UDACITY

‹ Return to "Data Foundations" in the classroom

Music SQL Database

REVIEW

CODE REVIEW 3

HISTORY

▾ SQL_Project_Udacity_BAND/sql-project-queries.txt 3

```
/* 1. Query for most popular genre in the market*/

SELECT count(*) as NumOfPurchases, g.Name
from Genre g JOIN Track t on g.GenreId=t.GenreId
JOIN InvoiceLine l on t.TrackId= l.TrackId GROUP by g.GenreId ORDER by NumOfPurchases
```

AWESOME

Good use of the aliases in all queries. Congratulations

```


/* 2. Query for number of songs in each playlist*/

SELECT p.Name, COUNT(pt.TrackId) AS playlist_songs_count
FROM Playlist p JOIN PlaylistTrack pt ON p.PlaylistId = pt.PlaylistId
GROUP BY p.Name ORDER BY playlist_songs_count DESC


/* 3. Query for comaprison between number of tracks and earnings from thos tracks for

SELECT ar.ArtistId, ar.Name AS artist_name, SUM(il.Quantity) AS tracks_count, SUM(il.
FROM Artist ar JOIN Album al ON ar.ArtistId = al.ArtistId
JOIN Track t ON al.AlbumId = t.AlbumId
JOIN InvoiceLine il ON t.TrackId = il.TrackId
```

AWESOME

Very well utilized aggregation clauses.

```
GROUP BY ar.ArtistId, artist_name ORDER BY earned DESC limit 10;


/* 4. Query for artists who earned the most by selling the tracks*/

SELECT ar.Name AS Artist_Name , sum(il.UnitPrice*il.Quantity) AS Song_Cost
FROM Invoice i JOIN InvoiceLine il ON i.invoiceId = il.InvoiceId
Join Customer c On i.CustomerId = c.CustomerId
JOIN Track t ON t.trackId = il.TrackId
JOIN Album al ON al.AlbumId= t.AlbumId
JOIN Artist ar ON ar.ArtistId= al.ArtistId
GROUP BY ar.Name order by Song_cost desc LIMIT 10;
```

SUGGESTION

In this query to get the intended results you do not need to chain with Invoice and Customer tables, theref
the JOIN/On clauses related to that table. Adding JOIN clauses increases the query execution time. If they a
is not good programming practice to add them. Try to analyze and apply this commentary.

RETURN TO PATH

Rate this review