Object-Relational Databases 42901
Database Design And Implementation Assignment
(Online Music Database)

Student Details

Name	Student_ID
Piyush Kumar	13677493
Vraj Mehta	13488642

Oracle Username – 13488642 (Assignment Implemented)

Introduction

This assignment explains the design for Online music database with the help of logical model and relational model in barker notations with fully normalized form in section 2.1. We will also be creating an object relational design implementing collections including variable array and nested table (features of object-relational database). In order to deal with disk type, we will be using super-type feature for “Mp3” and “Audio CDs & Vinyl Albums” as these both entities will be the sub type of Albums entity. After that we will insert all the data as provided for the database in section 2.2.

Apart from that, we will go through “Methods and Queries” in section 2.3, in which queries will show how to use max, avg, intersect, union, nvl and many other pre-defined functions. With the help of queries, we will represent implementation of object types and fetching the values from the types using treat function. These queries will also demonstrate how to implement method and view. Not only implementation, it will also describe how to modify view.

In the Final section “2.4 Discussion”, we will compare our both designs (Relational and Object-Relational design) from section 2.1 and 2.2 and will highlight the advantages and features of both design and which is better to be implemented in the given scenario for online music database (OMDB).

Table of Contents:

2.1 Entity-Relationship Model And Relational Design (A). Part I – Logical Model

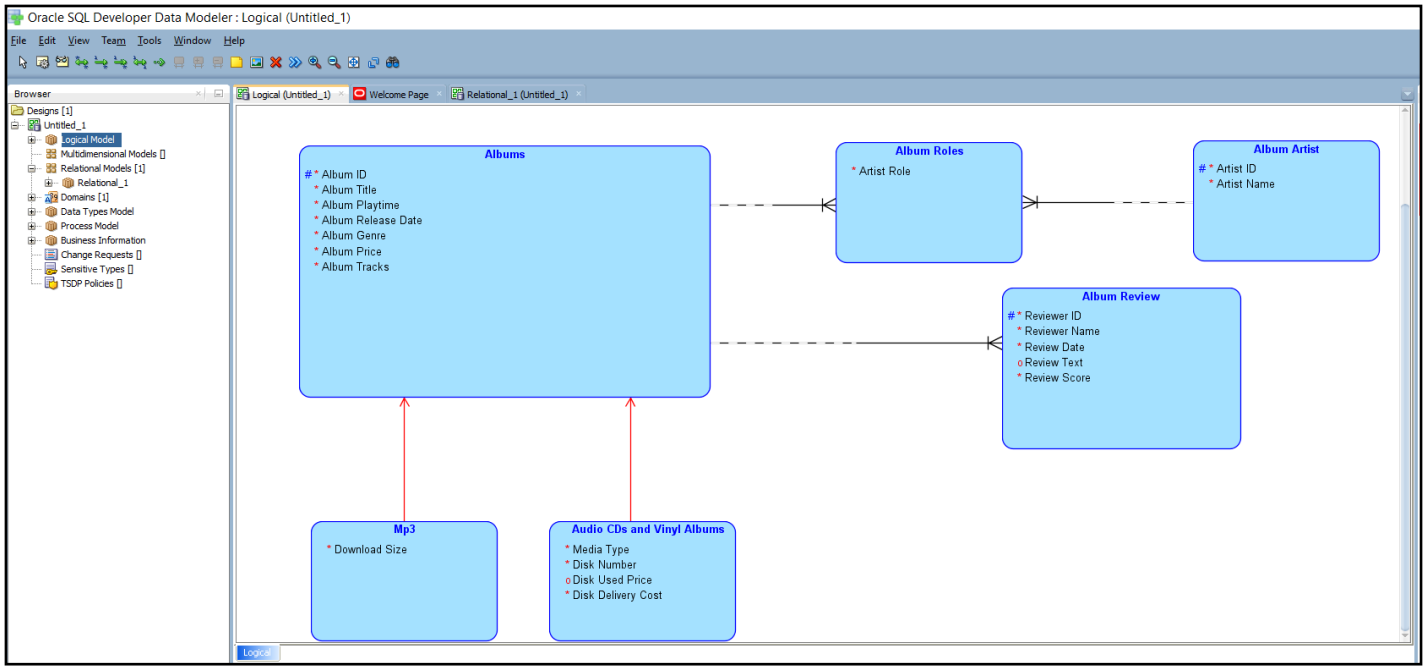
ER (Entity Relationship) diagram is kind of a flowchart that represents how different entities within a system are interrelated. This logical model is considered as the easiest way to understand the relationship between the objects such as tables.

According to the given scenario of online music database (OMDB), Albums is the main table containing Album Title, Album Playtime, Album Release Date, Album Genre, Album Price, Album Tracks columns. There are two subtypes of the Albums table and that are "Mp3" and another one is "Audio CDs and Vinyl Albums" because according to the given dataset it is clear that a single album can have different media types and that is the reason these media types are the subtype of Albums table. Mp3 subtype contains one column and that is Download Size. Whereas, in case of Audio CDs and Vinyl disk subtype there are Media Type, Disk Number, Disk Used Price, Disk Delivery Cost. Due to different column exists in "Mp3" and "Audio CDs and Vinyl Albums" both are considered as different subtypes of Albums. There is another table exist and that is Album Artists that has Artist Name, Artist ID.

The **relationship** between Albums and Album Roles is one to many because there can be many roles for one album. The relationship between Album Artist and Album Roles is one to many as there can multiple roles for a single Artist. Album Reviews is one table that consists of Reviewer Name, Review Date, Review Text, Review Score. The relationship between Albums and Album Reviews is one to many.

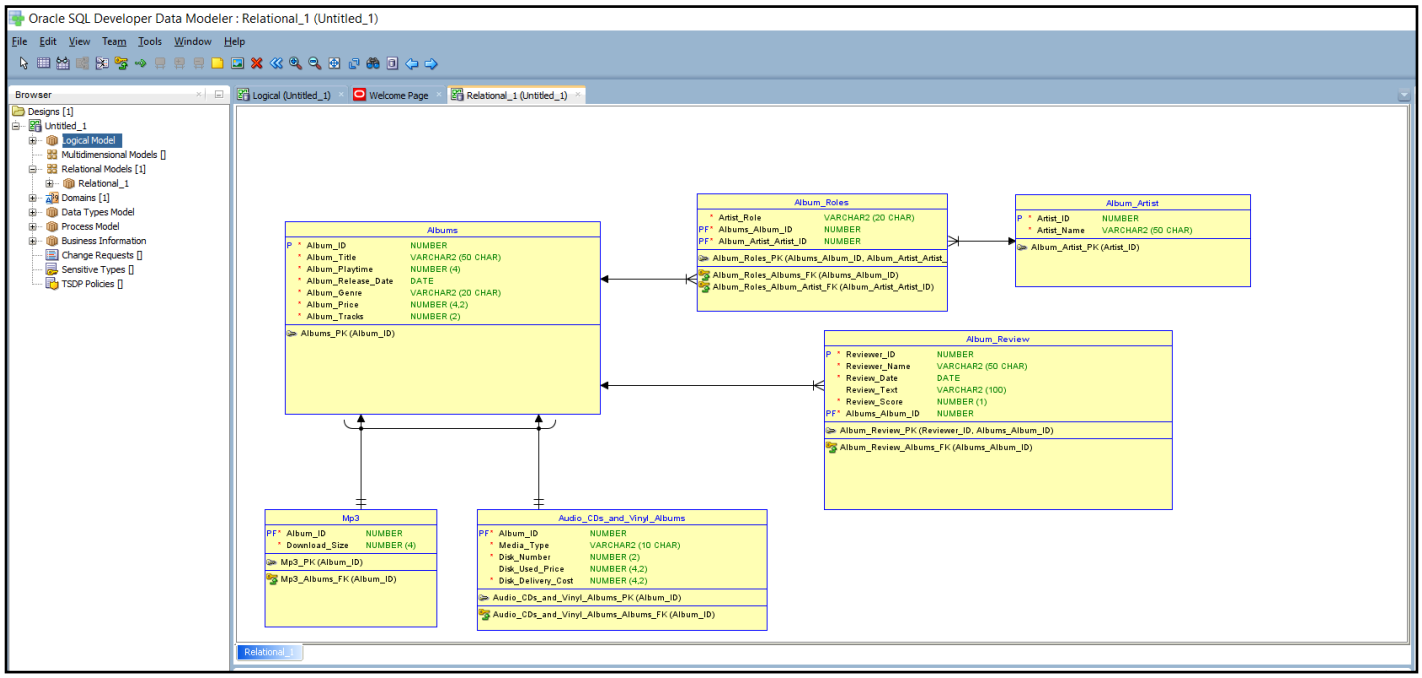
Album ID is the **unique identifier** for Albums. Similarly, Artist ID & Reviewer ID are the unique identifiers for Album Artists and Album Review respectively. We have created unique identifier as no suitable attribute exists.

Below if the Figure for the **Logical Model** for the given scenario. It also shows whether a attribute should be mandatory or not.



(B). Part II – Relational Model

With the help of "Logical designed Entity Relationship Model" we have engineer it to "Relational Model".



As Above Table Diagram or Relational Model is not completely normalized. In order to Normalize this Model, we need to find out the functional dependencies and then remove them from this model.

So, Table is "**Albums**" that contains (Albums_ID, Album_Title, Album_Playtime, Album_Release_Date, Album_Genre, Album_Price, Album_Tracks) attributes has dependencies such as

Album_Title --> Album_Playtime, Album_Tracks, Album_Genre

Albums_ID, Album_Title --> Album_Release_Date, Album_Price

Normalizing the given Scenario.

Step 1. Find merged minimal cover of Functional Dependencies, which contains:
Album_Title --> Album_Playtime, Album_Tracks, Album_Genre

Albums_ID, Album_Title --> Album_Release_Date,Album_Price

Initially relation [1] contains the original table, with the Functional Dependencies above

Check: Checking whether table relation [1] is in BCNF

The Functional Dependency [Album_Title --> Album_Playtime, Album_Tracks,

Album_Genre] violates BCNF as the Left-Hand Side is not super key. Table is split into the two below:

relation [2] = (Album_Title, Album_Playtime, Album_Tracks, Album_Genre)
With Functional Dependencies: There are no functional Dependency in relation 2.

relation [3] = (Albums_ID, Alum_Title, Album_Release_Date, Album_Price)
With Functional Dependencies: There are no functional Dependency in relation 3.

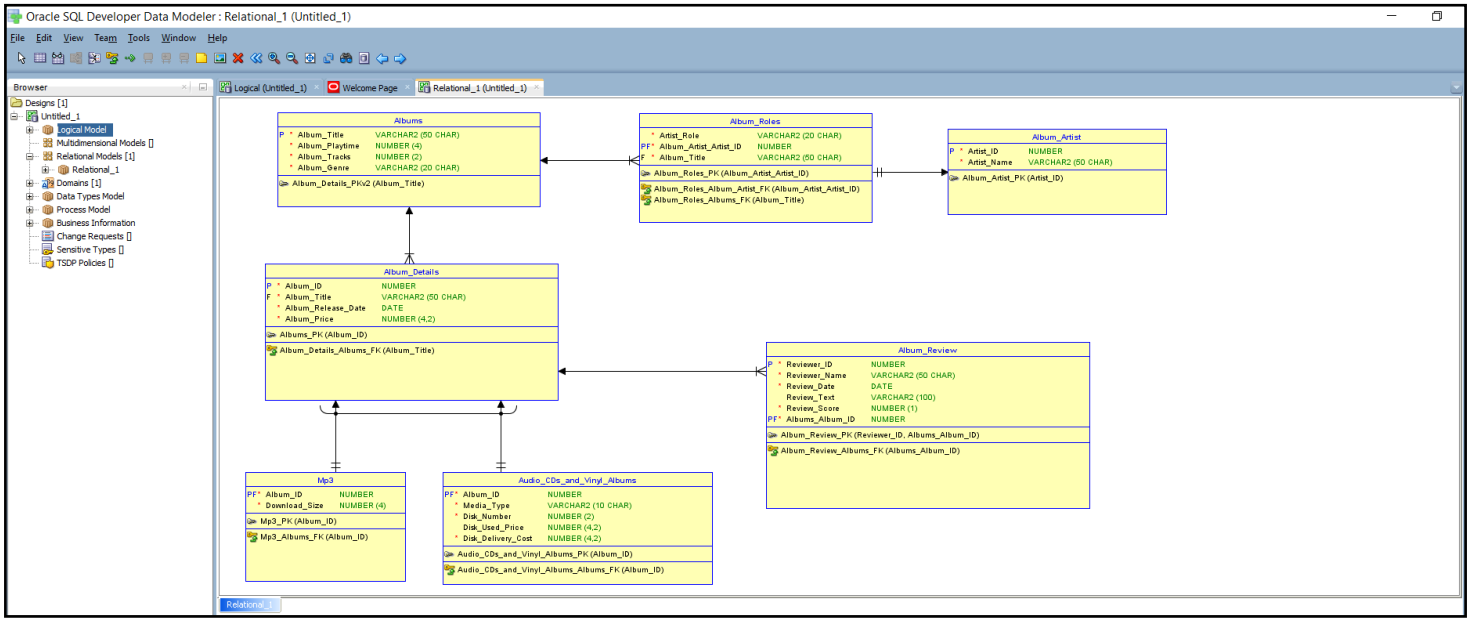
Check: Checking whether table relation [2] is in normalized form

Yes, Relation [2] – (Album_Title, Album_Playtime, Album_Tracks, Album_Genre) is in fully normalized form because it follows 1NF (as it has all atomic values), 2NF (as there is no partial Dependency), 3NF (As there is no Transitive Dependency), BCNF (as in this case now LHS is a super key).

Check: Checking whether table relation [3] is in normalized form

Yes, Relation [3] – (Albums_ID, Alum_Title, Album_Release_Date, Album_Price) is in fully normalized form because it follows 1NF (as it has all atomic values), 2NF (as there is no partial Dependency), 3NF (As there is no Transitive Dependency), BCNF (as in this case now LHS is a super key).

Applying Normalization to Relational Model –



As table Albums is now split into two new entities that are Albums that contains (Album_Title, Album_Playtime, Album_Tracks, Album_Genre) and Album_Details that contains (Album_ID, Album_Title, Album_Release_Date, Album_Price). Due to the splitting of table Album, there will be certain changes in relationships between entities.

- Albums entity has one to many relation with Album_Roles.
- Albums table has one to many relation with Album_Details.
- Albums_Details entity has one to many relation with Album_Review.
- MP3 entity has one to one relation with Album_Details.
- Audio_CDs and Vinyl_Albums table has one to one relation with Album_Details.

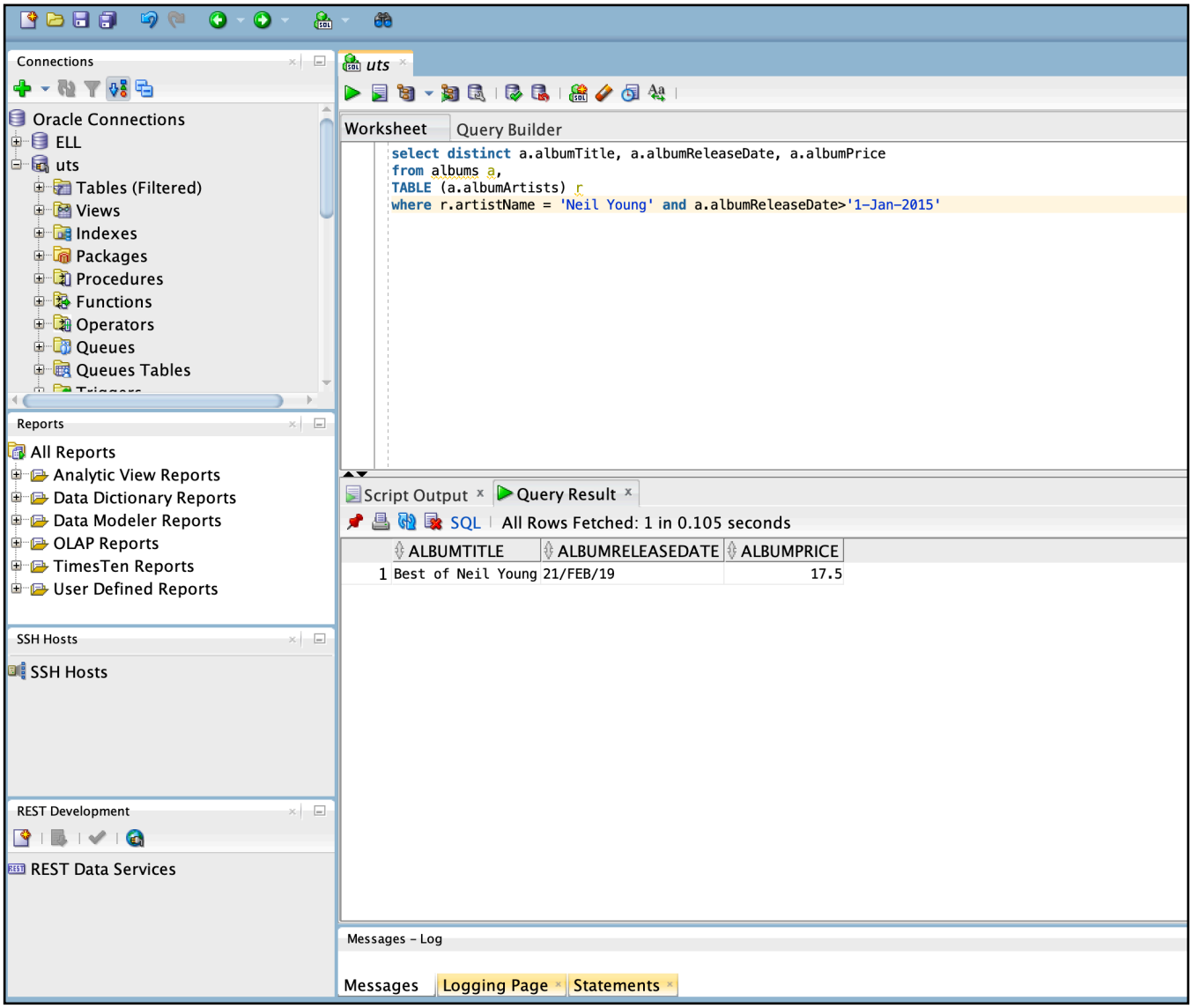
2.3 Omdb Methods And Queries

1. **Give album title, album release date and album price of all Neil Young's albums released after 1st January 2015.**

Query:

```
select distinct a.albumTitle, a.albumReleaseDate, a.albumPrice
from albums a,
TABLE (a.albumArtists) r
where r.artistName = 'Neil Young' and a.albumReleaseDate>'1-Jan-2015'
```

Result:

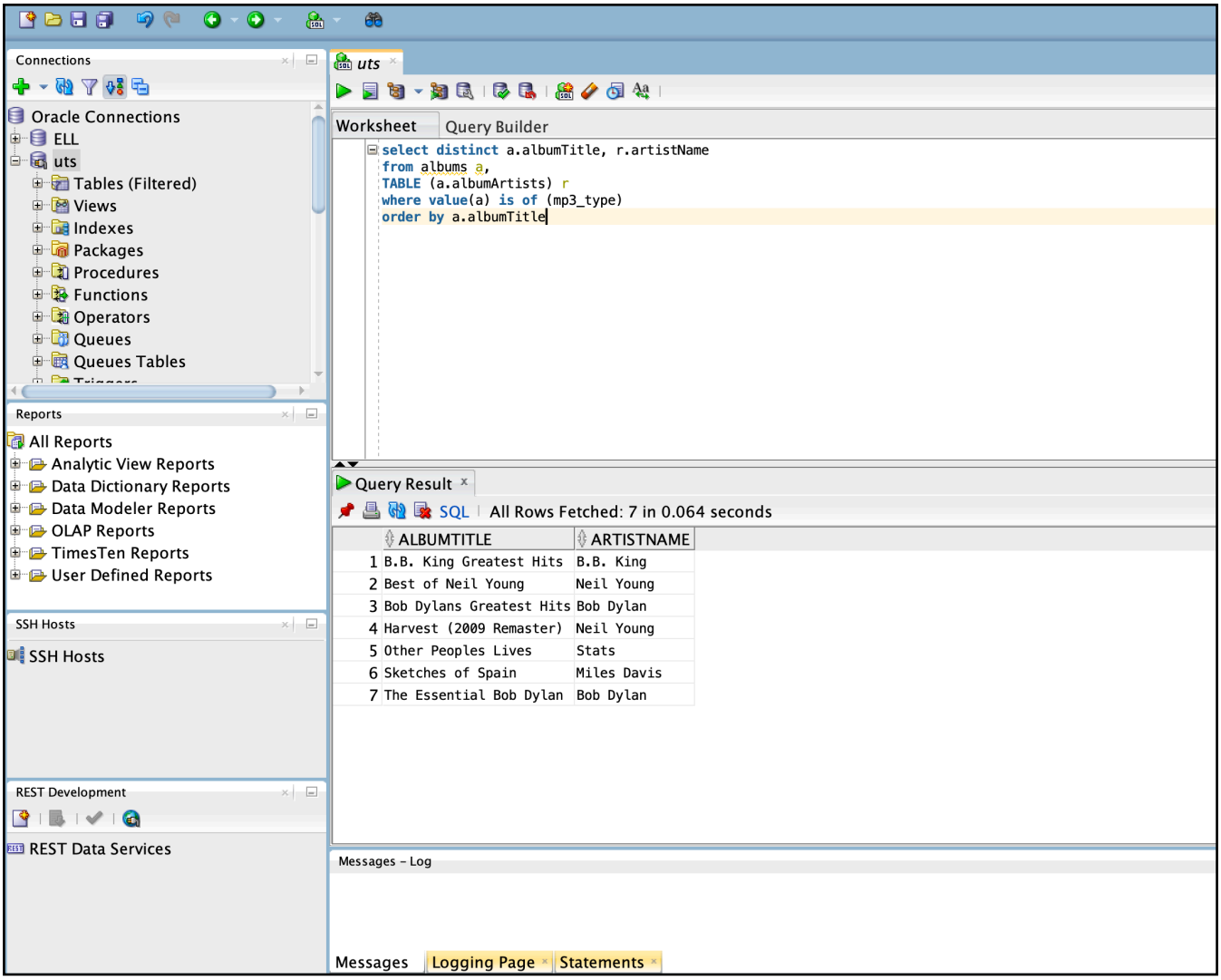


2. Give album title and artist name for albums released only in MP3 format. Order by album title.

Query:

```
select distinct a.albumTitle, r.artistName
from albums a,
TABLE (a.albumArtists) r
where value(a) is of (mp3_type)
order by a.albumTitle
```

Result:

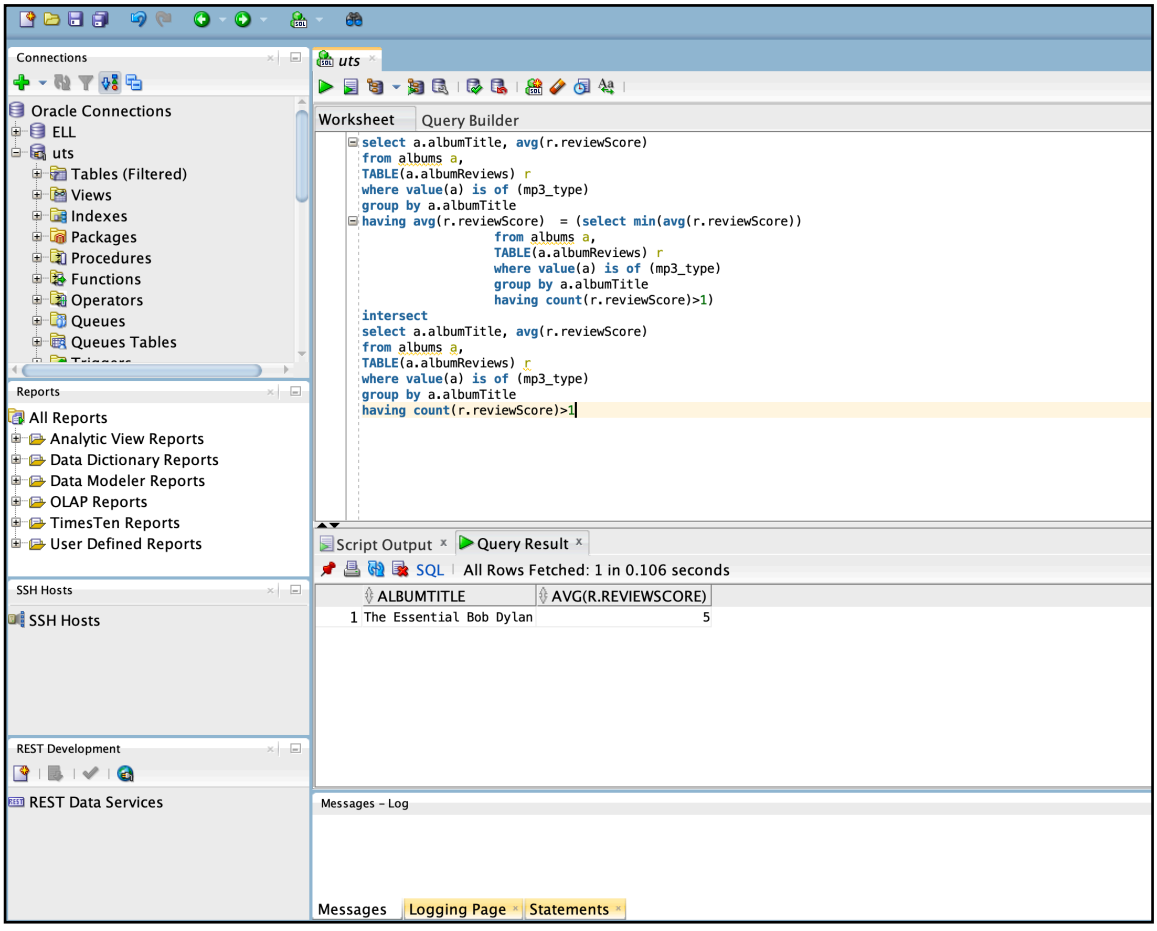


3. Give lowest rated MP3 album (i.e. album with the lowest average review score). Show album title and the average score. Exclude albums with only one review.

Query:

```
select a.albumTitle, avg(r.reviewScore)
from albums a,
TABLE(a.albumReviews) r
where value(a) is of (mp3_type)
group by a.albumTitle
having avg(r.reviewScore)  = (select min(avg(r.reviewScore))
                              from albums a,
                              TABLE(a.albumReviews) r
                              where value(a) is of (mp3_type)
                              group by a.albumTitle
                              having count(r.reviewScore)>1)
intersect
select a.albumTitle, avg(r.reviewScore)
from albums a,
TABLE(a.albumReviews) r
where value(a) is of (mp3_type)
group by a.albumTitle
having count(r.reviewScore)>1
```

Result:

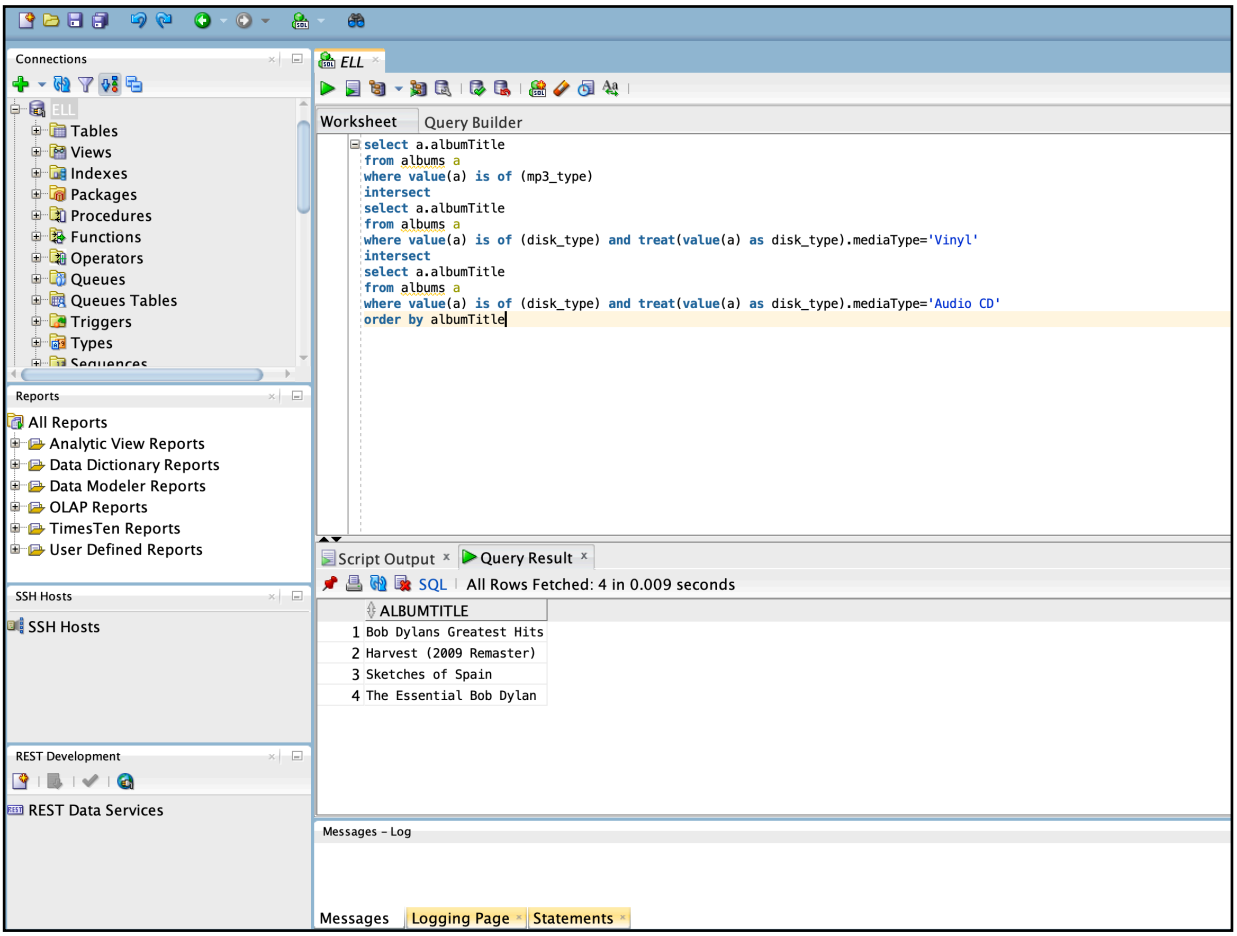


4. Are there any albums released on all media, i.e. on MP3, audio CD and vinyl? Show album title and order by album title.

Query:

```
select a.albumTitle
from albums a
where value(a) is of (mp3_type)
intersect
select a.albumTitle
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Vinyl'
intersect
select a.albumTitle
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Audio CD'
order by albumTitle
```

Result:



5. Implement the method discountPrice() that returns a discounted price using the following business rule:

a. for audio CDs released more than one year ago the discount is 20%

b. for vinyl records released more than one year ago the discount is 15%

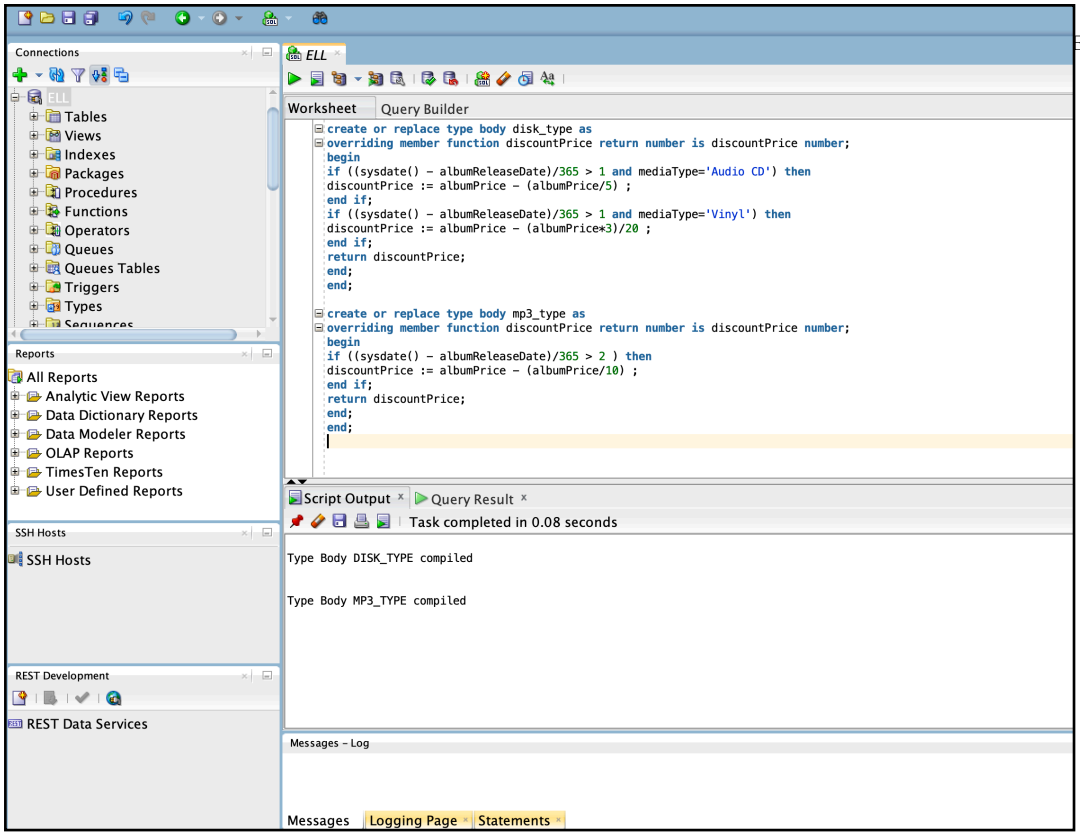
c. for MP3 downloads released more than two years ago the discount is 10%

Method:

```
create or replace type body disk_type as
overriding member function discountPrice return number is discountPrice
number;
begin
if ((sysdate() – albumReleaseDate)/365 > 1 and mediaType='Audio CD') then
discountPrice := albumPrice – (albumPrice/5) ;
end if;
if ((sysdate() – albumReleaseDate)/365 > 1 and mediaType='Vinyl') then
discountPrice := albumPrice – (albumPrice*3)/20 ;
end if;
return discountPrice;
end;
end;
--------
create or replace type body mp3_type as
overriding member function discountPrice return number is discountPrice
number;
begin
if ((sysdate() – albumReleaseDate)/365 > 2 ) then
discountPrice := albumPrice – (albumPrice/10) ;
end if;
return discountPrice;
end;
end;
```

Note: This method returns **NULL** if discount is not applied according to the given conditions

Result:



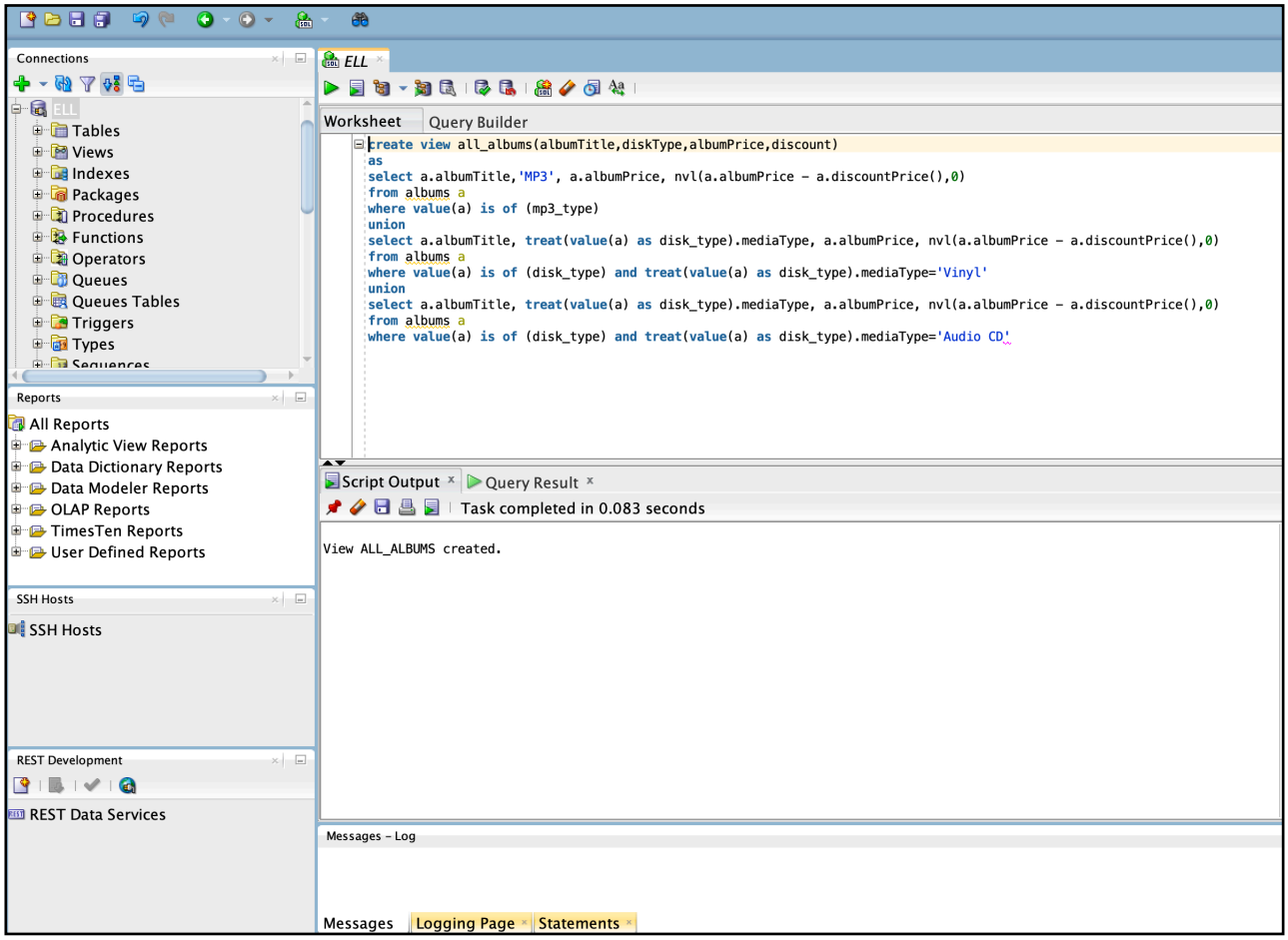
6. Create a view all_albums that includes the columns: album title, media type ('MP3', 'Vinyl', 'Audio CD'), album price, and discount (album price – discount price). Use this view to find the album that received the largest discount; show all view columns.

View:

```
create view all_albums(albumTitle,diskType,albumPrice,discount)
as
select a.albumTitle,'MP3', a.albumPrice, nvl(a.albumPrice -
a.discountPrice(),0)
from albums a
where value(a) is of (mp3_type)
union
select a.albumTitle, treat(value(a) as disk_type).mediaType, a.albumPrice,
nvl(a.albumPrice - a.discountPrice(),0)
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Vinyl'
union
select a.albumTitle, treat(value(a) as disk_type).mediaType, a.albumPrice,
nvl(a.albumPrice - a.discountPrice(),0)
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Audio CD'
```

Note: NVL function to check If discount() function return NULL and change it to 0.

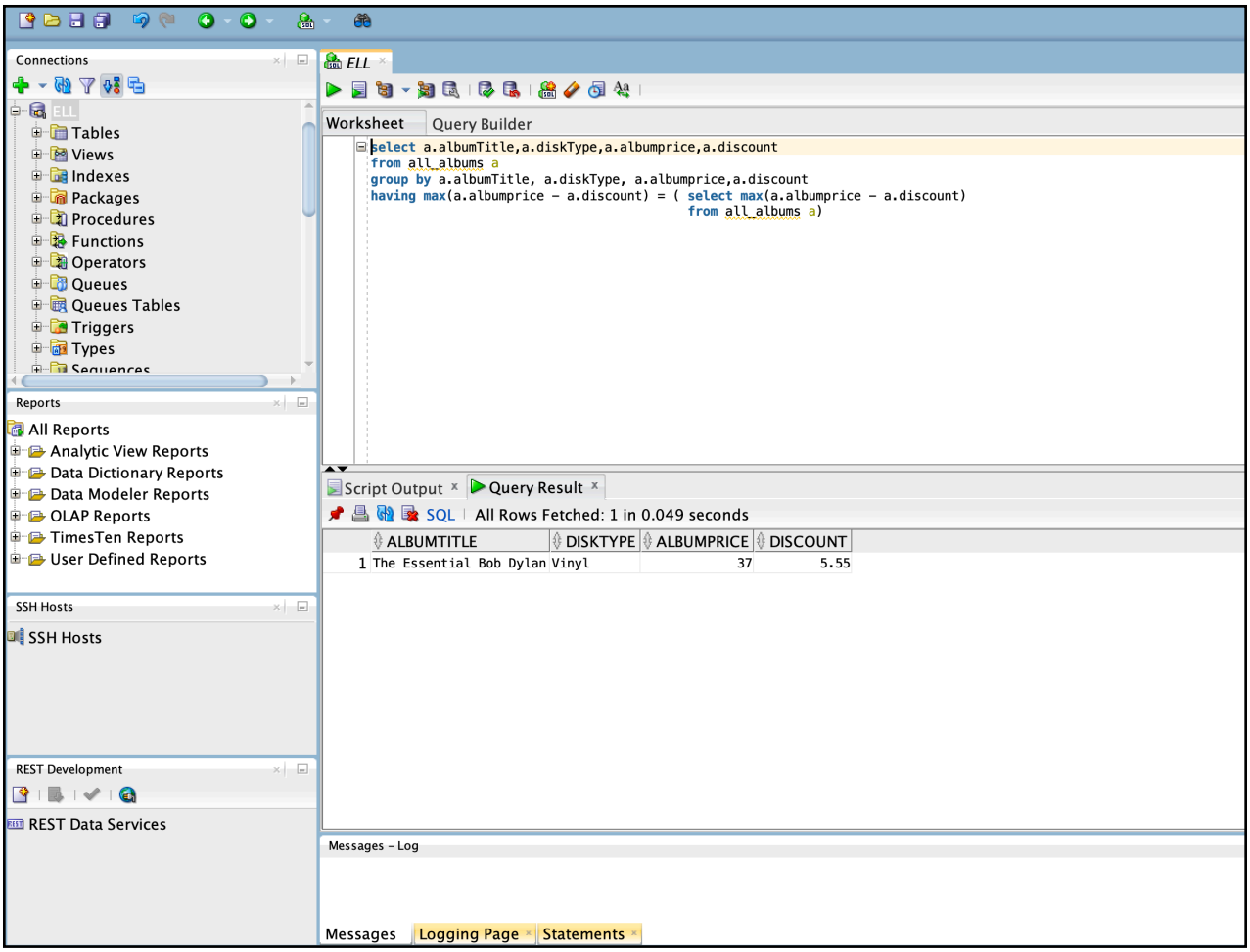
Result:



Query (Album received highest discount):

```
select a.albumTitle,a.diskType,a.albumprice,a.discount
from all_albums a
group by a.albumTitle, a.diskType, a.albumprice,a.discount
having max(a.albumprice - a.discount) = ( select max(a.albumprice -
a.discount)
                                          from all_albums a)
```

Result:



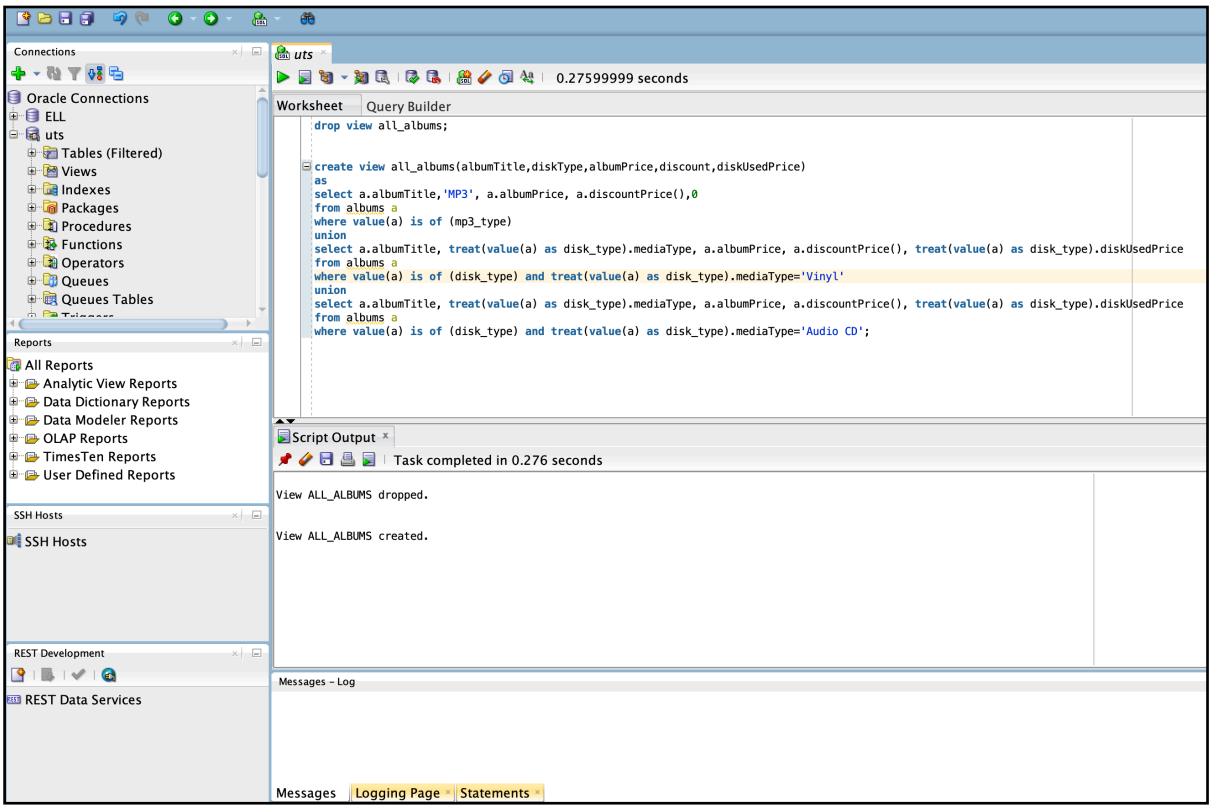
7. Now, modify the view all_albums to also include the column album used price for disks; set album used price to zero for MP3 albums. Use this view to find the most expensive used album; show all view columns.

View:

```
drop view all_albums;
```

```
create view
all_albums(albumTitle,diskType,albumPrice,discount,diskUsedPrice)
as
select a.albumTitle,'MP3', a.albumPrice, a.discountPrice(),0
from albums a
where value(a) is of (mp3_type)
union
select a.albumTitle, treat(value(a) as disk_type).mediaType, a.albumPrice,
a.discountPrice(), treat(value(a) as disk_type).diskUsedPrice
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Vinyl'
union
select a.albumTitle, treat(value(a) as disk_type).mediaType, a.albumPrice,
a.discountPrice(), treat(value(a) as disk_type).diskUsedPrice
from albums a
where value(a) is of (disk_type) and treat(value(a) as
disk_type).mediaType='Audio CD';
```

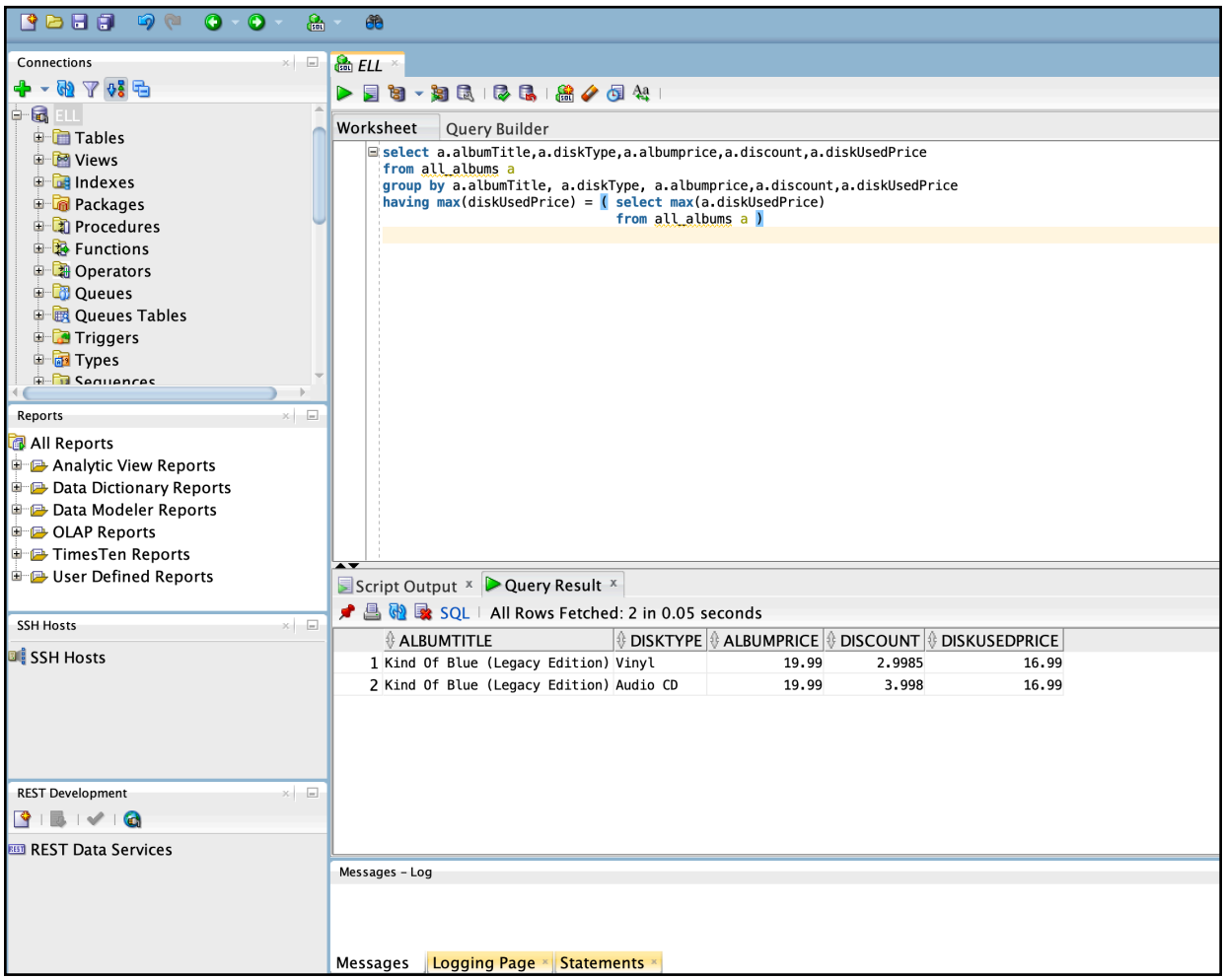
Result:



Query (Most expensive used Album):

```
select a.albumTitle,a.diskType,a.albumprice,a.discount,a.diskUsedPrice
from all_albums a
group by a.albumTitle, a.diskType, a.albumprice,a.discount,a.diskUsedPrice
having max(diskUsedPrice) = ( select max(a.diskUsedPrice)
                              from all_albums a )
```

Result:



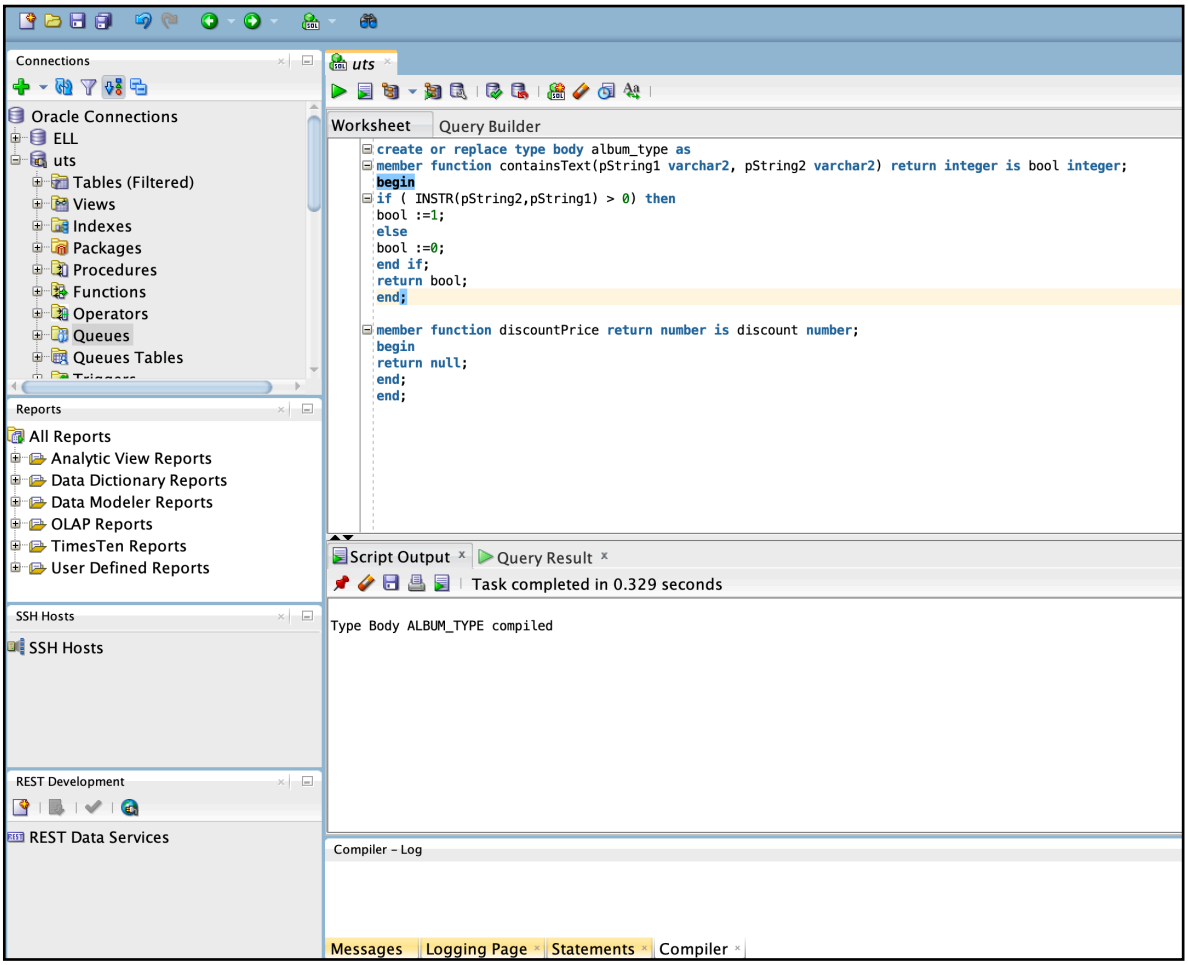
8. Implement the method containsText (pString1, pString2) that returns 1 if pString1 contains pString2, and 0 if it does not. Use this method to find albums with reviews that contain the word 'Great'. Show album title, review text, review score. Note that the signature of the containsText method is included in the original OMDB script.

Method:

```
create or replace type body album_type as
member function containsText(pString1 varchar2, pString2 varchar2) return
integer is bool integer;
begin
if ( INSTR(pString2,pString1) > 0) then
bool :=1;
else
bool :=0;
end if;
return bool;
end;

member function discountPrice return number is discount number;
begin
return null;
end;
end;
```

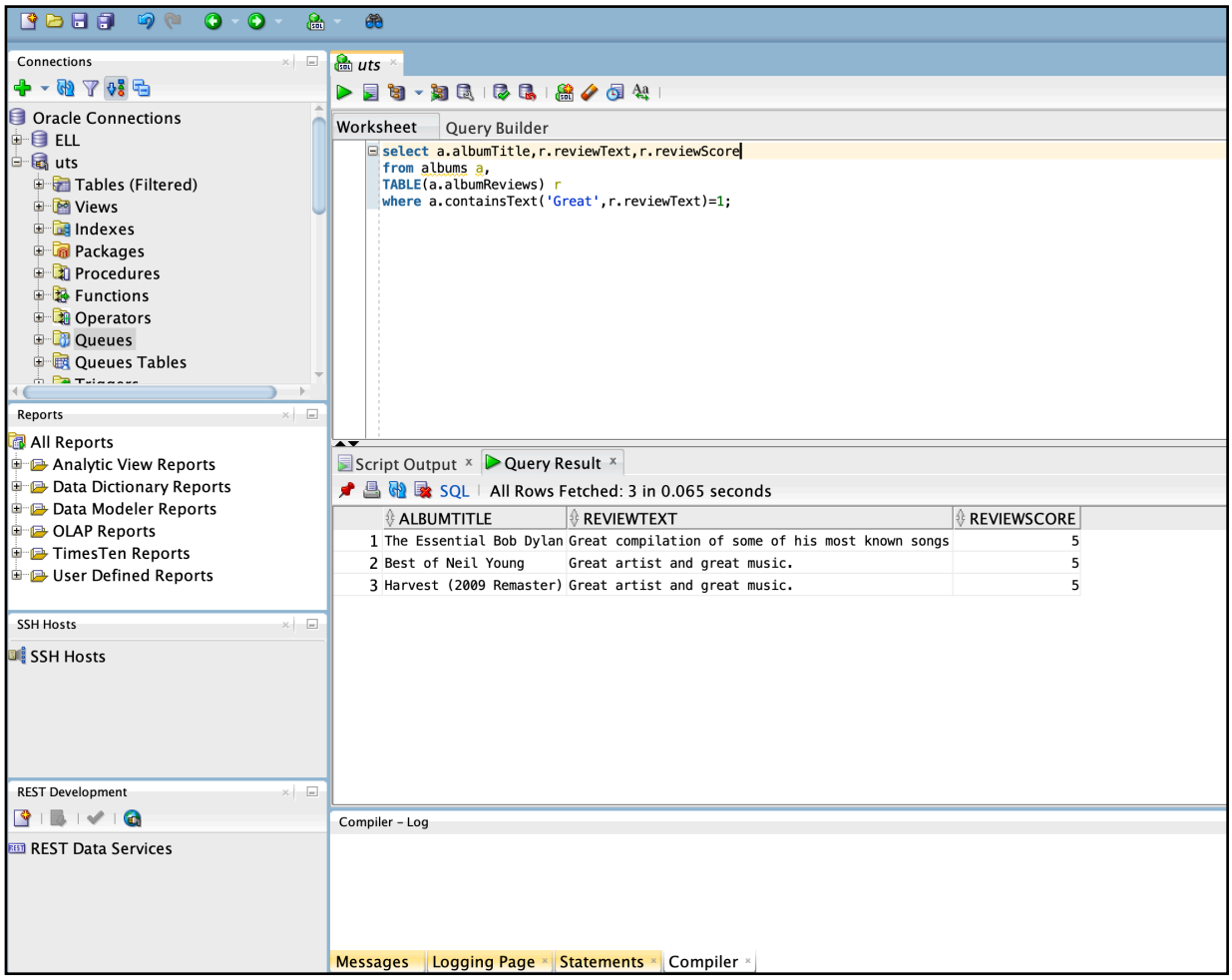
Result:



Query (Review text that contains 'Great'):

```
select a.albumTitle, r.reviewText, r.reviewScore
from albums a,
TABLE(a.albumReviews) r
where a.containsText('Great',r.reviewText)=1;
```

Result:



2.4 Discussion

There are both advantages and disadvantages of Relational design when compared to Object Relational design.

In object relational databases there are pre-defined join paths by the virtue of class design. Relational is more free-form, one can join anything they want; which is very helpful to the one who is mining data. This freedom to join anything yield better results. In our case, if one wants to add more information regarding Albums, eg Album Tracks or Album Location. One cannot do that in object relational as it is pre-defined (it can be done with relatively expensive operation, the object needs to be deleted and then re-inserted), but it is much more easier in relational design to update such as create extra columns or tables and join them.

Interpretation of object relational database is extremely difficult. For eg to access Album Reviews, one needs to create an object and access it. It is trivial only when you are working with limited dataset. If there are a large database (suppose 100 objects to deal with) with a lot of attributes, it will be difficult to access each of them. Relational databases are easier to access and do manipulation. One can easily create a new table, join and perform operations.

Whereas, there are cases when object relational database represent the real world scenario much better than relational database. For this Online Music Database(OMDB), we were able to restrict the number of artists to maximum of five using nested table. We also used VARRAY's to store Album Reviews. These things are very essential in real world databases, which cannot be implemented in relational databases.

Appendix

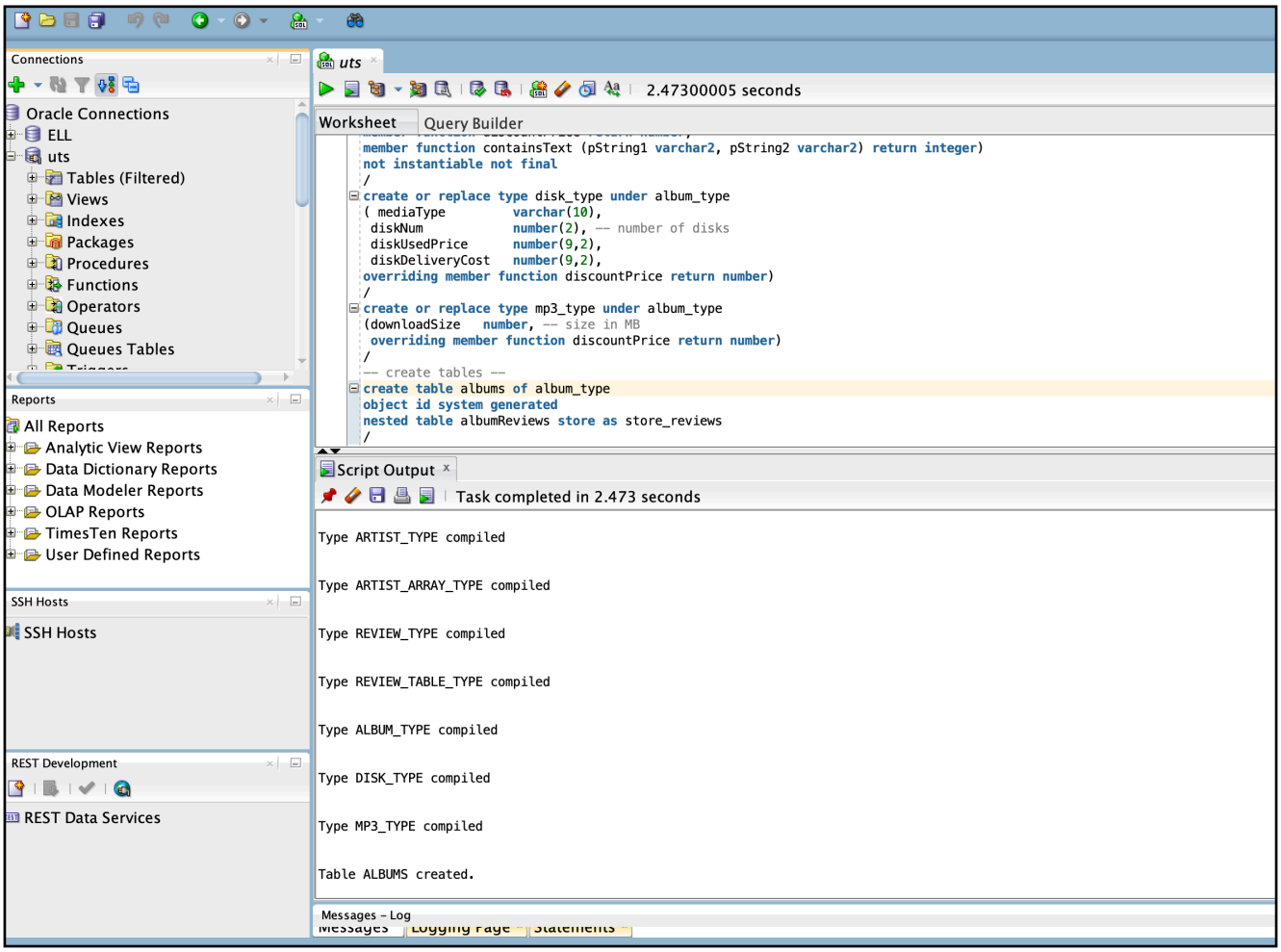
2.2 Omdb Object-Relational Design And Implementation

->The script used for creating the OMDB database (omdb.sql).

```
-- create OMDB --
-----------------
-- drop tables --
drop table albums
/
drop type disk_type
/
drop type mp3_type
/
drop type album_type
/
drop type artist_array_type
/
drop type artist_type
/
drop type review_table_type
/
drop type review_type
/
-- create types --
create or replace type artist_type as object
(artistName     varchar(50),
 artistRole     varchar(25))
/
create type artist_array_type
as varray(5) of artist_type
/
```

```
create or replace type review_type as object
(reviewerName   varchar(25),
 reviewDate     date,
 reviewText     varchar(250),
 reviewScore    number)
/
create or replace type review_table_type as table of review_type
/
create or replace type album_type as object
(albumTitle         varchar(50),
 albumPlaytime      number(3), -- minutes
 albumReleaseDate   date,
 albumGenre         varchar(15),
 albumPrice         number(9,2),
 albumTracks        number(2),
 albumArtists       artist_array_type,
 albumReviews       review_table_type,
member function discountPrice return number,
member function containsText (pString1 varchar2, pString2 varchar2) return
 integer)
not instantiable not final
/
create or replace type disk_type under album_type
( mediaType         varchar(10),
 diskNum            number(2), -- number of disks
 diskUsedPrice      number(9,2),
 diskDeliveryCost   number(9,2),
overriding member function discountPrice return number)
/
create or replace type mp3_type under album_type
(downloadSize   number, -- size in MB
 overriding member function discountPrice return number)
/
-- create tables --
create table albums of album_type
object id system generated
nested table albumReviews store as store_reviews
/
```

*Result of the script(create database) execution:



->A script of INSERT statements to populate the OMDB database using the data supplied in the omdb_data.txt file and suitable constructor statements.

```
--------INSERTION-----------------
-----------Vinyl Disk-------------------
INSERT INTO albums
values (disk_type('The Essential Bob Dylan',99,'8-
Jul-2016','Pop',37.00,32,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
),
review_table_type(
    review_type('Shawn','24-Jul-2018',' Wife loved it!',5),
    review_type('Reuben','2-
Aug-2019','Great compilation of some of his most known songs',5)
),
'Vinyl', 2, null,11.00));

INSERT INTO albums
values (disk_type('Sketches of Spain',45,'8-Mar-2011','Jazz',14.99,6,
artist_array_type(
    artist_type('Miles Davis','Composer'),
    artist_type('Miles Davis','Musician')
),
review_table_type(
    review_type('Frederick','16-
Sep-2016','Recommend listening while viewing a sunset',5),
    review_type('Juliet','12-Mar-2018','Early days of thr greates Miles--
no lover of jazz should be without this album',5)
),
'Vinyl', 1, 16.29, 7.00));

INSERT INTO albums
values (disk_type('Bob Dylans Greatest Hits',45, '31-
Jan-2017','Pop Rock',29.87,10,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
```

```
),
review_table_type(
    review_type('Kandy','16-Mar-2015','Early Dylan in all his glory.',5),
    review_type('Stewart','18-
Feb-2013','Captures Bob Dylan transformation from a folk song Composer to
a rock legend',4)
),
'Vinyl', 1, null, 11.00));

INSERT INTO albums
values (disk_type('Harvest (2009 Remaster)',44,'21-
Jun-2009','Rock Country',28.50,10,
artist_array_type(
    artist_type('Neil Young','Composer'),
    artist_type('Neil Young','Vocals')
),
review_table_type(
    review_type('John','18-Feb-2019','I absolutely LOVE this CD!.',5),
    review_type('Stewart','18-Feb-2013','Sounds good on vinyl!',5)
),
'Vinyl', 1, 14.99, 11.00));

INSERT INTO albums
values (disk_type('Kind Of Blue (Legacy Edition)',155,'20-
Jan-2009','Jazz',19.99,21,
artist_array_type(
    artist_type('Miles Davis','Composer'),
    artist_type('Miles Davis','Musician')
),
review_table_type(
    review_type('Laurence','10-Sep-2014','Very very special recording.',5)
),
'Vinyl', 3, 16.99, 10.00));

----------------Audio CDs--------------------
INSERT INTO albums
values (disk_type('Harvest (2009 Remaster)',44,'21-
Jun-2009','Rock Country',10.50, 10,
artist_array_type(
```

```
    artist_type('Neil Young','Composer'),
    artist_type('Neil Young','Vocals')
),
review_table_type(
    review_type('John','18-Feb-2019',' I absolutely LOVE this CD!',5),
    review_type('Anthony','16-Aug-2019','Neil Youngs signature album.',4)
),
'Audio CD', 1, 4.99, 11.00));

INSERT INTO albums
values (disk_type('The Essential Bob Dylan',99,'8-
Jul-2016','Pop',26.17,32,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
),
review_table_type(
    review_type('Christopher','24-
Jun-2016','This is a terrific album.',5),
    review_type('Cauley','2-
Aug-2015','There can only be one Bob Dylan. God blessed him with the gift
of verse.',5)
),
'Audio CD', 2, null, 7.00));

INSERT INTO albums
values (disk_type('Bob Dylans Greatest Hits', 50,'1-
Jun-1999','Pop Rock', 20.81, 10,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
),
review_table_type(
    review_type('Kandy','16-Mar-2015','Early Dylan in all his glory.',5),
    review_type('Stewart','18-
Feb-2013','Captures Bob Dylan transformation from a folk song Composer to
a rock legend',4)
),
'Audio CD', 1, null, 7.00));
```

```
INSERT INTO albums
values (disk_type('Kind Of Blue (Legacy Edition)',155,'20-
Jan-2009','Jazz', 19.99, 21,
artist_array_type(
    artist_type('Miles Davis','Composer'),
    artist_type('Miles Davis','Musician')
),
review_table_type(
    review_type('Amy','17-
Apr-2018','Poor quality sound compared to the vinyl record.',2)
),
'Audio CD', 3, 16.99, 10.00));

INSERT INTO albums
values (disk_type('Sketches of Spain', 45,'20-Jan-2009','Jazz', 3.11, 6,
artist_array_type(
    artist_type('Miles Davis','Composer'),
    artist_type('Miles Davis','Musician')
),
review_table_type(
    review_type('Sara','3-
Oct-2016','Another Must Have! One of Miles finest works.',5),
    review_type('Douglas','14-
Jun-2014','You might like it, but I admit it seems like a difficult listen
.',5)
),
'Audio CD', 1, 6.41, 7.00));

INSERT INTO albums
values (disk_type('Gustav Mahler Symphony No. 9',45,'12-
Oct-2017','Classical', 23.10, 5,
artist_array_type(
    artist_type('David Zinman','Conductor'),
    artist_type('Gustav Mahler','Composer'),
    artist_type('Tonhalle Orchestra','Orchestra')
),
review_table_type(
    review_type('Lindon','3-
Dec-2010','This is an uneventful but fine recording.',3),
```

```
    review_type('Prescott','24-
Aug-2013','This is truly a spellbinding record.',5)
),
'Audio CD', 1, 15.20, 7.00));

-------------------------MP3---------------------

INSERT INTO albums
values (mp3_type('Bob Dylans Greatest Hits',55, '1-
Jan-2019','Pop Rock', 5.98, 10,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
),
review_table_type(
    review_type('Mandy','16-Mar-2019','Fantastic music!',5)
),
60));

INSERT INTO albums
values (mp3_type('Best of Neil Young', 153,'21-
Feb-2019','Pop Rock', 17.50, 35,
artist_array_type(
    artist_type('Neil Young','Composer'),
    artist_type('Neil Young','Vocals')
),
review_table_type(
    review_type('John','16-Apr-2019','Great artist and great music.',5)
),
165));

INSERT INTO albums
values (mp3_type('Harvest (2009 Remaster)', 44,'21-
Jun-2009','Rock Country', 9.49, 10,
artist_array_type(
    artist_type('Neil Young','Composer'),
    artist_type('Neil Young','Vocals')
),
review_table_type(
```

```
    review_type('John','16-Apr-2019','Great artist and great music.',5)
),
52));

INSERT INTO albums
values (mp3_type('Sketches of Spain', 45,'16-Aug-2013','Jazz', 24.99, 6,
artist_array_type(
    artist_type('Miles Davis','Composer'),
    artist_type('Miles Davis','Musician')
),
review_table_type(
    review_type('Douglas','14-
Jun-2014','You might like it but I admit it seems like a difficult listen.
',5)
),
51));

INSERT INTO albums
values (mp3_type('B.B. King Greatest Hits', 114,'16-
Jul-2013','Rock Blues', 11.49, 24,
artist_array_type(
    artist_type('B.B. King','Vocals'),
    artist_type('B.B. King','Guitar')
),
review_table_type(
    review_type('David','18-
May-2015','I highly recommend this album to anyone who want to see what BB
 King is all about.',4)
),
125));

INSERT INTO albums
values (mp3_type('The Essential Bob Dylan', 99,'8-
Jul-2016','Pop', 16.00, 32,
artist_array_type(
    artist_type('Bob Dylan','Composer'),
    artist_type('Bob Dylan','Vocals')
),
review_table_type(
```

```
    review_type('Christopher','24-
Jun-2016','This is a terrific album.',5),
    review_type('Cauley','2-
Apr-2015','There can only be one Bob Dylan. God blessed him with the gift
of verse',5)
),
112));

INSERT INTO albums
values (mp3_type('Other Peoples Lives', 42,'15-
Feb-2019','Rock Dance', 9.49, 10,
artist_array_type(
    artist_type('Stats','Composer'),
    artist_type('Stats','Vocals')
),
review_table_type(
    review_type('George','17-Sep-2019','Good dancing music.',3)
),
45));
```

***Results of the script(insert) execution:**