COSC2307006-Database Programming

Nitish Shrestha

Student no. 239404130

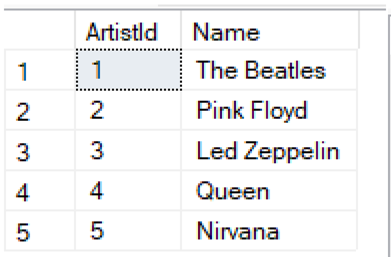
Section: 006

Assignment 3: SQL Queries

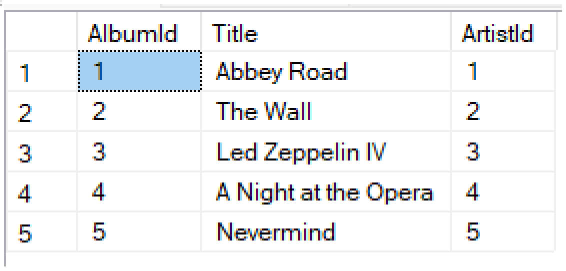
Methodology:

To do this assignment we first created a required table using create table function and supplied the required constrains using primary key and foreign key.

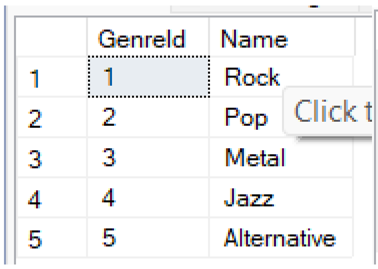
The table we used for this question is given bellow:



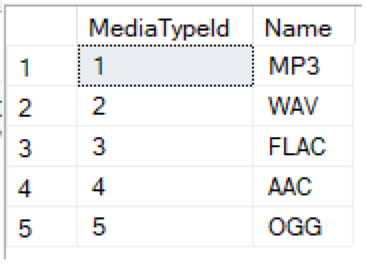
	ArtistId	Name
1	1	The Beatles
2	2	Pink Floyd
3	3	Led Zeppelin
4	4	Queen
5	5	Nirvana



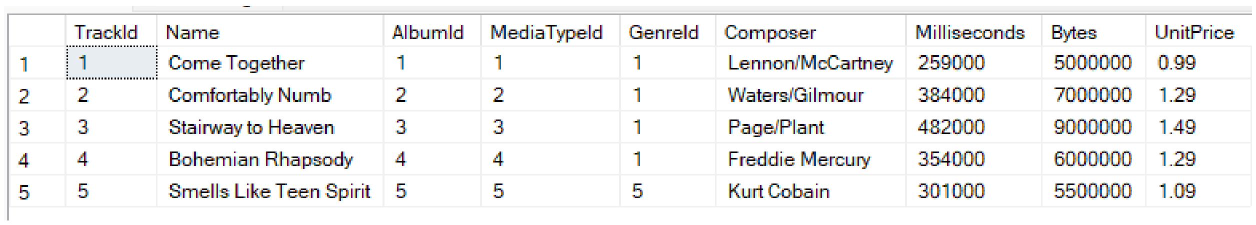
	AlbumId	Title	ArtistId
1	1	Abbey Road	1
2	2	The Wall	2
3	3	Led Zeppelin IV	3
4	4	A Night at the Opera	4
5	5	Nevermind	5



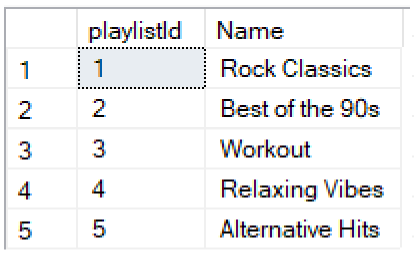
	GenreId	Name
1	1	Rock
2	2	Pop
3	3	Metal
4	4	Jazz
5	5	Alternative



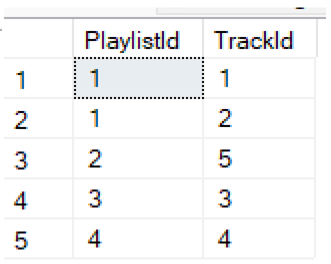
	MediaTypeId	Name
1	1	MP3
2	2	WAV
3	3	FLAC
4	4	AAC
5	5	OGG



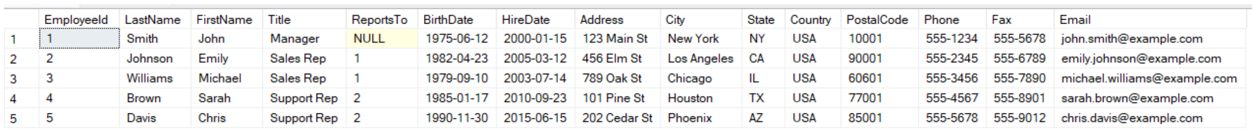
	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	1	Come Together	1	1	1	Lennon/McCartney	259000	5000000	0.99
2	2	Comfortably Numb	2	2	1	Waters/Gilmour	384000	7000000	1.29
3	3	Stairway to Heaven	3	3	1	Page/Plant	482000	9000000	1.49
4	4	Bohemian Rhapsody	4	4	1	Freddie Mercury	354000	6000000	1.29
5	5	Smells Like Teen Spirit	5	5	5	Kurt Cobain	301000	5500000	1.09



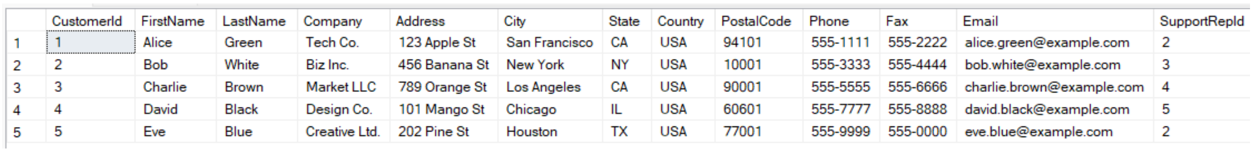
	playlistId	Name
1	1	Rock Classics
2	2	Best of the 90s
3	3	Workout
4	4	Relaxing Vibes
5	5	Alternative Hits



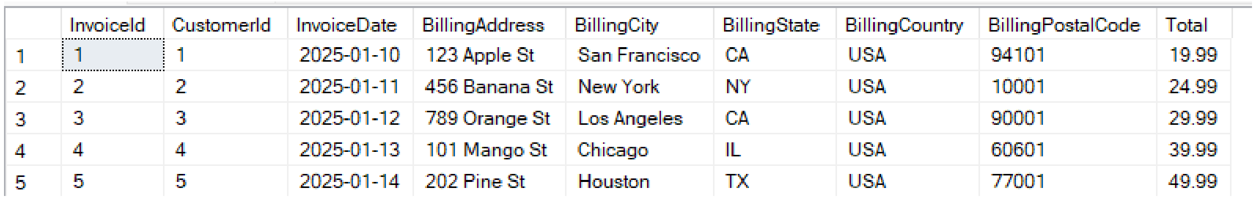
	PlaylistId	TrackId
1	1	1
2	1	2
3	2	5
4	3	3
5	4	4



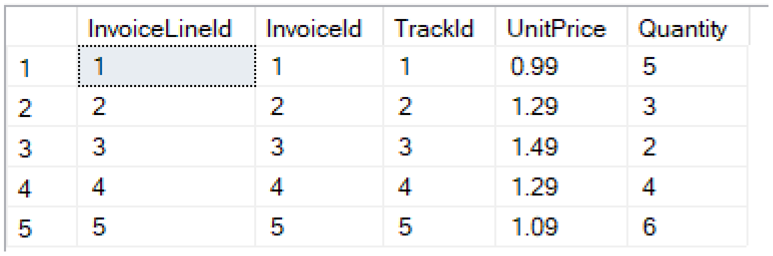
	EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	Phone	Fax	Email
1	1	Smith	John	Manager	NULL	1975-06-12	2000-01-15	123 Main St	New York	NY	USA	10001	555-1234	555-5678	john.smith@example.com
2	2	Johnson	Emily	Sales Rep	1	1982-04-23	2005-03-12	456 Elm St	Los Angeles	CA	USA	90001	555-2345	555-6789	emily.johnson@example.com
3	3	Williams	Michael	Sales Rep	1	1979-09-10	2003-07-14	789 Oak St	Chicago	IL	USA	60601	555-3456	555-7890	michael.williams@example.com
4	4	Brown	Sarah	Support Rep	2	1985-01-17	2010-09-23	101 Pine St	Houston	TX	USA	77001	555-4567	555-8901	sarah.brown@example.com
5	5	Davis	Chris	Support Rep	2	1990-11-30	2015-06-15	202 Cedar St	Phoenix	AZ	USA	85001	555-5678	555-9012	chris.davis@example.com



	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax	Email	SupportRepId
1	1	Alice	Green	Tech Co.	123 Apple St	San Francisco	CA	USA	94101	555-1111	555-2222	alice.green@example.com	2
2	2	Bob	White	Biz Inc.	456 Banana St	New York	NY	USA	10001	555-3333	555-4444	bob.white@example.com	3
3	3	Charlie	Brown	Market LLC	789 Orange St	Los Angeles	CA	USA	90001	555-5555	555-6666	charlie.brown@example.com	4
4	4	David	Black	Design Co.	101 Mango St	Chicago	IL	USA	60601	555-7777	555-8888	david.black@example.com	5
5	5	Eve	Blue	Creative Ltd.	202 Pine St	Houston	TX	USA	77001	555-9999	555-0000	eve.blue@example.com	2



	InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
1	1	1	2025-01-10	123 Apple St	San Francisco	CA	USA	94101	19.99
2	2	2	2025-01-11	456 Banana St	New York	NY	USA	10001	24.99
3	3	3	2025-01-12	789 Orange St	Los Angeles	CA	USA	90001	29.99
4	4	4	2025-01-13	101 Mango St	Chicago	IL	USA	60601	39.99
5	5	5	2025-01-14	202 Pine St	Houston	TX	USA	77001	49.99



	InvoiceLineId	InvoiceId	TrackId	UnitPrice	Quantity
1	1	1	1	0.99	5
2	2	2	2	1.29	3
3	3	3	3	1.49	2
4	4	4	4	1.29	4
5	5	5	5	1.09	6

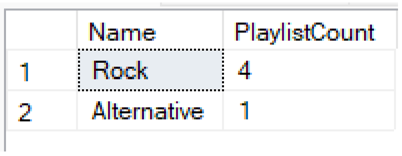
Questions and answers:

1. List the genre of tracks which is contained in the most playlist

   Query:

   ```
   SELECT Genre.Name, COUNT(PlaylistTrack.PlaylistId) AS PlaylistCount
   FROM Genre
   JOIN Track ON Genre.GenreId = Track.GenreId
   JOIN PlaylistTrack ON Track.TrackId = PlaylistTrack.TrackId
   GROUP BY Genre.Name
   ORDER BY PlaylistCount DESC;
   ```

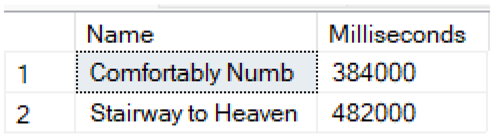
   Output:



	Name	PlaylistCount
1	Rock	4
2	Alternative	1

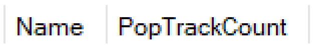
   Conclusion:

   he query uses JOIN to connect the Genre, Track, and PlaylistTrack tables. It then counts how many times each genre appears in the playlists using COUNT() and groups the results by Genre.Name with GROUP BY. Finally, ORDER BY sorts the result by the number of playlists in descending order.

2. Find audio tracks which have a length longer than the average length of all the audio tracks
   Query:
   ```
   SELECT Name, Milliseconds
   FROM Track
   WHERE Milliseconds > (
     SELECT AVG(Milliseconds) FROM Track
   );
   ```
   Output:
   

	Name	Milliseconds
1	Comfortably Numb	384000
2	Stairway to Heaven	482000

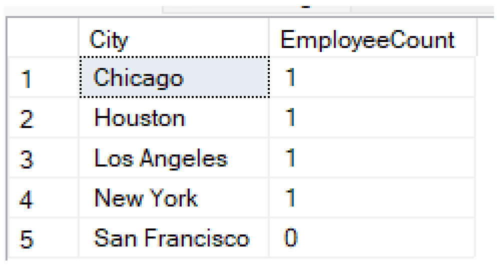
   Conclusion:
   The query selects playlists containing the most "Pop" genre tracks. It joins Playlist, PlaylistTrack, Track, and Genre tables and filters for "Pop" tracks. The COUNT() function is used to count the number of Pop tracks in each playlist, and GROUP BY groups the results by Playlist.Name.

3. Which playlist(s) contain the largest number of pop tracks
   Query:
   ```
   SELECT Playlist.Name, COUNT(*) AS PopTrackCount
   FROM Playlist
   JOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
   JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
   JOIN Genre ON Track.GenreId = Genre.GenreId
   WHERE Genre.Name = 'Pop'
   GROUP BY Playlist.Name
   ORDER BY PopTrackCount DESC;
   ```
   Output:
   

Name	PopTrackCount

4. Find the number of employees live in the same city with each customer, sorted by descending order
   Query:
   ```
   SELECT c.City, COUNT(e.EmployeeId) AS EmployeeCount
   ```

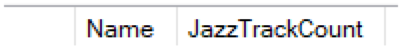
FROM Customer c
LEFT JOIN Employee e ON c.City = e.City
GROUP BY c.City
ORDER BY EmployeeCount DESC;
Output:



	City	EmployeeCount
1	Chicago	1
2	Houston	1
3	Los Angeles	1
4	New York	1
5	San Francisco	0

Conclusion:
The query counts how many employees live in the same city as each customer. A LEFT JOIN is used to include all customers even if there are no matching employees, and GROUP BY groups the results by city. The COUNT() function counts employees per city.

5. Which artist(s) has the most tracks which can be classified to Jazz
   Query:
   SELECT Artist.Name, COUNT(*) AS JazzTrackCount
   FROM Artist
   JOIN Album ON Artist.ArtistId = Album.ArtistId
   JOIN Track ON Album.AlbumId = Track.AlbumId
   JOIN Genre ON Track.GenreId = Genre.GenreId
   WHERE Genre.Name = 'Jazz'
   GROUP BY Artist.Name
   ORDER BY JazzTrackCount DESC;
   Output:

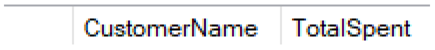


	Name	JazzTrackCount

6. Find the name of the German customer(s) who has paid the most in total without company name
   Query:
   SELECT FirstName + ' ' + LastName AS CustomerName, SUM(Invoice.Total) AS TotalSpent
   FROM Customer

```
JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE Country = 'Germany' AND Company IS NULL
GROUP BY FirstName, LastName
ORDER BY TotalSpent DESC;
```

Output:



	CustomerName	TotalSpent

Conclusion:

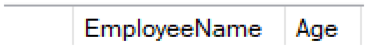
This query identifies the German customers who have spent the most without a company affiliation. It uses JOIN to combine the Customer and Invoice tables, sums the Invoice.Total for each customer, and groups by customer name. The ORDER BY sorts the customers by total spent in descending order.

7. List the name and age of the employees who support more than 5 customers (Hint: You can use GETDATE() function to get the current date, and use an other function from last assignment to calculate ages)

   Query:

```
SELECT e.FirstName + ' ' + e.LastName AS EmployeeName,
    DATEDIFF(YEAR, BirthDate, GETDATE()) AS Age
FROM Employee e
JOIN Customer c ON e.EmployeeId = c.SupportRepId
GROUP BY e.FirstName, e.LastName, BirthDate
HAVING COUNT(c.CustomerId) > 5;
```

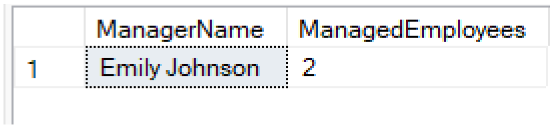
Output:



	EmployeeName	Age

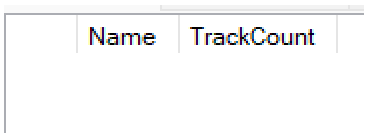
Conclusion:

The query identifies employees supporting more than five customers. It joins Employee and Customer tables, calculates the employee's age using DATEDIFF(), and groups by employee name. The HAVING clause filters the results to employees supporting more than five customers.

8. Find the manger who manages most employees but also being managed by someone else (Note: there are employees who do not have managers, i.e., there may be NULL values in ReportsTo column)
   Query:
   SELECT e1.FirstName + ' ' + e1.LastName AS ManagerName,
   COUNT(e2.EmployeeId) AS ManagedEmployees
   FROM Employee e1
   JOIN Employee e2 ON e1.EmployeeId = e2.ReportsTo
   WHERE e1.ReportsTo IS NOT NULL
   GROUP BY e1.FirstName, e1.LastName
   ORDER BY ManagedEmployees DESC;
   Output:
   

	ManagerName	ManagedEmployees
1	Emily Johnson	2

   Conclusion:
   The query finds the manager with the most direct reports, excluding employees who do not report to anyone (i.e., ReportsTo IS NOT NULL). It uses JOIN to combine the Employee table with itself and counts employees managed by each manager.

9. List the name of the artists with more than 5 tracks
   Query:
   SELECT Artist.Name, COUNT(Track.TrackId) AS TrackCount
   FROM Artist
   JOIN Album ON Artist.ArtistId = Album.ArtistId
   JOIN Track ON Album.AlbumId = Track.AlbumId
   GROUP BY Artist.Name
   HAVING COUNT(Track.TrackId) > 5;
   Output:
   

	Name	TrackCount

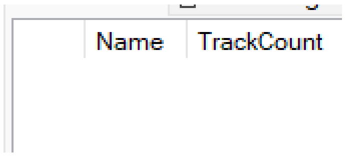
   Conclusion:
   This query lists artists who have more than five tracks. It uses JOIN to combine the Artist, Album, and Track tables and counts the number of tracks per artist using COUNT(). The HAVING clause ensures only artists with more than five tracks are returned.

10. Find the playlist(s) which contains most tracks by artist ”AC/DC”

    Query:

```
SELECT Playlist.Name, COUNT(*) AS TrackCount
FROM Playlist
JOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
JOIN Album ON Track.AlbumId = Album.AlbumId
JOIN Artist ON Album.ArtistId = Artist.ArtistId
WHERE Artist.Name = 'AC/DC'
GROUP BY Playlist.Name
ORDER BY TrackCount DESC;
```

    Output:



Name	TrackCount

    Conclusion:

    The query finds playlists containing the most tracks by the band AC/DC. It uses JOIN to combine Playlist, PlaylistTrack, Track, Album, and Artist tables and filters for tracks by AC/DC. The COUNT() function counts the number of AC/DC tracks per playlist, and the results are sorted using ORDER BY.

Thank you...