | 13 |
| 8 | 404 | 11 |
| 9 | 405 | 5 |
| 10 | 406 | 4 |
| 11 | 407 | 7 |
| 12 | 408 | 10 |
| 13 | 409 | 12 |
| 14 | 410 | 8 |
| 15 | 411 | 9 |
| 16 | 412 | 16 |

## Create Table Statements

This table references Artists and awards they may have won and is a connecting table between the two.

```
CREATE TABLE Artist_Award (
	AID		Int not null,
	AwardID		Int not null,
Primary key (AID, AwardID),
Foreign key (AID) references Artist (AID),
Foreign key (AwardID) references Award (AwardID)
);
```

Functional Dependencies: None



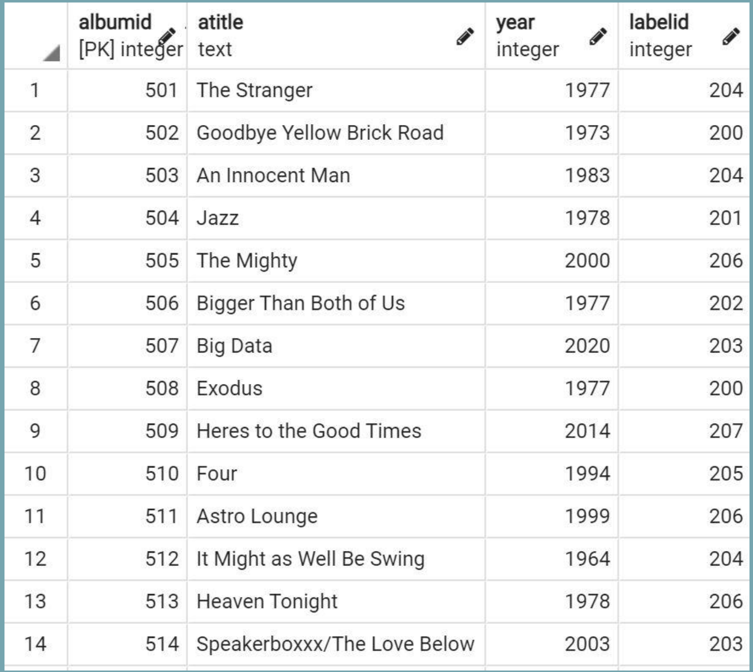
| | aid [PK] integer | awardid [PK] integer |
|---|---|---|
| 1 | 400 | 100 |
| 2 | 400 | 102 |
| 3 | 401 | 100 |
| 4 | 401 | 102 |
| 5 | 402 | 100 |
| 6 | 402 | 106 |
| 7 | 403 | 101 |
| 8 | 403 | 104 |
| 9 | 404 | 103 |
| 10 | 406 | 100 |
| 11 | 406 | 102 |
| 12 | 406 | 106 |
| 13 | 407 | 100 |
| 14 | 407 | 102 |
| 15 | 407 | 103 |
| 16 | 407 | 106 |
| 17 | 408 | 100 |

## Create Table Statements

This table stores the names of albums and the year each one came out. It is connected to the table Record_Label where each album is produced by a label.

```
CREATE TABLE Album (
        AlbumID        Int not null,
        ATitle         Text not null,
        Year           Int not null,
        LabelID        Int not null,
Primary key (AlbumID),
Foreign key (LabelID) references Record_Label
(LabelID)
);
```

Functional Dependencies: AlbumID -> ATitle, Year, LabelID



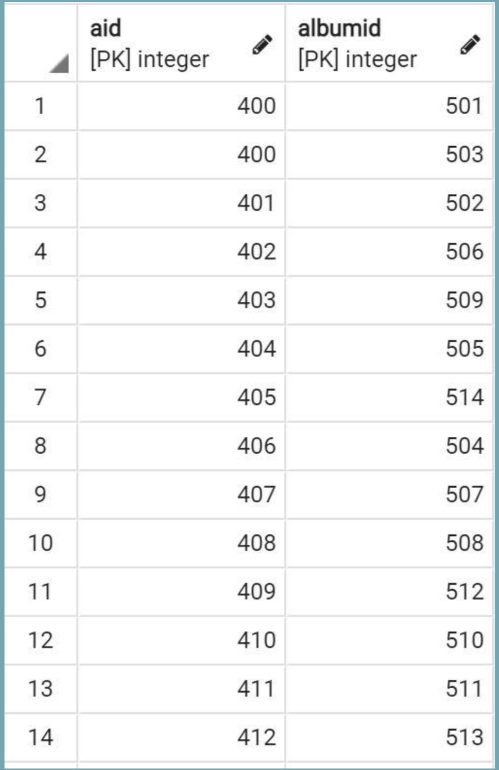
| | albumid [PK] integer | atitle text | year integer | labelid integer |
|---|---|---|---|---|
| 1 | 501 | The Stranger | 1977 | 204 |
| 2 | 502 | Goodbye Yellow Brick Road | 1973 | 200 |
| 3 | 503 | An Innocent Man | 1983 | 204 |
| 4 | 504 | Jazz | 1978 | 201 |
| 5 | 505 | The Mighty | 2000 | 206 |
| 6 | 506 | Bigger Than Both of Us | 1977 | 202 |
| 7 | 507 | Big Data | 2020 | 203 |
| 8 | 508 | Exodus | 1977 | 200 |
| 9 | 509 | Heres to the Good Times | 2014 | 207 |
| 10 | 510 | Four | 1994 | 205 |
| 11 | 511 | Astro Lounge | 1999 | 206 |
| 12 | 512 | It Might as Well Be Swing | 1964 | 204 |
| 13 | 513 | Heaven Tonight | 1978 | 206 |
| 14 | 514 | Speakerboxxx/The Love Below | 2003 | 203 |

## Create Table Statements

This table references Artists and their albums and serves as a connecting table between the two.

```
CREATE TABLE Artist_Album (
    AID         Int not null,
    AlbumID     Int not null,
Primary key (AID, AlbumID),
Foreign key (AID) references Artist (AID),
Foreign key (AlbumID) references Album (AlbumID)
);
```

Functional Dependencies: None



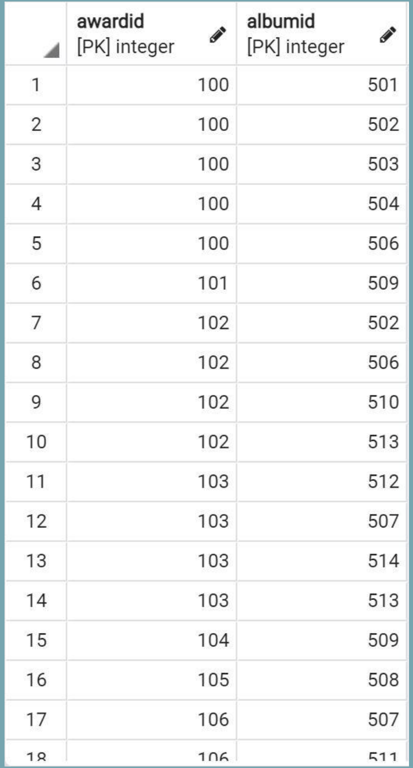
| | aid [PK] integer | albumid [PK] integer |
|---|---|---|
| 1 | 400 | 501 |
| 2 | 400 | 503 |
| 3 | 401 | 502 |
| 4 | 402 | 506 |
| 5 | 403 | 509 |
| 6 | 404 | 505 |
| 7 | 405 | 514 |
| 8 | 406 | 504 |
| 9 | 407 | 507 |
| 10 | 408 | 508 |
| 11 | 409 | 512 |
| 12 | 410 | 510 |
| 13 | 411 | 511 |
| 14 | 412 | 513 |

## Create Table Statements

This table references albums and their awards they may have won and serves as a connecting table between the two.

```
CREATE TABLE Album_Award (
	AwardID		Int not null,
	AlbumID		Int not null,
Primary key (AwardID, AlbumID),
Foreign key (AwardID) references Award (AwardID),
Foreign key (AlbumID) references Album (AlbumID)
);
```

Functional Dependencies: None



| | awardid [PK] integer | albumid [PK] integer |
|---|---|---|
| 1 | 100 | 501 |
| 2 | 100 | 502 |
| 3 | 100 | 503 |
| 4 | 100 | 504 |
| 5 | 100 | 506 |
| 6 | 101 | 509 |
| 7 | 102 | 502 |
| 8 | 102 | 506 |
| 9 | 102 | 510 |
| 10 | 102 | 513 |
| 11 | 103 | 512 |
| 12 | 103 | 507 |
| 13 | 103 | 514 |
| 14 | 103 | 513 |
| 15 | 104 | 509 |
| 16 | 105 | 508 |
| 17 | 106 | 507 |
| 18 | 106 | 511 |

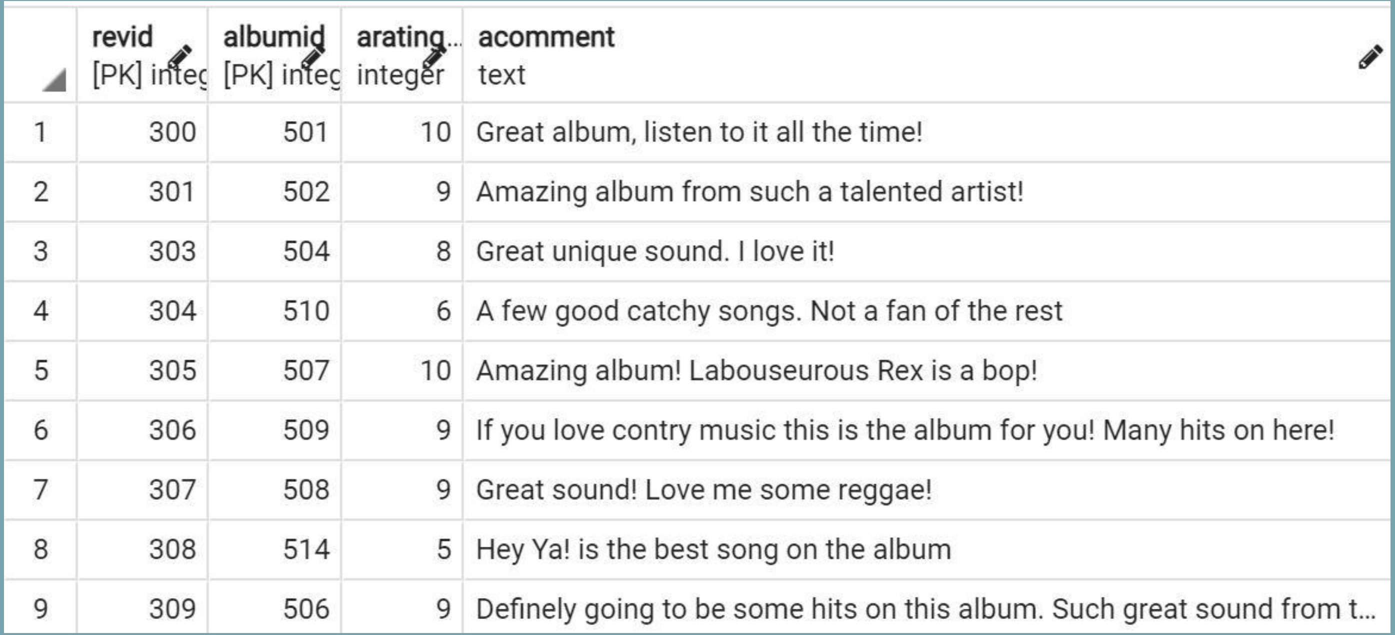
## Create Table Statements

```
CREATE TABLE Album_Reviews (
        RevID        Int not null,
        AlbumID      Int not null,
        ARating      Int not null,
        AComment     Text not null,
Primary key (RevID, AlbumID),
Foreign key (RevID) references Reviewers (RevID),
Foreign key (AlbumID) references Album (AlbumID)
);
```

Functional Dependencies: RevID, AlbumID -> ARating, AComment

Data for Album_Reviews table.

This table references albums and reviews associated with them. This table also stores both a rating and a comment reviewers can make on an albums.



| | revid [PK] integ | albumid [PK] integ | arating... integer | acomment text |
|---|---|---|---|---|
| 1 | 300 | 501 | 10 | Great album, listen to it all the time! |
| 2 | 301 | 502 | 9 | Amazing album from such a talented artist! |
| 3 | 303 | 504 | 8 | Great unique sound. I love it! |
| 4 | 304 | 510 | 6 | A few good catchy songs. Not a fan of the rest |
| 5 | 305 | 507 | 10 | Amazing album! Labouseurous Rex is a bop! |
| 6 | 306 | 509 | 9 | If you love contry music this is the album for you! Many hits on here! |
| 7 | 307 | 508 | 9 | Great sound! Love me some reggae! |
| 8 | 308 | 514 | 5 | Hey Ya! is the best song on the album |
| 9 | 309 | 506 | 9 | Definely going to be some hits on this album. Such great sound from t... |

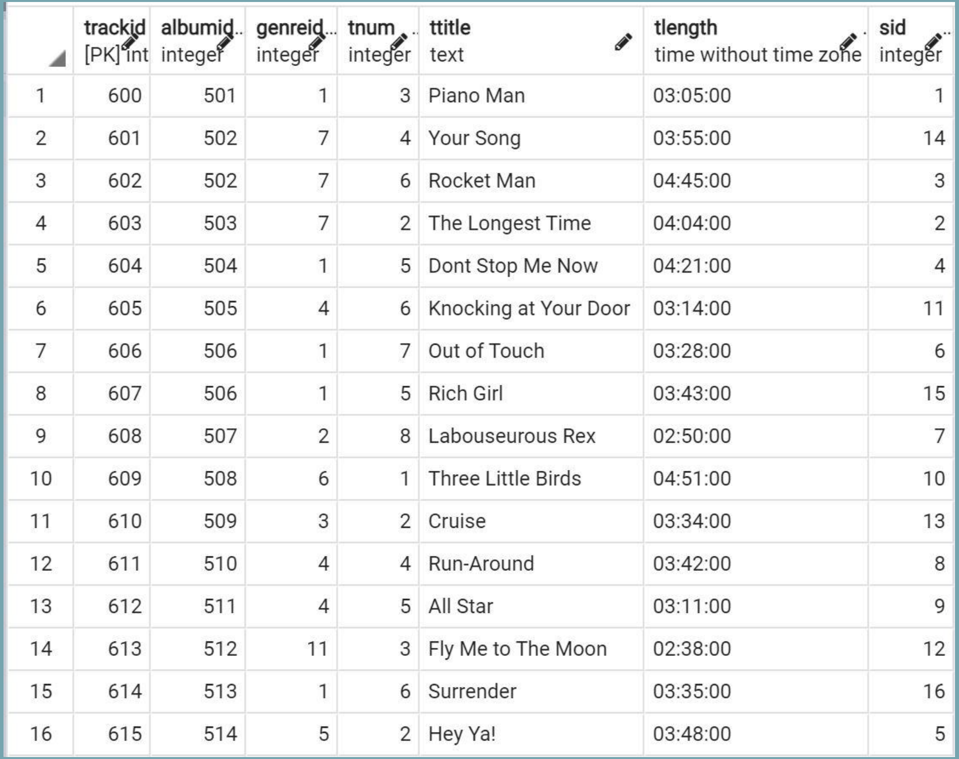
## Create Table Statements

```
CREATE TABLE Track (
    TrackID     Int not null,
    AlbumID     Int not null,
    GenreID     Int not null,
    TNum        Int not null,
    TTitle      Text not null,
    TLength     Time not null,
    SID         Int not null,
Primary key (TrackID),
Foreign key (AlbumID) references Album (AlbumID),
Foreign key (GenreID) references Genre (GenreID),
Foreign key (SID) references Song (SID)
);
```

Functional Dependencies: TrackID -> TNum, TTitle, TLength, AlbumID, GenreID, SID

## Data for Track table.

This table references the song, album, and genre tables. A track can be for a song, be in an album, and can have a genre of music associated with it. This table stores a track number, a track title, and a track length.



| | trackid [PK] int | albumid.. integer | genreid... integer | tnum . integer | ttitle text | tlength time without time zone | sid integer |
|---|---|---|---|---|---|---|---|
| 1 | 600 | 501 | 1 | 3 | Piano Man | 03:05:00 | 1 |
| 2 | 601 | 502 | 7 | 4 | Your Song | 03:55:00 | 14 |
| 3 | 602 | 502 | 7 | 6 | Rocket Man | 04:45:00 | 3 |
| 4 | 603 | 503 | 7 | 2 | The Longest Time | 04:04:00 | 2 |
| 5 | 604 | 504 | 1 | 5 | Dont Stop Me Now | 04:21:00 | 4 |
| 6 | 605 | 505 | 4 | 6 | Knocking at Your Door | 03:14:00 | 11 |
| 7 | 606 | 506 | 1 | 7 | Out of Touch | 03:28:00 | 6 |
| 8 | 607 | 506 | 1 | 5 | Rich Girl | 03:43:00 | 15 |
| 9 | 608 | 507 | 2 | 8 | Labouseurous Rex | 02:50:00 | 7 |
| 10 | 609 | 508 | 6 | 1 | Three Little Birds | 04:51:00 | 10 |
| 11 | 610 | 509 | 3 | 2 | Cruise | 03:34:00 | 13 |
| 12 | 611 | 510 | 4 | 4 | Run-Around | 03:42:00 | 8 |
| 13 | 612 | 511 | 4 | 5 | All Star | 03:11:00 | 9 |
| 14 | 613 | 512 | 11 | 3 | Fly Me to The Moon | 02:38:00 | 12 |
| 15 | 614 | 513 | 1 | 6 | Surrender | 03:35:00 | 16 |
| 16 | 615 | 514 | 5 | 2 | Hey Ya! | 03:48:00 | 5 |

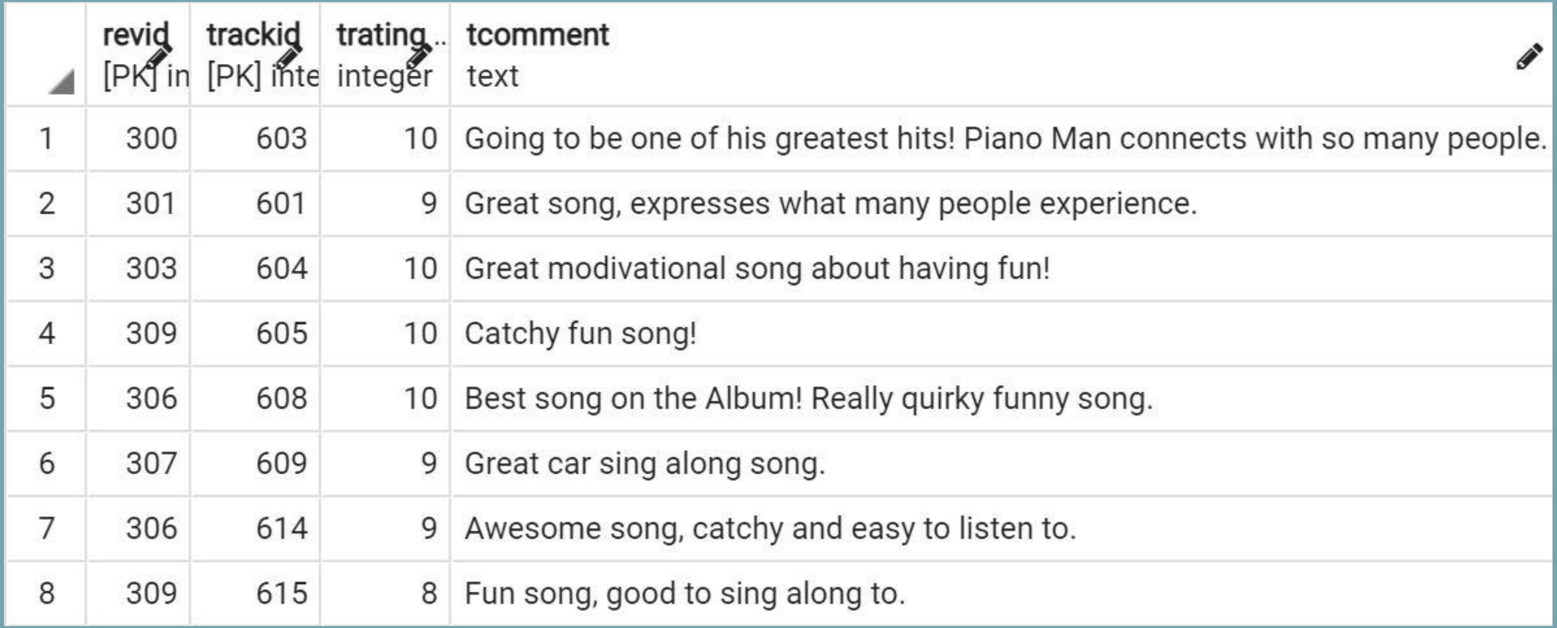
## Create Table Statements

```
CREATE TABLE Track_Reviews (
	RevID		Int not null,
	TrackID		Int not null,
	TRating		Int not null,
	TComment	Text not null,
Primary key (RevID, TrackID),
Foreign key (RevID) references Reviewers (RevID),
Foreign key (TrackID) references Track (TrackID),
);
```

Functional Dependencies: RevID, TrackID -> TRating, TComment

Data for Track_Reviews table.

This table references tracks and reviews associated with them. This table also stores both a rating and a comment reviewers can make on tracks.



| | revid [PK] in | trackid [PK] inte | trating... integer | tcomment text |
|---|---|---|---|---|
| 1 | 300 | 603 | 10 | Going to be one of his greatest hits! Piano Man connects with so many people. |
| 2 | 301 | 601 | 9 | Great song, expresses what many people experience. |
| 3 | 303 | 604 | 10 | Great modivational song about having fun! |
| 4 | 309 | 605 | 10 | Catchy fun song! |
| 5 | 306 | 608 | 10 | Best song on the Album! Really quirky funny song. |
| 6 | 307 | 609 | 9 | Great car sing along song. |
| 7 | 306 | 614 | 9 | Awesome song, catchy and easy to listen to. |
| 8 | 309 | 615 | 8 | Fun song, good to sing along to. |

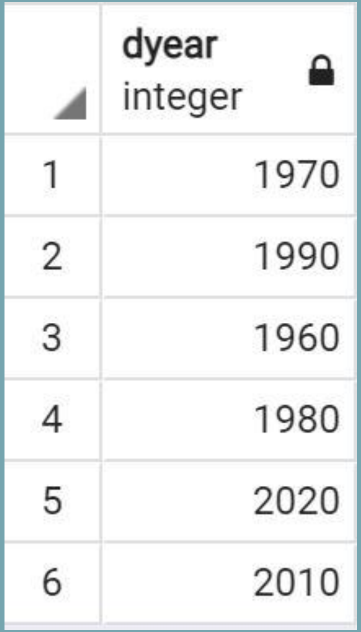
## Views

View: Artist Decade

```
Create View ArtistDecade
As
Select DYear
From Decade
Where DID In
      (Select DID
       From Artist
       Where DOB >= '01-01-1950'
       And DOB <= '01-01-2000');

Select *
From ArtistDecade;
```

Show the decades Artists are apart of based on date of birth of Artists. This can be helpful in finding Artists DOBs between a set range, in this case DOB greater than or equal to '01-01-1950' and DOB less than or equal to '01-01-2000'.



| | dyear integer |
|---|---|
| 1 | 1970 |
| 2 | 1990 |
| 3 | 1960 |
| 4 | 1980 |
| 5 | 2020 |
| 6 | 2010 |

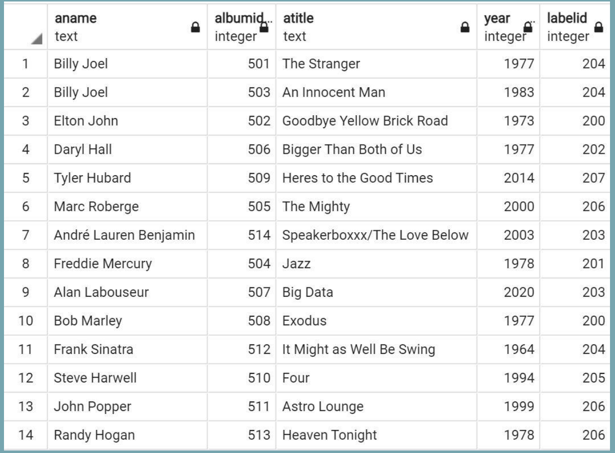
## Views

This view gives information about albums and the Artists they are by. This can be helpful in finding which albums are by which Artists, as well as the year the album was released.

View: Artist Album

```
Create View ArtistAlbum
As
Select A.AName, Al.AlbumID, Al.ATitle, Al.Year, Al.LabelID
From Artist_Album As ArtAl
Inner Join Artist As A On ArtAl.AID = A.AID
Inner Join Album As Al On ArtAl.AlbumID = Al.AlbumID;

Select *
From ArtistAlbum;
```



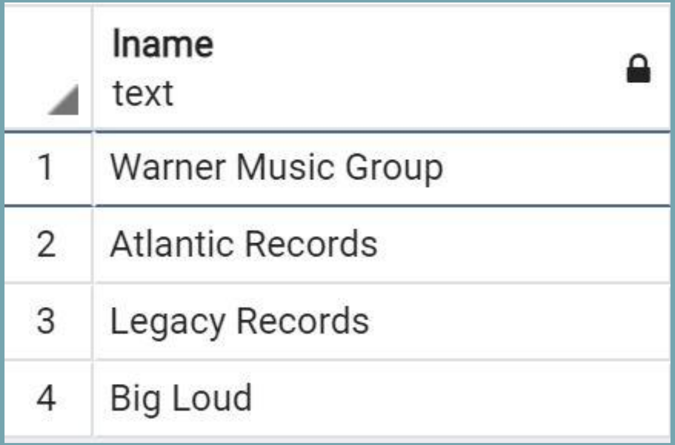
| | aname text | albumid integer | atitle text | year integer | labelid integer |
|---|---|---|---|---|---|
| 1 | Billy Joel | 501 | The Stranger | 1977 | 204 |
| 2 | Billy Joel | 503 | An Innocent Man | 1983 | 204 |
| 3 | Elton John | 502 | Goodbye Yellow Brick Road | 1973 | 200 |
| 4 | Daryl Hall | 506 | Bigger Than Both of Us | 1977 | 202 |
| 5 | Tyler Hubard | 509 | Heres to the Good Times | 2014 | 207 |
| 6 | Marc Roberge | 505 | The Mighty | 2000 | 206 |
| 7 | André Lauren Benjamin | 514 | Speakerboxxx/The Love Below | 2003 | 203 |
| 8 | Freddie Mercury | 504 | Jazz | 1978 | 201 |
| 9 | Alan Labouseur | 507 | Big Data | 2020 | 203 |
| 10 | Bob Marley | 508 | Exodus | 1977 | 200 |
| 11 | Frank Sinatra | 512 | It Might as Well Be Swing | 1964 | 204 |
| 12 | Steve Harwell | 510 | Four | 1994 | 205 |
| 13 | John Popper | 511 | Astro Lounge | 1999 | 206 |
| 14 | Randy Hogan | 513 | Heaven Tonight | 1978 | 206 |

## Views

```
View: Album Record Label
Create View AlbumRecordLabel
As
Select LName
From Record_Label
Where LabelID In
      (Select LabelID
       From Album
       Where Year > 1990);

Select *
From AlbumRecordLabel;
```

This view shows the record labels names for albums released after 1990. This could be helpful with finding albums after a certain time, in this case it is 1990.



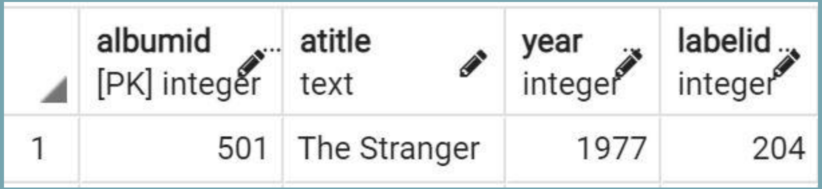
|  | lname<br>text |
|---|---|
| 1 | Warner Music Group |
| 2 | Atlantic Records |
| 3 | Legacy Records |
| 4 | Big Loud |

## Stored Procedures

```
create or replace function findAlbum(text, REFCURSOR) returns refcursor as
$$
declare
   _findAlbum     text      := $1;
   resultset      REFCURSOR := $2;
begin
   open resultset for
      Select *
      From Album
      Where ATitle = _findAlbum;
   return resultset;
end;
$$
language plpgsql;

Select findAlbum('The Stranger', 'results');
Fetch all from results;
```

This procedure provides an easy way to find albums based on only their album IDs.



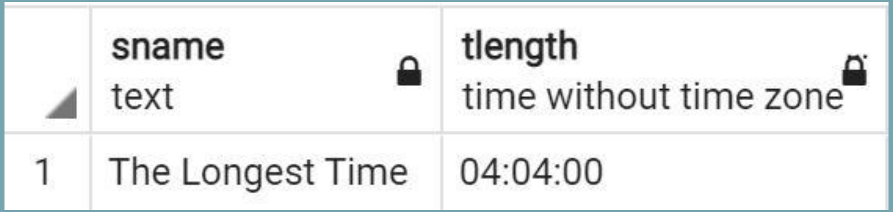
|   | albumid [PK] integer | atitle text | year integer | labelid integer |
|---|---|---|---|---|
| 1 | 501 | The Stranger | 1977 | 204 |

## Stored Procedures

```
create or replace function findSongs(text, int, REFCURSOR) returns refcursor as
$$
declare
  _findSong      text       := $1;
  _findS              int           := $2;
  resultset      REFCURSOR := $3;
begin
  open resultset for
    Select S.SName, T.TLength
    From Song as S, Track as T
    Where S.SName = _findSong
        And T.SID = _findS;
  return resultset;
end;
$$
language plpgsql;

Select findSongs('The Longest Time', '2','results');
Fetch all from results;
```

This procedure provides an easy way to find the length of songs based on their names and song IDs.

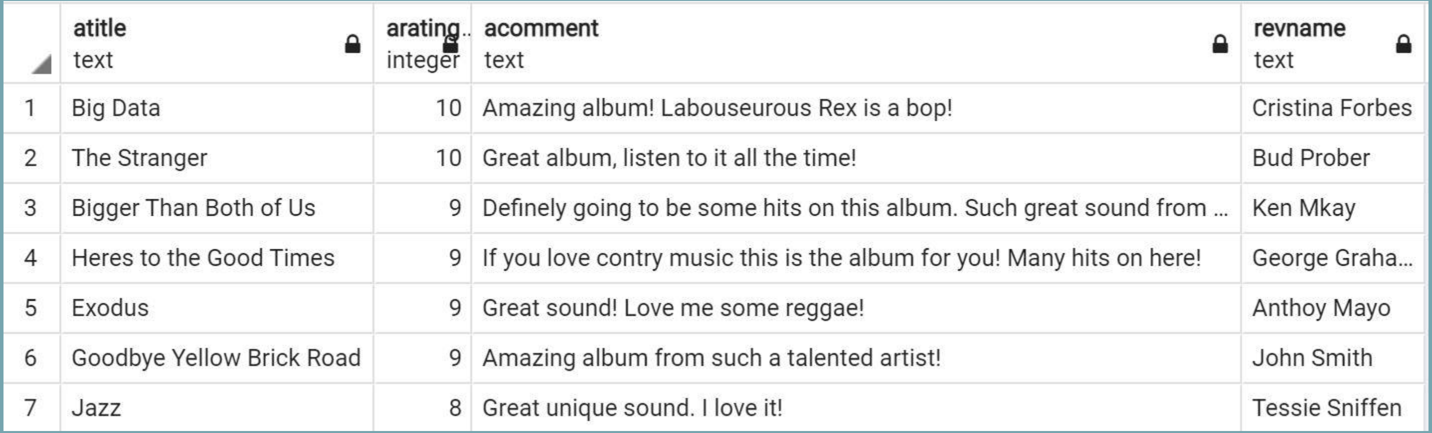


| | sname text | tlength time without time zone |
|---|---|---|
| 1 | The Longest Time | 04:04:00 |

# Reports/Interesting Queries

```
Select ATitle, ARating, AComment, RevName
From Album, Album_Reviews, Reviewers
Where Album.AlbumID = Album_Reviews.AlbumID
And Album_Reviews.RevID = Reviewers.RevID
And ARating >= 8
Order By ARating DESC;
```

This query displays the album title, album rating, and album comments made by reviewers where the rating made is greater than or equal to 8, sorted from high to low. This could be helpful in finding excellent rated albums.

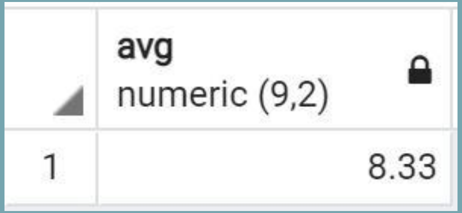


| | atitle<br>text | arating..<br>integer | acomment<br>text | revname<br>text |
|---|---|---|---|---|
| 1 | Big Data | 10 | Amazing album! Labouseurous Rex is a bop! | Cristina Forbes |
| 2 | The Stranger | 10 | Great album, listen to it all the time! | Bud Prober |
| 3 | Bigger Than Both of Us | 9 | Definely going to be some hits on this album. Such great sound from ... | Ken Mkay |
| 4 | Heres to the Good Times | 9 | If you love contry music this is the album for you! Many hits on here! | George Graha... |
| 5 | Exodus | 9 | Great sound! Love me some reggae! | Anthoy Mayo |
| 6 | Goodbye Yellow Brick Road | 9 | Amazing album from such a talented artist! | John Smith |
| 7 | Jazz | 8 | Great unique sound. I love it! | Tessie Sniffen |

## Reports/Interesting Queries

This query finds the average album rating made by reviewers down to decimal. This could be helpful in finding what the average rating of albums are.

```
Select Cast(AVG(ARating) As Decimal(9,2))
From Album_Reviews;
```

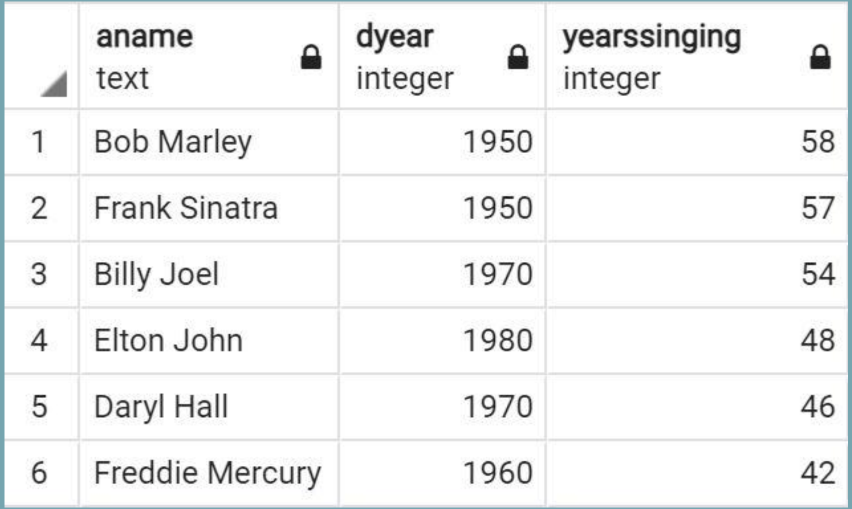


```
    avg
    numeric (9,2)
1           8.33
```

## Reports/Interesting Queries

This query returns the artists names, decades, and number of years singing of artists who have been singing for at least 40 years. Sorted by years singing from high to low.

```
Select A.AName, D.DYear, A.YearsSinging
From Artist As A, Decade As D
Where A.DID = D.DID
And A.YearsSinging >= 40
Order By A.YearsSinging DESC;
```



| | aname text | dyear integer | yearssinging integer |
|---|---|---|---|
| 1 | Bob Marley | 1950 | 58 |
| 2 | Frank Sinatra | 1950 | 57 |
| 3 | Billy Joel | 1970 | 54 |
| 4 | Elton John | 1980 | 48 |
| 5 | Daryl Hall | 1970 | 46 |
| 6 | Freddie Mercury | 1960 | 42 |

## Triggers

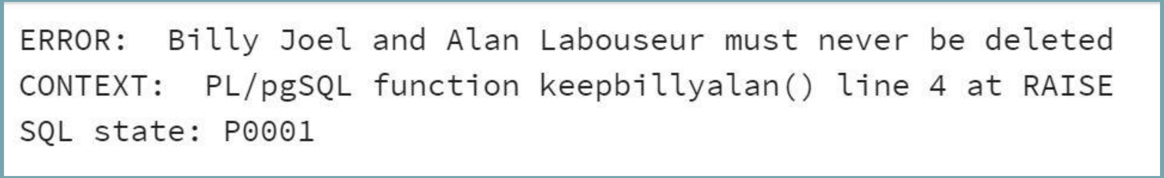
```
Create Or Replace Function keepBillyAlan() Returns Trigger As
$$
Begin
    If Old.AName = 'Billy Joel' And Old.AName = 'Alan Labouseur'
    Then Raise Exception 'Billy Joel and Alan Labouseur must never be deleted';
    End If;
End;
$$
Language plpgsql;

Create Trigger keepBillyAlan Before Delete on Artist
For Each Row Execute Procedure keepBillyAlan();

Delete From Artist
Where AName = 'Billy Joel';

Delete From Artist
Where AName = 'Alan Labouseur';
```

This trigger helps to ensure that the artists 'Billy Joel' and 'Alan Labouseur' are never deleted from the table. If attempted to be done this error will pop up.



```
ERROR:  Billy Joel and Alan Labouseur must never be deleted
CONTEXT:  PL/pgSQL function keepbillyalan() line 4 at RAISE
SQL state: P0001
```

## Security

```
Create Role AlbumManager
Grant All
On Album, Record_Label, Track, Song
In Schema Public To AlbumManager;

Create Role Reviewers;
Grant Select, Insert, Update, Delete
On Album_Reviews, Track_Reviews
To Reviewers;

Grant Select (ARating, AComment)
On Album_Reviews
To Reviewers;

Grant Select (TRating, TComment)
On Track_Reviews
To Reviewers;
```

Album Manager: This is the manager of albums. This person needs to be able to edit all information regarding songs, tracks, record labels, and the album of course.

Reviewers: This is the Reviewers. They need to be able to insert, update, and delete a rating and or comment from either an album or track review.

- The Select in this second part can be replaced with Insert, Update, and or Delete to grant those rights.

## Security

```
Create Role Artist;
Grant Select, Insert, Update, Delete
On Artist
To Artist;

Grant Select (AName, DOB, YearsSinging)
On Artist
To Artist;
```

Artists: This is the Artists. They need to be able to select, insert, update, and delete their name, date of birth, and or their years singing.

- The Select in this second part can be replaced with Insert, Update, and or Delete to grant those rights.

## Implementation Notes

This database can be used to help organize Artists music of all kinds, from all decades and styles of music. It can also help manage what awards Artists received and reviews made on both albums and tracks.

If implemented there would have to be more data, as this is just sample data. There could even be more fields/columns added to tables as needed.

## Problems/Enhancements/Conclusions

- Upon looking at my ER diagram at the end of my project I noticed in the Award table there could have been award labels for each award name. If the award was a Grammy award there might be different types such as Best Pop Solo Performance, Best Rock Vocal Album, etc.
- If needed depending on implementation more fields/columns can be added to tables such as Artist, Album, Reviewers, etc.
- Tables such as Decade and Genre, if there are no other field to be added they could potentially be added to the tables they connect to. (Decade can go in Artist and Genre can go in track)
- A band table or Artist group table could be created to account for band names such as Aerosmith, Queen, Fleetwood Mac, as well as others.
- If this database were to be implemented on a music listening/streaming platform other tables such as playlist, account, and others would be needed.
- This was a very useful project as I learned a lot about what it takes to build and manage a database system.