MUSIC STORE

ANALYSIS

Prepared by : Kumar Balaram Boste

Index

INTRODUCATION

Music Store Analysis project simulates a real-world database for an online/offline music store. It contains information about artists, albums, tracks, playlists, customers, employees, invoices, and sales. This project helps in analyzing customer preferences, sales performance, popular tracks, and employee management.

Overview of the Schema

The schema consists of 11 interconnected tables that provide insights into the operations of Music Store.

The Project involves multiple tables :

- ❖ Artist
- ❖ Album
- ❖ Track
- ❖ Genre
- ❖ Media Type
- ❖ Playlist
- ❖ Playlist Track
- ❖ Customer
- ❖ Employee
- ❖ Invoice
- ❖ Invoice Line

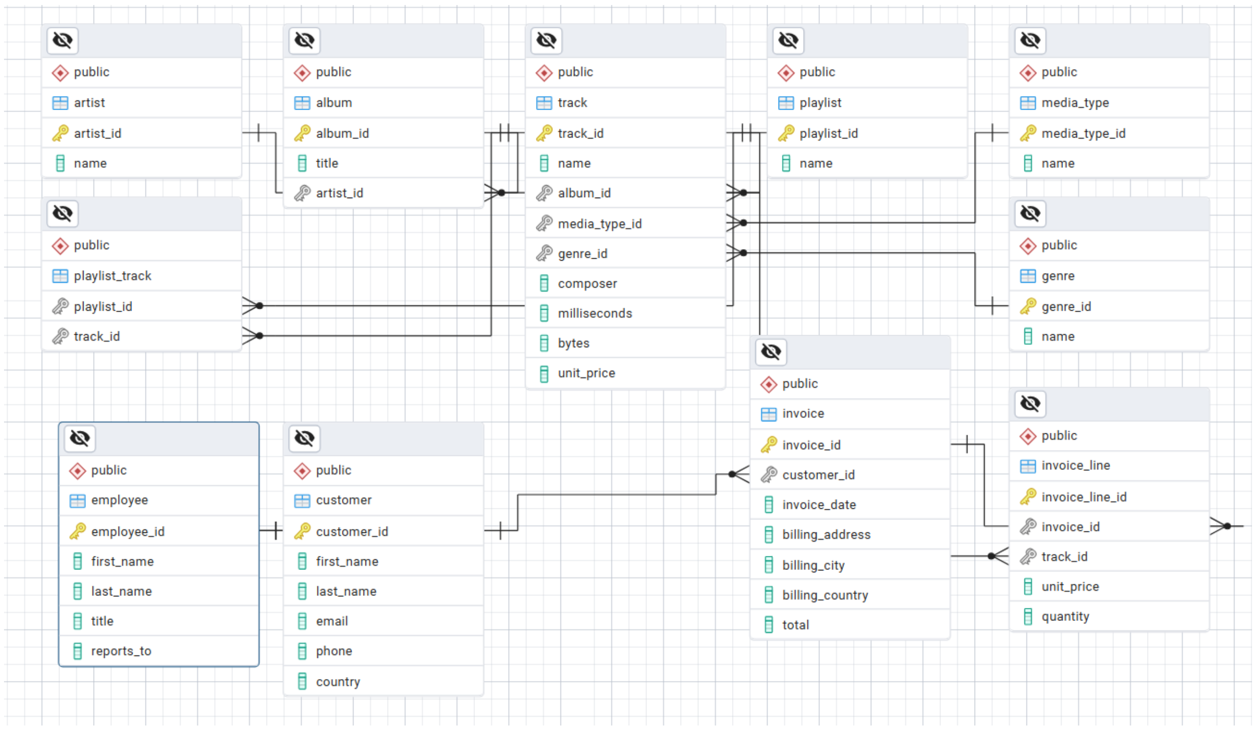
BACKGROUND OF PROJECT

The Music Industry has evolved significantly with the rise of digital platforms, streaming services, and global customer reach. Music stores—whether physical or online—must efficiently manage large volumes of data related to **artists, albums, tracks, playlists, customers, employees, and sales transactions**.

This project, **Music Store Analysis**, is designed to simulate a real-world database system for such a store. By building a relational database with **11 interconnected tables**, the project provides a structured way to store and analyze information. It allows tracking of customer purchases, employee contributions, artist performance, and genre preferences.

The analysis through SQL queries generates insights into **sales trends, customer spending behavior, popular tracks, and revenue contributions by artists and genres**. This background provides the foundation for better decision-making, targeted marketing strategies, and enhanced customer satisfaction.

Entity Relationship Diagram (ERD)



ANALYSIS - 1

1.1 List all Customers

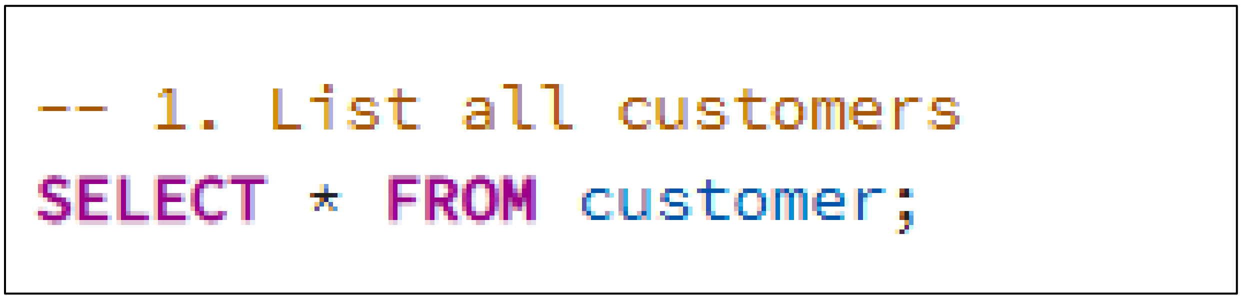
Problem Statement :

The store needs to maintain a complete list of all customers for sales tracking, communication, and marketing purposes. Having a centralized customer database helps in managing customer relationships and segmenting users for different campaigns.

Approach :

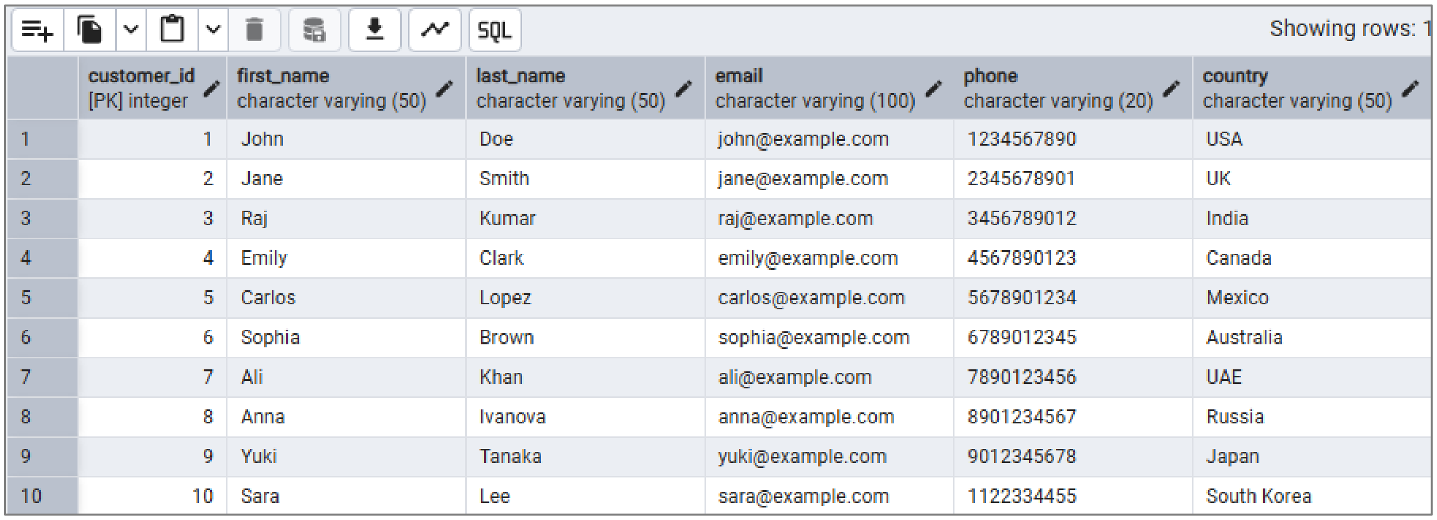
A simple SELECT * FROM customer retrieves all available customer records, including their names, addresses, contact information, and associated details.

SQL Query :



```sql
-- 1. List all customers
SELECT * FROM customer;
```

Output :



	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone character varying (20)	country character varying (50)
1	1	John	Doe	john@example.com	1234567890	USA
2	2	Jane	Smith	jane@example.com	2345678901	UK
3	3	Raj	Kumar	raj@example.com	3456789012	India
4	4	Emily	Clark	emily@example.com	4567890123	Canada
5	5	Carlos	Lopez	carlos@example.com	5678901234	Mexico
6	6	Sophia	Brown	sophia@example.com	6789012345	Australia
7	7	Ali	Khan	ali@example.com	7890123456	UAE
8	8	Anna	Ivanova	anna@example.com	8901234567	Russia
9	9	Yuki	Tanaka	yuki@example.com	9012345678	Japan
10	10	Sara	Lee	sara@example.com	1122334455	South Korea

Analysis :

This result serves as the master customer list. It helps identify the geographical spread of customers, track how many are active, and provides the foundation for advanced segmentation such as by country or spending level.

Recommendations:

- Keep this table updated regularly with new and modified customer details.
- Use customer information to design personalized offers and communication strategies.
- Integrate this with a CRM (Customer Relationship Management) system for better targeting.

1.2 Get Tracks with Price Greater than 1.20

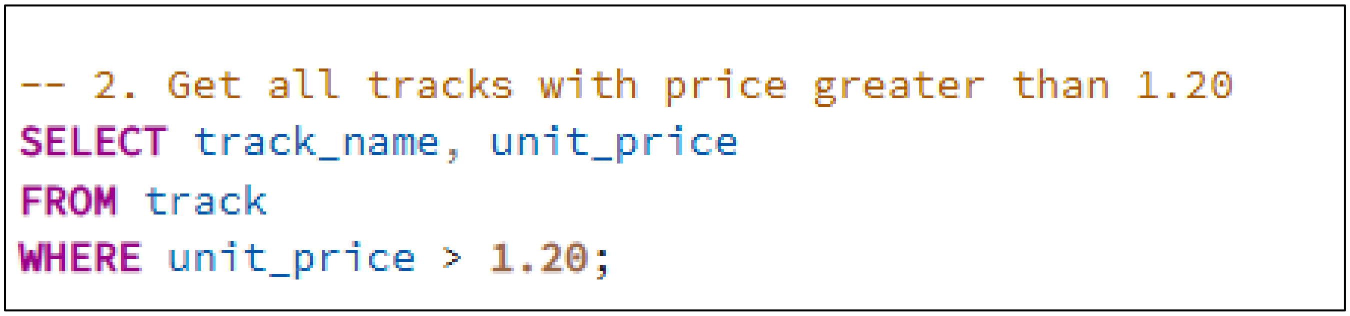
Problem Statement :

The business needs to identify premium-priced tracks to understand which songs contribute more to revenue and to evaluate their role in pricing strategy.

Approach :

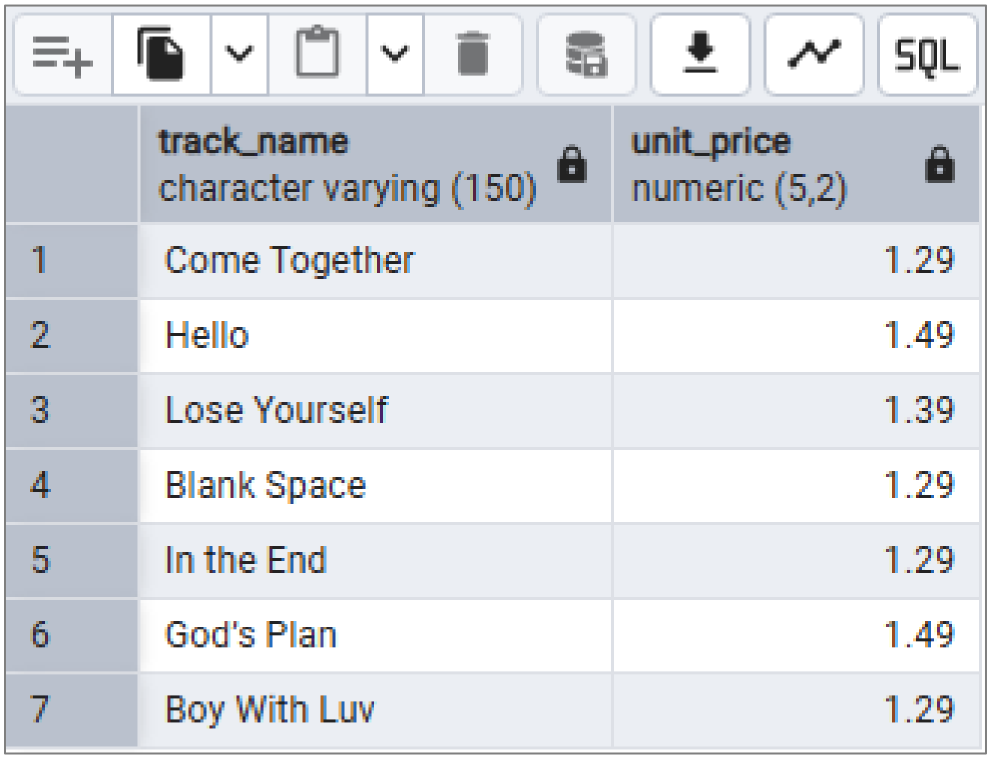
Querying the track table and filtering rows where unit price > 1.20 helps isolate tracks that are above the standard baseline price.

SQL Query :



```sql
-- 2. Get all tracks with price greater than 1.20
SELECT track_name, unit_price
FROM track
WHERE unit_price > 1.20;
```

Output :



	track_name character varying (150)	unit_price numeric (5,2)
1	Come Together	1.29
2	Hello	1.49
3	Lose Yourself	1.39
4	Blank Space	1.29
5	In the End	1.29
6	God's Plan	1.49
7	Boy With Luv	1.29

Analysis :

Premium-priced tracks often represent popular, exclusive, or high-demand songs. Customers willing to pay for these tracks demonstrate stronger brand loyalty or preference.

Recommendations :

- Promote these premium tracks more aggressively through featured playlists.
- Bundle them with standard-priced tracks for up-selling.
- Analyze whether higher prices negatively affect sales volume.

1.3 Find all Albums by Ed Sheeran

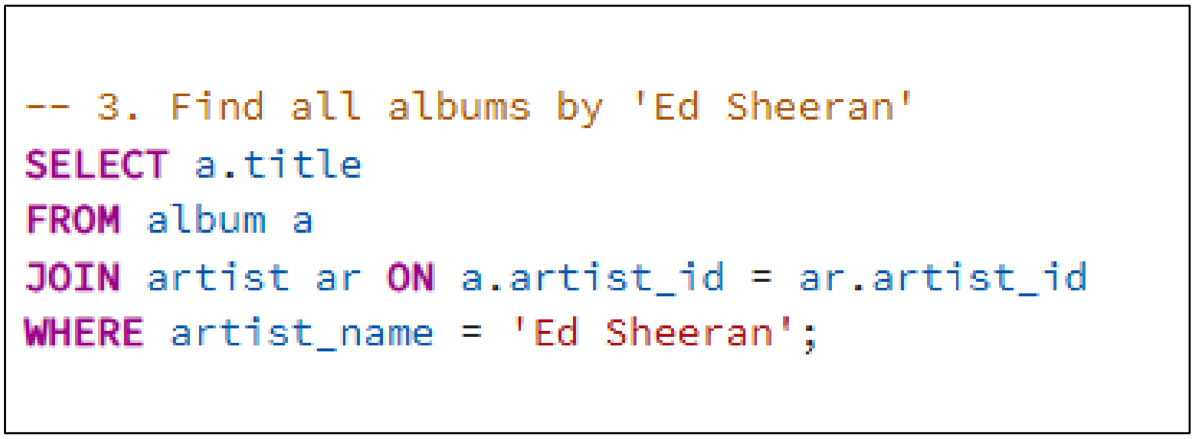
Problem Statement :

The store wants to find all albums released by a specific artist, in this case Ed Sheeran. This is useful for artist-focused campaigns and promotions.

Approach :

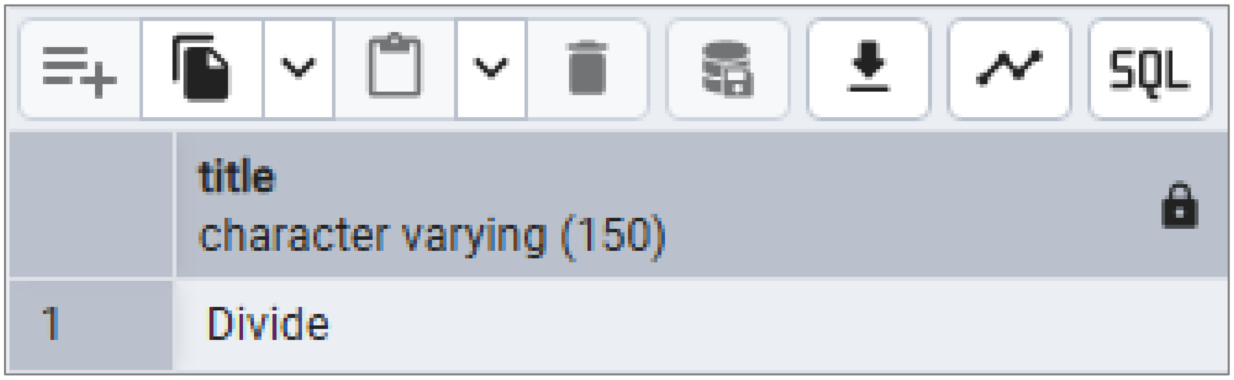
Join the album and artist tables on their artist_id and filter where artist_name = 'Ed Sheeran'.

SQL Query :



```sql
-- 3. Find all albums by 'Ed Sheeran'
SELECT a.title
FROM album a
JOIN artist ar ON a.artist_id = ar.artist_id
WHERE artist_name = 'Ed Sheeran';
```

Output :



	title character varying (150)
1	Divide

Analysis :

This identifies all albums in the catalog belonging to Ed Sheeran. Such queries help in understanding catalog coverage for specific artists.

Recommendations :

- Promote artist collections (e.g., "All Ed Sheeran Albums").
- Identify gaps in catalog completeness for top-selling artists.
- Use this data for artist-specific recommendations to customers.

1.4 Count Total Customers from India

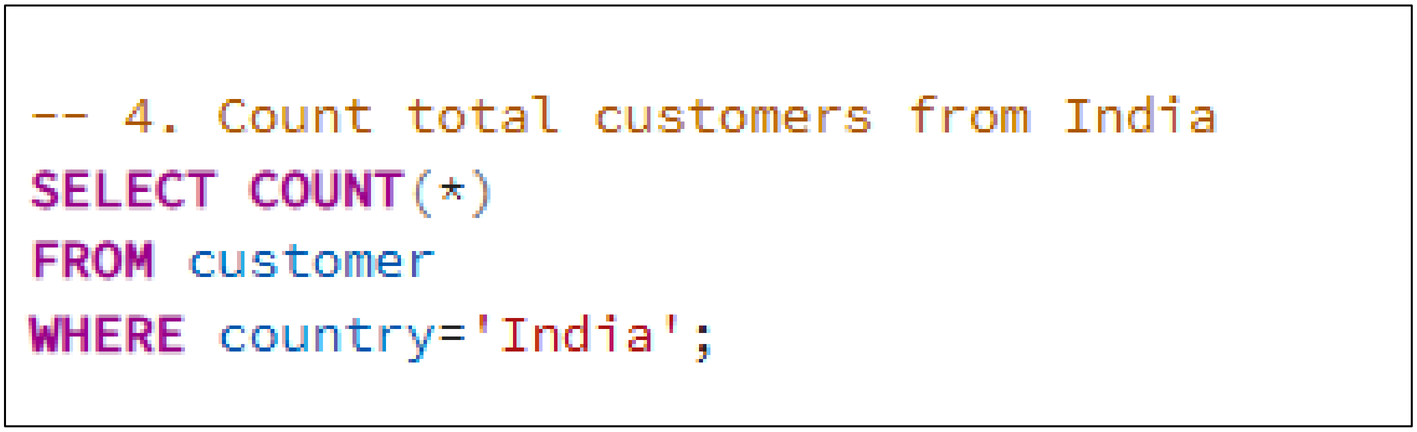
Problem Statement :

To measure regional customer distribution, the store wants to find how many customers are from India. This supports regional marketing and expansion strategies.

Approach :

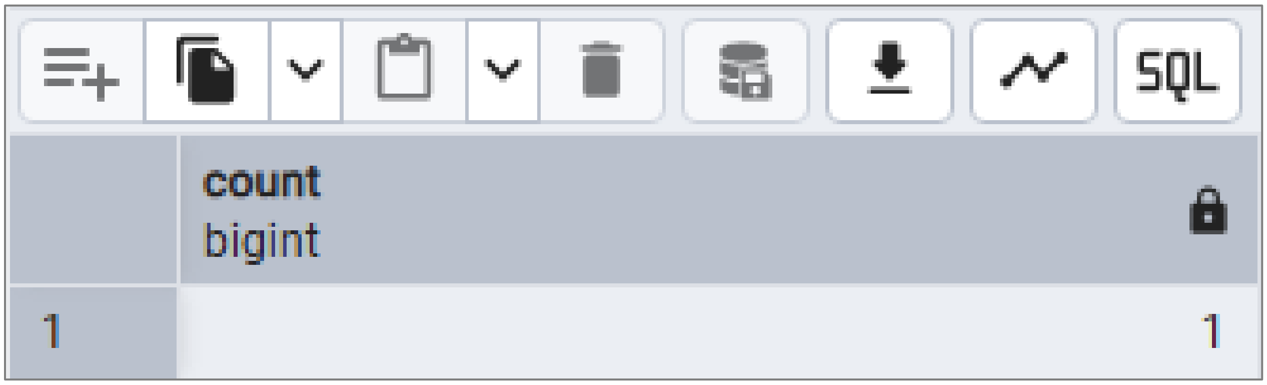
Use COUNT(*) on the customer table filtered by country = 'India'.

SQL Query :



```sql
-- 4. Count total customers from India
SELECT COUNT(*)
FROM customer
WHERE country='India';
```

Output :



	count bigint
1	1

Analysis :

The result gives a numerical measure of the Indian customer base. This metric is useful for understanding international presence.

Recommendations :

- Launch localized promotions in India.
- Introduce region-specific playlists (e.g., Bollywood, Indian classical).
- Monitor customer growth in this region over time.

1.5 Display all Employees Who are Sales Reps

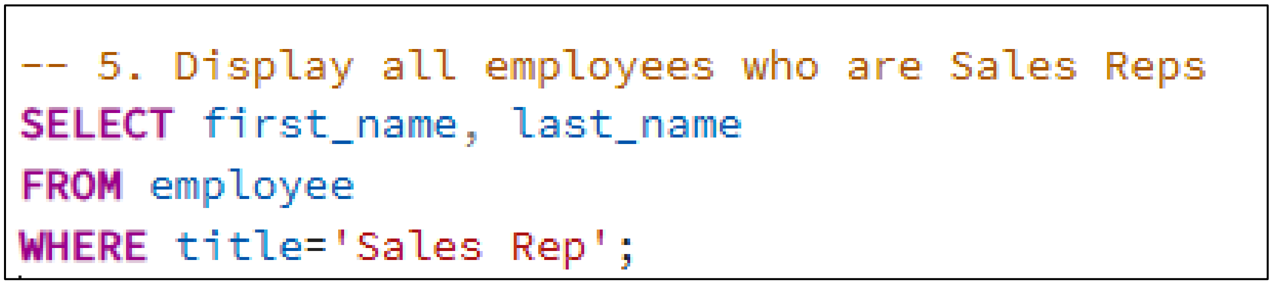
Problem Statement :

Identify employees working in direct customer-facing sales roles. This is important for performance management and workforce planning.

Approach :

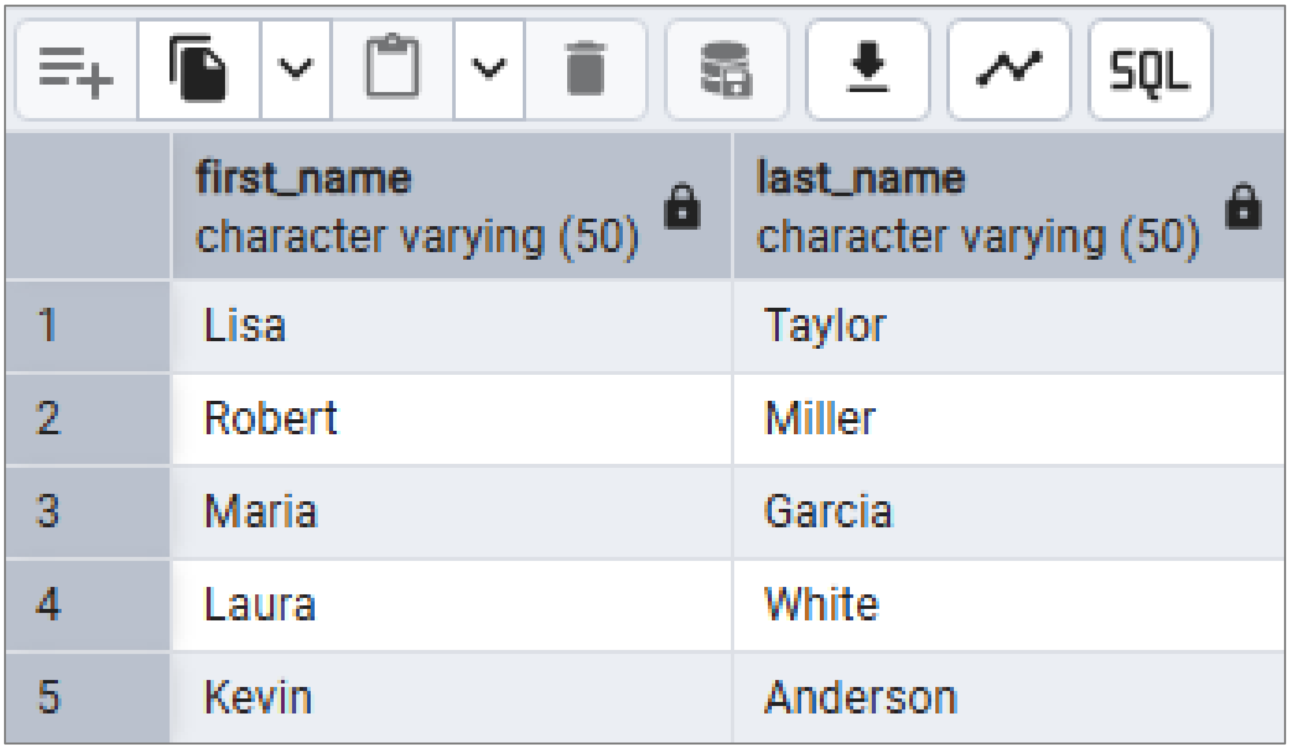
Query the employee table where title = 'Sales Rep'.

SQL Query :



```sql
-- 5. Display all employees who are Sales Reps
SELECT first_name, last_name
FROM employee
WHERE title='Sales Rep';
```

Output :



	first_name character varying (50)	last_name character varying (50)
1	Lisa	Taylor
2	Robert	Miller
3	Maria	Garcia
4	Laura	White
5	Kevin	Anderson

Analysis:

This returns the names of employees who directly handle customer accounts. These individuals are critical for driving revenue.

Recommendations:

- Provide training programs to Sales Reps to improve customer experience.
- Track their sales performance via linked invoices.
- Incentivize top-performing Sales Reps with bonuses.

ANALYSIS - 2

2.1 Find the Total Number of Tracks Per Genre

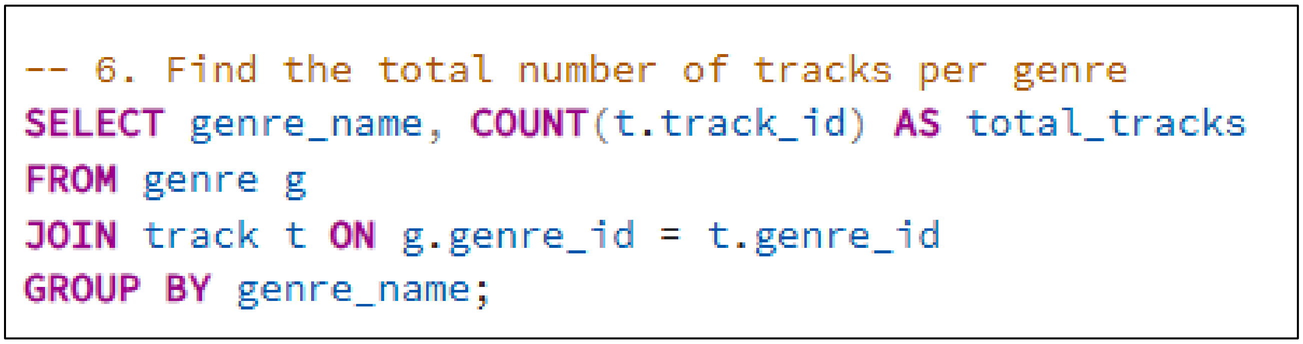
Problem Statement:

The business wants to analyze content diversity and determine how tracks are distributed across genres.

Approach:

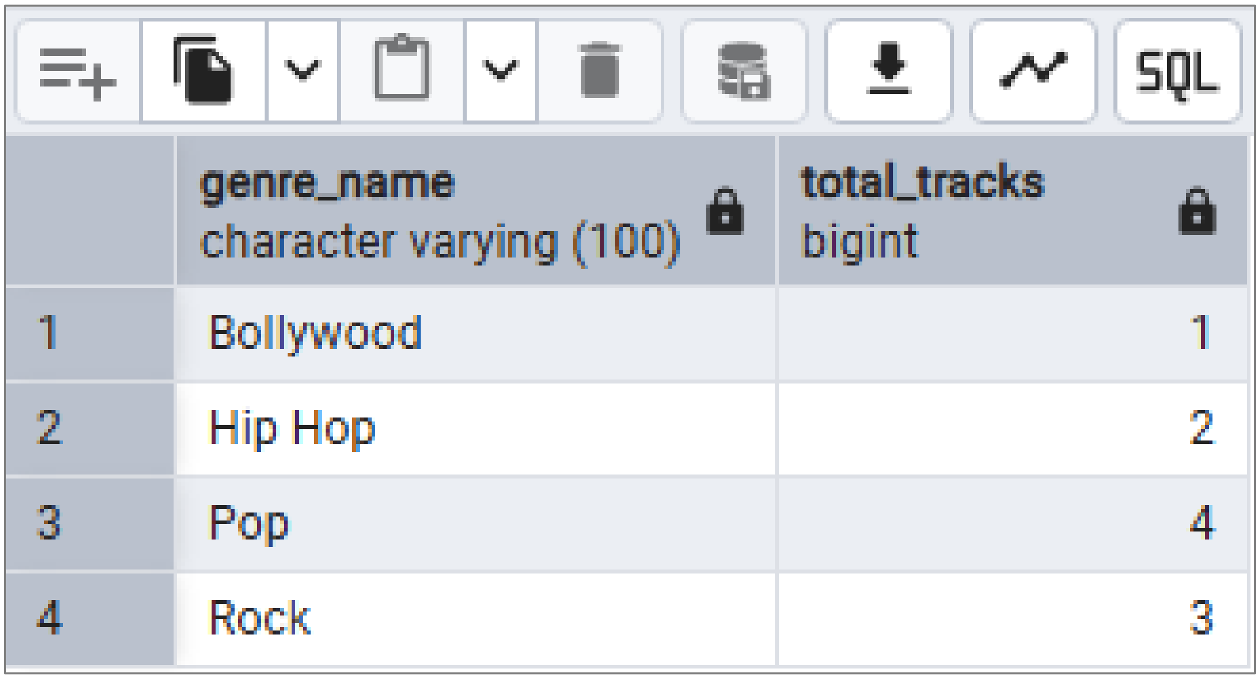
Join genre and track tables, group by genre, and count the number of tracks in each.

SQL Query :



```sql
-- 6. Find the total number of tracks per genre
SELECT genre_name, COUNT(t.track_id) AS total_tracks
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
GROUP BY genre_name;
```

Output :



	genre_name character varying (100)	total_tracks bigint
1	Bollywood	1
2	Hip Hop	2
3	Pop	4
4	Rock	3

Analysis:

This query shows whether the catalog is well-balanced or skewed toward certain genres.

Recommendations:

- Expand in genres with fewer tracks to attract a wider audience.
- Focus marketing efforts on the most populated genres.
- Balance catalog diversity to cater to niche markets.

2.2 Show Invoice Details along with Customer Names

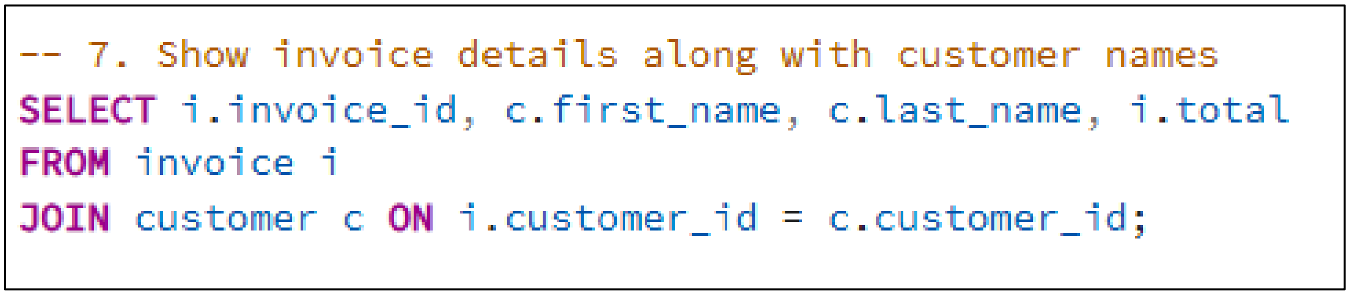
Problem Statement:

Management wants a detailed view of invoices along with the customers who generated them for better financial tracking.

Approach:

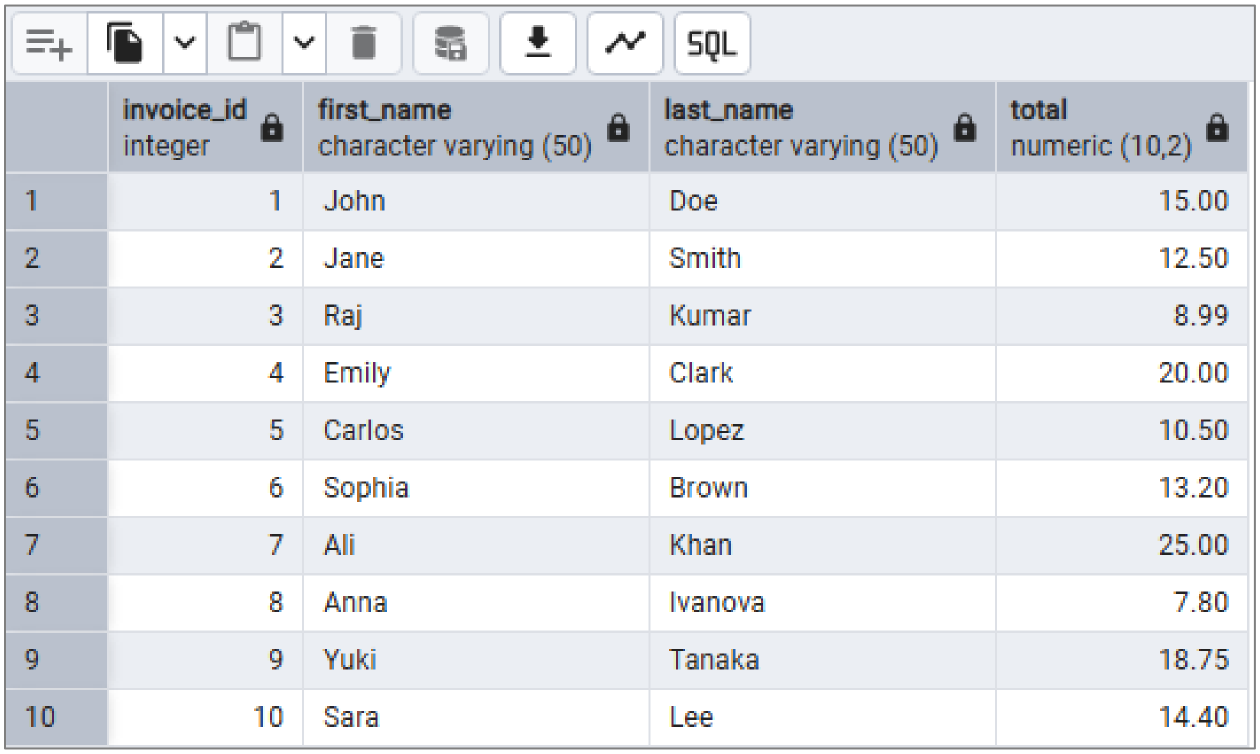
Join invoice with customer to display invoice ID, customer name, and invoice total.

SQL Query :



```sql
-- 7. Show invoice details along with customer names
SELECT i.invoice_id, c.first_name, c.last_name, i.total
FROM invoice i
JOIN customer c ON i.customer_id = c.customer_id;
```

Output :



	invoice_id integer	first_name character varying (50)	last_name character varying (50)	total numeric (10,2)
1	1	John	Doe	15.00
2	2	Jane	Smith	12.50
3	3	Raj	Kumar	8.99
4	4	Emily	Clark	20.00
5	5	Carlos	Lopez	10.50
6	6	Sophia	Brown	13.20
7	7	Ali	Khan	25.00
8	8	Anna	Ivanova	7.80
9	9	Yuki	Tanaka	18.75
10	10	Sara	Lee	14.40

Analysis:

The output helps connect each transaction to a customer, making it easier to analyze spending behavior and purchase history.

Recommendations:

- Use this for auditing and reconciliation.
- Implement customer segmentation based on spending patterns.
- Provide customers with invoice history for transparency.

2.3 Find Top 5 Most Expensive Tracks

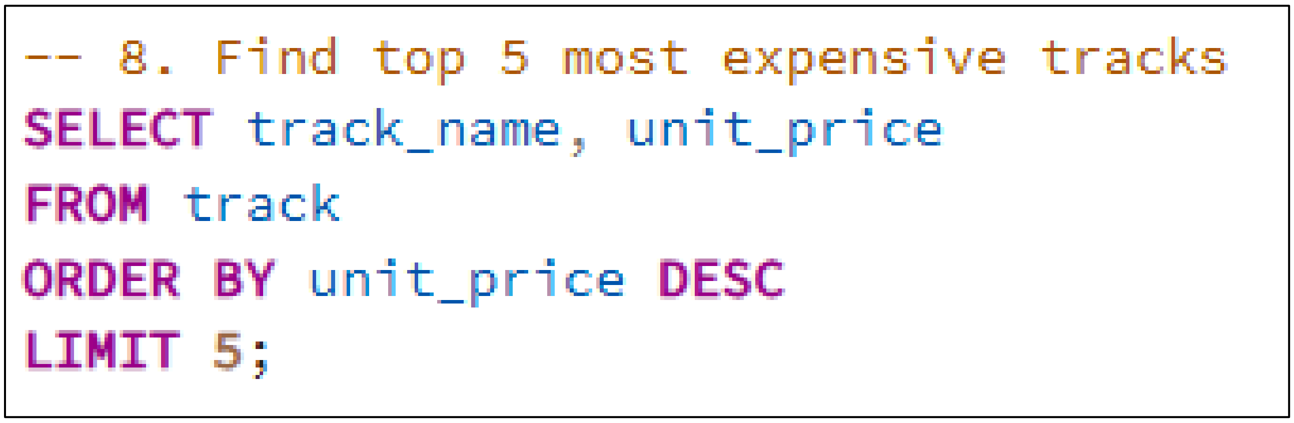
Problem Statement:

Identify tracks with the highest unit prices to assess catalog pricing.

Approach:

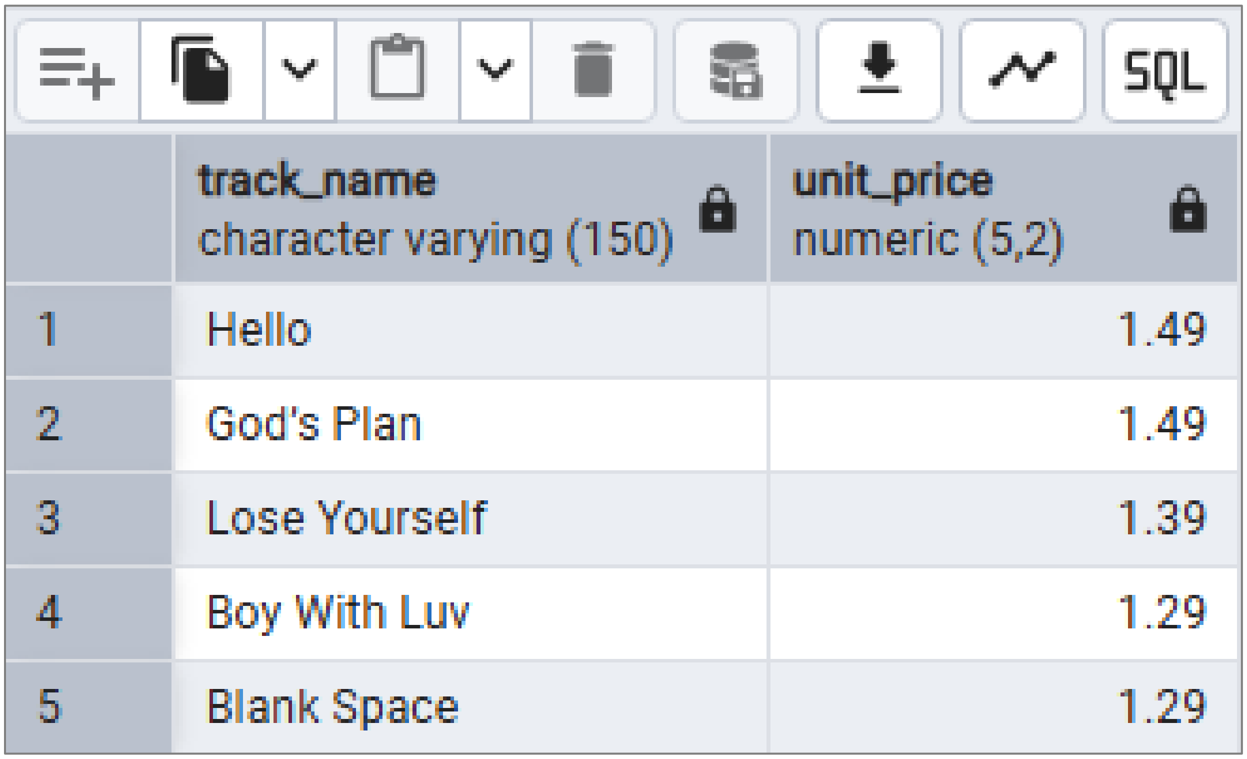
Order track records by unit_price descending and select the top 5.

SQL Query :



```sql
-- 8. Find top 5 most expensive tracks
SELECT track_name, unit_price
FROM track
ORDER BY unit_price DESC
LIMIT 5;
```

Output :



	track_name character varying (150)	unit_price numeric (5,2)
1	Hello	1.49
2	God's Plan	1.49
3	Lose Yourself	1.39
4	Boy With Luv	1.29
5	Blank Space	1.29

Analysis:

These are the most premium-priced tracks, potentially influencing overall revenue.

Recommendations:

- Highlight these tracks in "Premium Collections."
- Study whether higher prices correlate with reduced sales volume.
- Consider limited discounts to increase demand.

2.4 Show all Playlists with Track Count

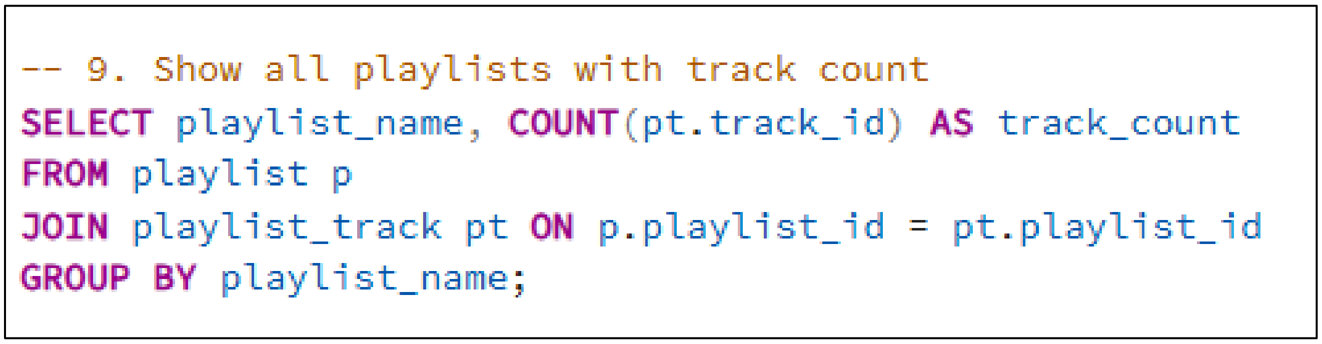
Problem Statement:

Determine the size of playlists to understand catalog curation.

Approach:

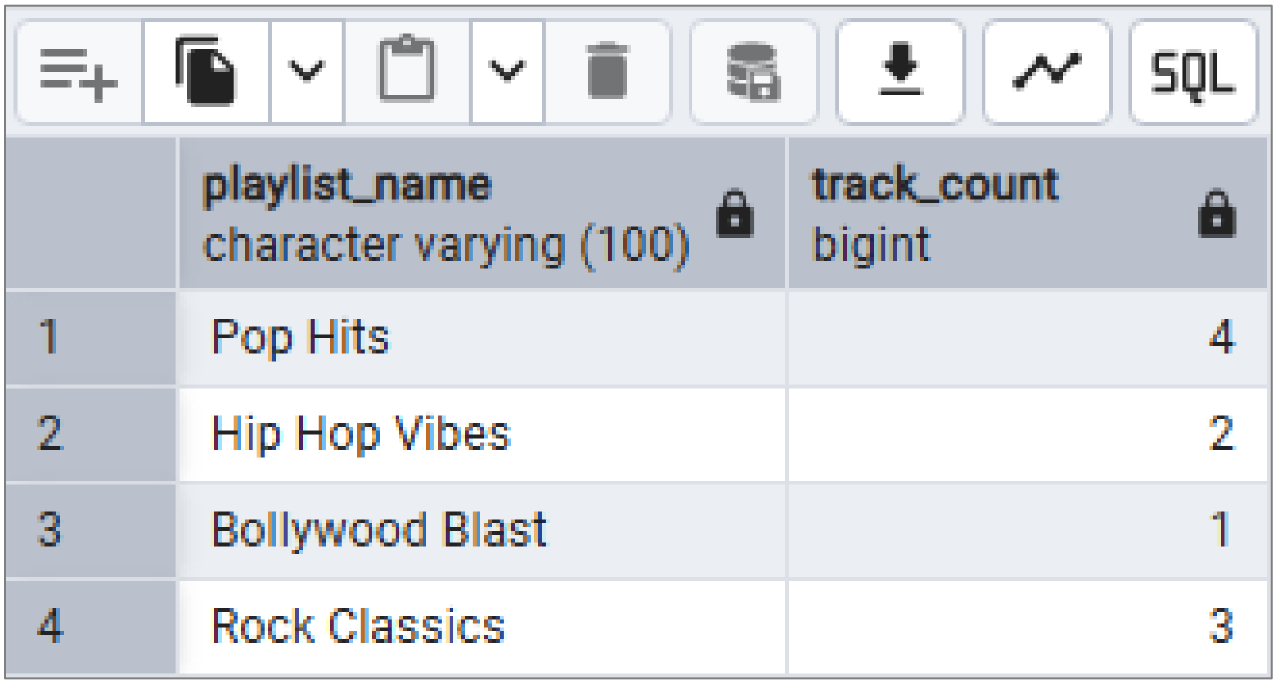
Join playlist with playlist_track, group by playlist, count tracks per playlist.

SQL Query :



```
-- 9. Show all playlists with track count
SELECT playlist_name, COUNT(pt.track_id) AS track_count
FROM playlist p
JOIN playlist_track pt ON p.playlist_id = pt.playlist_id
GROUP BY playlist_name;
```

Output :



	playlist_name character varying (100)	track_count bigint
1	Pop Hits	4
2	Hip Hop Vibes	2
3	Bollywood Blast	1
4	Rock Classics	3

Analysis:

Reveals which playlists are content-rich and which are sparse. This is important for customer satisfaction.

Recommendations:

- Standardize playlist sizes to ensure consistency.
- Promote playlists with balanced track counts (not too short, not too long).
- Curate niche playlists for specialized audiences.

2.5 Find Customers Who Spent more than 15

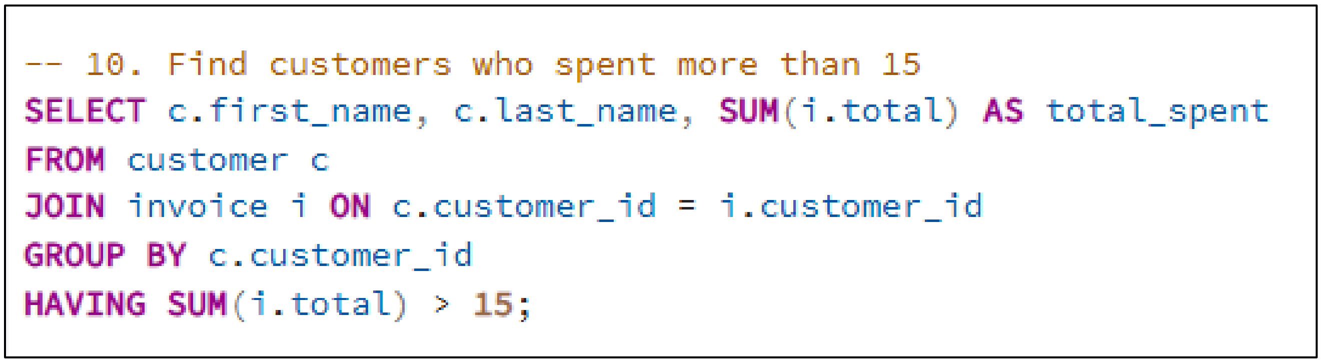
Problem Statement:

Identify high-value customers contributing more than $15 in purchases.

Approach:

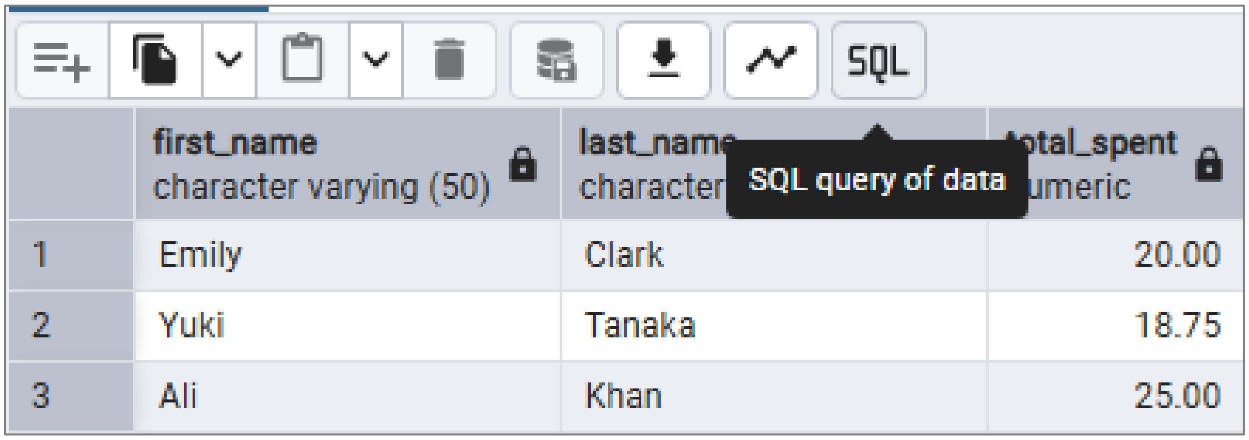
Join customer and invoice, group by customer, filter with SUM(total) > 15.

SQL Query :



```sql
-- 10. Find customers who spent more than 15
SELECT c.first_name, c.last_name, SUM(i.total) AS total_spent
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id
HAVING SUM(i.total) > 15;
```

Output :



	first_name character varying (50)	last_name character	total_spent numeric
1	Emily	Clark	20.00
2	Yuki	Tanaka	18.75
3	Ali	Khan	25.00

Analysis:

This highlights big spenders who are valuable to retain.

Recommendations:

- Enroll these customers in loyalty programs.
- Offer personalized recommendations and discounts.
- Send them early access to new releases.

2.6 Find the Best-Selling Track

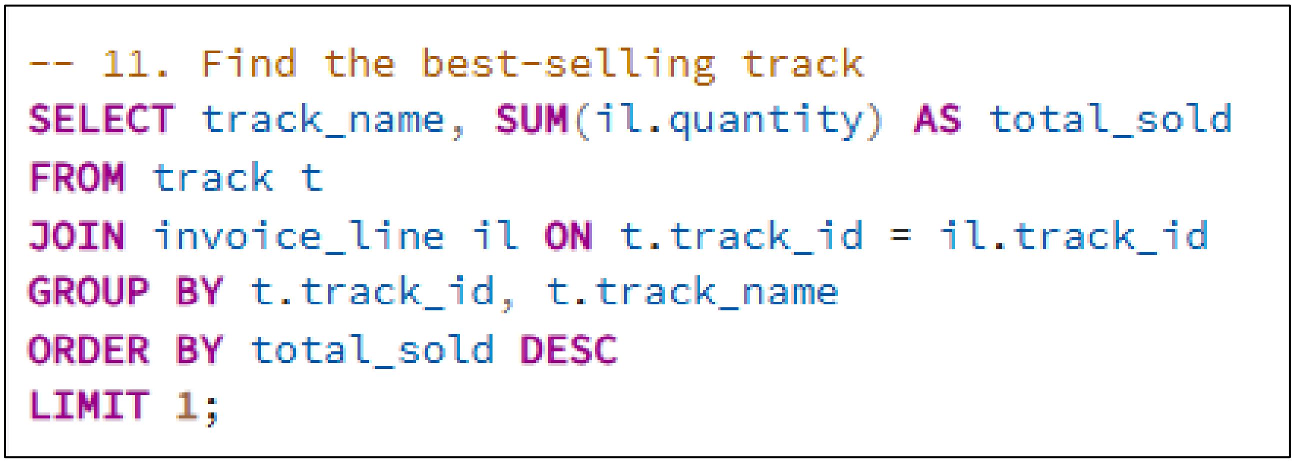
Problem Statement:

Find the track that has sold the most copies.

Approach:

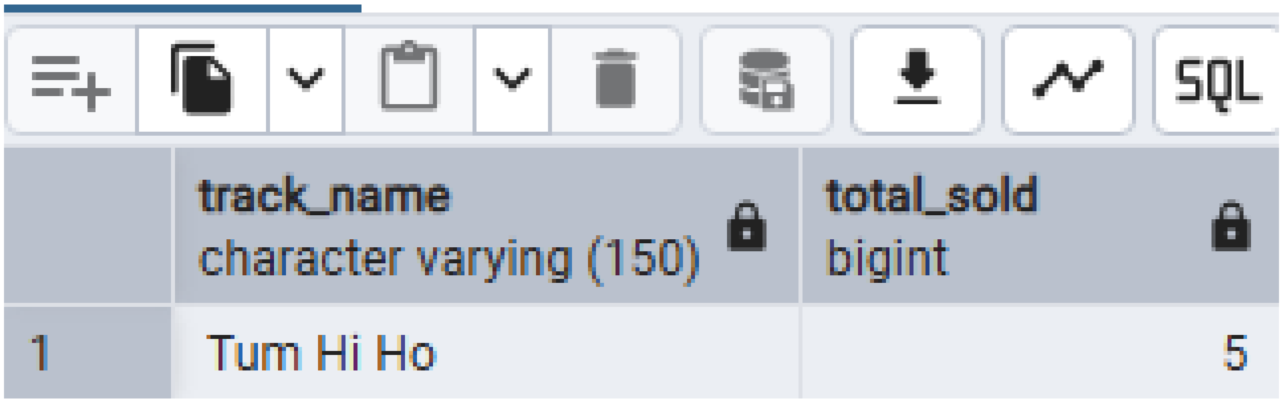
Join track with invoice_line, group by track, sum quantities, order descending.

SQL Query :



```
-- 11. Find the best-selling track
SELECT track_name, SUM(il.quantity) AS total_sold
FROM track t
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY t.track_id, t.track_name
ORDER BY total_sold DESC
LIMIT 1;
```

Output :



	track_name character varying (150)	total_sold bigint
1	Tum Hi Ho	5

Analysis:

The result shows the most popular track by sales volume.

Recommendations:

- Promote this track on the homepage or featured section.
- Create playlists around similar tracks.
- Use it as a model for predicting future hits

2.7 Find top 3 Artists by Sales Revenue

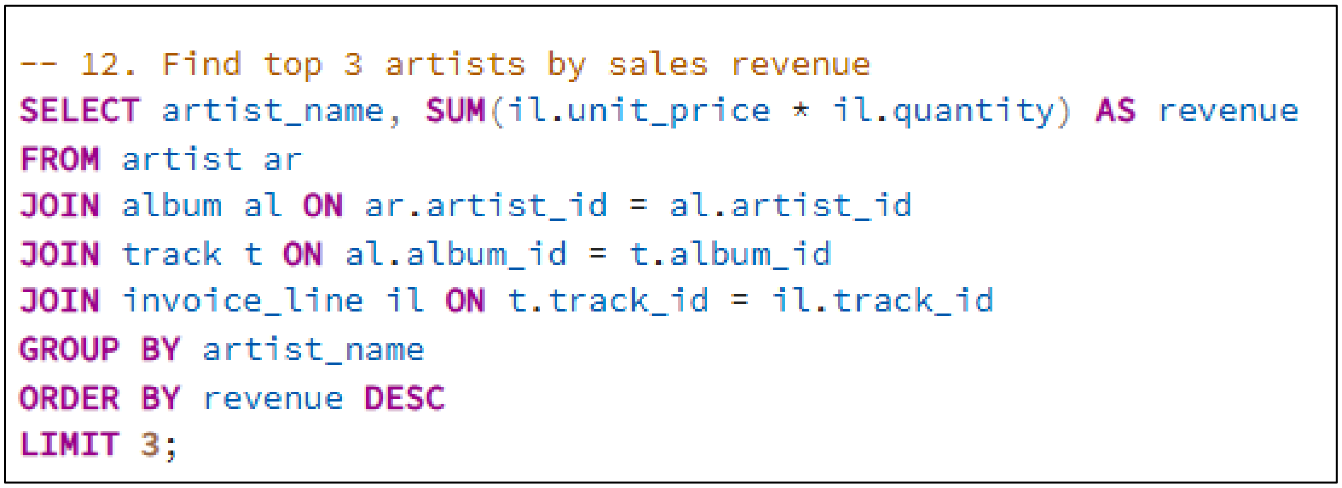
Problem Statement:

Identify the artists generating the highest revenue for the store.

Approach:

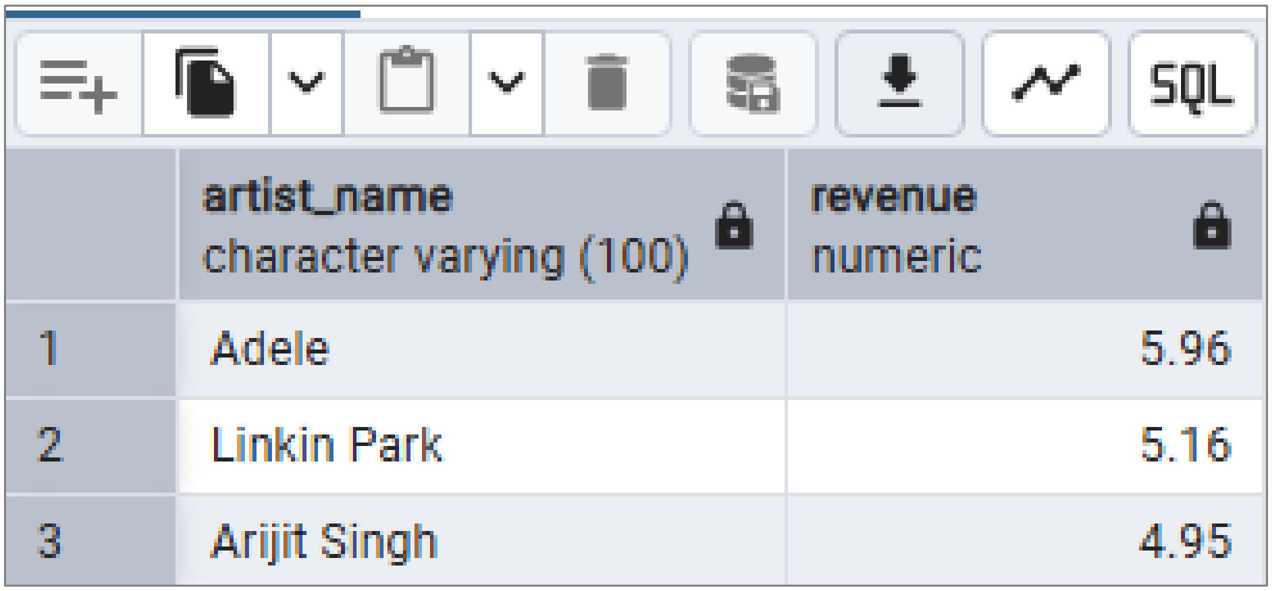
Join artist, album, track, invoice_line; group by artist, sum revenue, and select top 3.

SQL Query :



```sql
-- 12. Find top 3 artists by sales revenue
SELECT artist_name, SUM(il.unit_price * il.quantity) AS revenue
FROM artist ar
JOIN album al ON ar.artist_id = al.artist_id
JOIN track t ON al.album_id = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY artist_name
ORDER BY revenue DESC
LIMIT 3;
```

Output :



	artist_name character varying (100)	revenue numeric
1	Adele	5.96
2	Linkin Park	5.16
3	Arijit Singh	4.95

Analysis:

This shows which artists are driving the most income.

Recommendations:

- Prioritize top artists in promotions.
- Consider exclusive partnerships with these artists.
- Allocate more marketing funds to their albums.

2.8 Find the Most Popular Genre

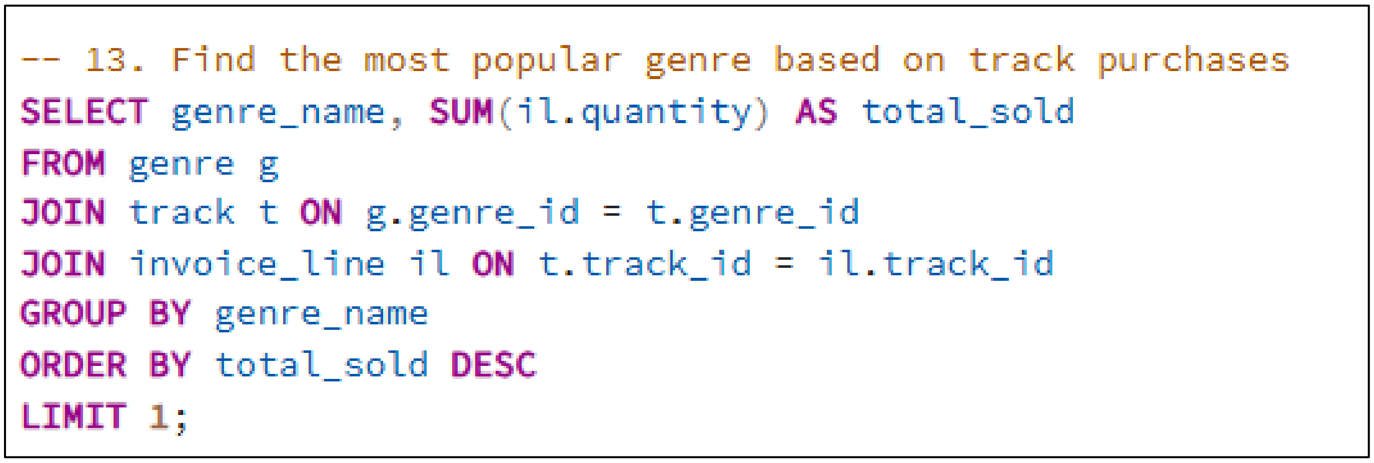
Problem Statement:

Determine which genre has the highest customer purchases.

Approach:

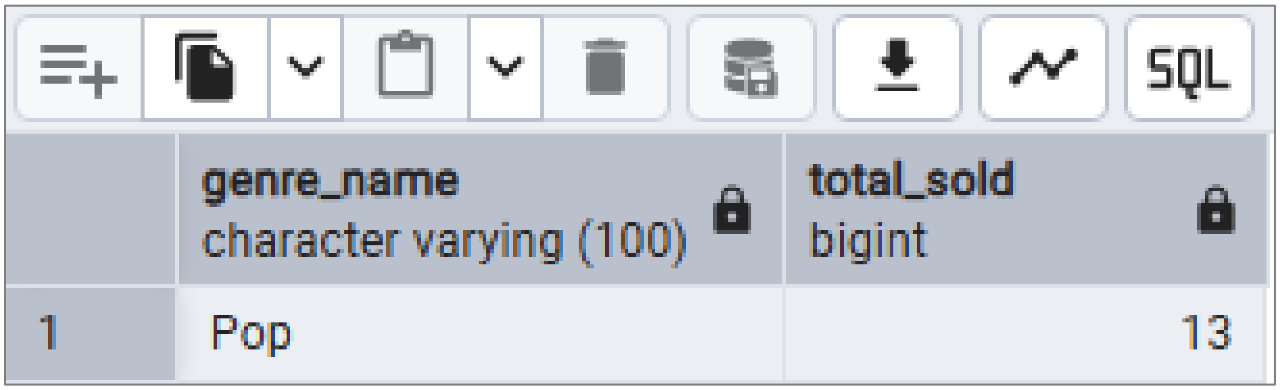
Join genre, track, invoice_line, group by genre, sum quantities, order descending.

SQL Query :



```
-- 13. Find the most popular genre based on track purchases
SELECT genre_name, SUM(il.quantity) AS total_sold
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY genre_name
ORDER BY total_sold DESC
LIMIT 1;
```

Output :



	genre_name character varying (100)	total_sold bigint
1	Pop	13

Analysis:

The output reveals the genre most preferred by customers.

Recommendations:

- Expand catalog in this genre.
- Target ads for this genre to maximize engagement.
- Create more playlists within this category.

ANALYSIS - 3

3.1 Get the Top Customer by Spending

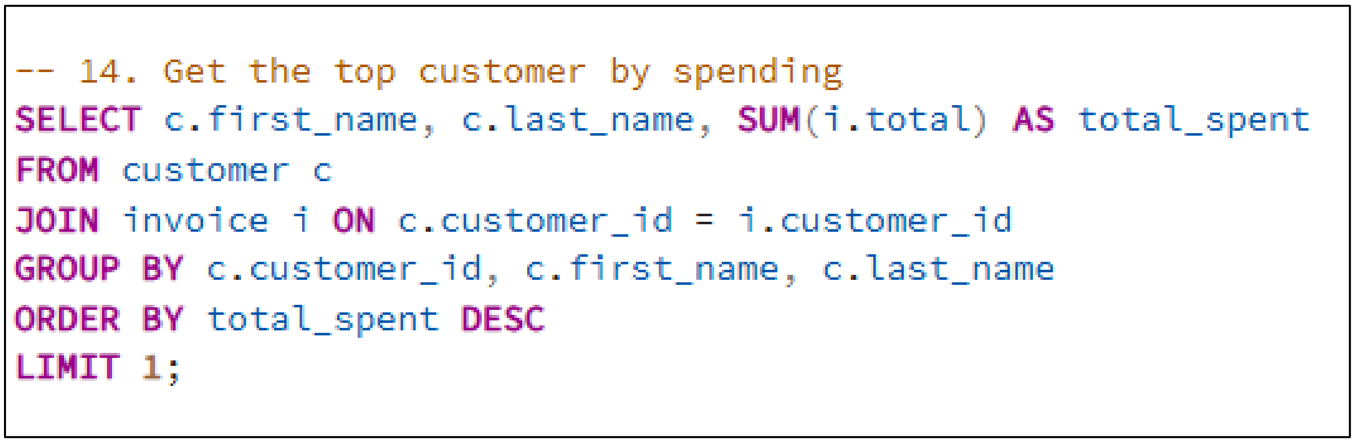
Problem Statement:

Find the single customer who has spent the most money.

Approach:

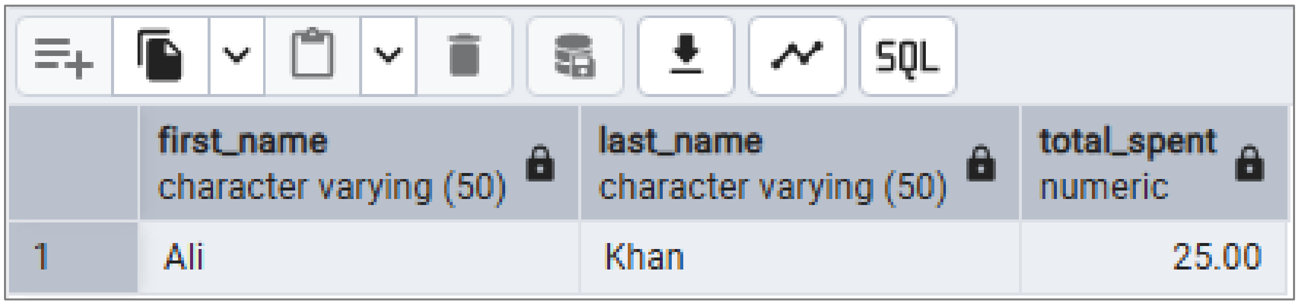
Join customer with invoice, sum totals per customer, order by spending descending, limit 1.

SQL Query :



```sql
-- 14. Get the top customer by spending
SELECT c.first_name, c.last_name, SUM(i.total) AS total_spent
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_spent DESC
LIMIT 1;
```

Output :



	first_name character varying (50)	last_name character varying (50)	total_spent numeric
1	Ali	Khan	25.00

Analysis:

Identifies the top-paying customer who contributes significantly to revenue.

Recommendations:

- Offer this customer premium perks (VIP membership).
- Build stronger customer relationships through personalized engagement.
- Use their buying habits as a model for customer personal building.

3.2 Find Employee with Maximum Customers Handled

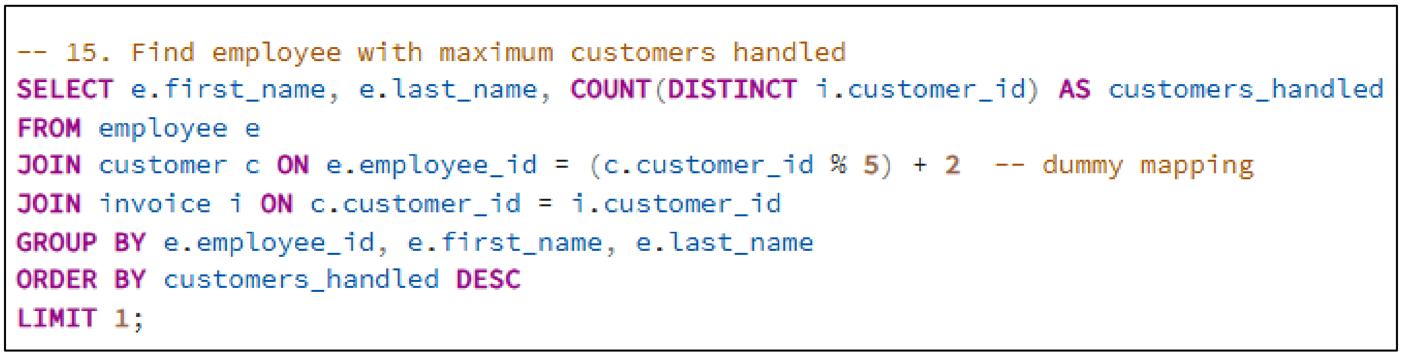
Problem Statement:

Determine which employee manages the largest customer base.

Approach:

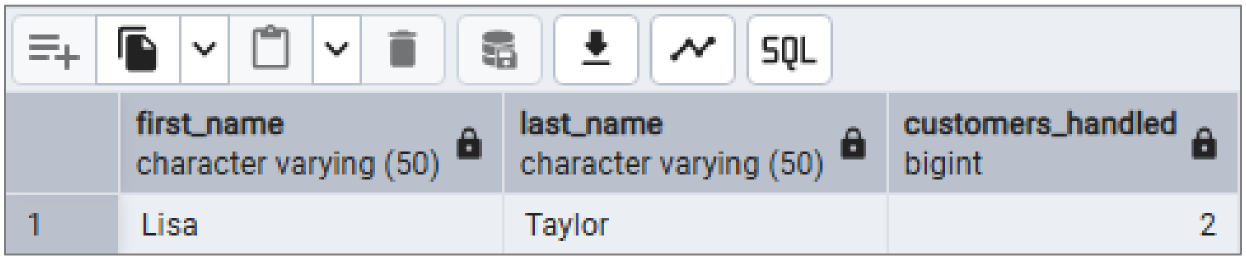
Join employee with customer and invoice, count distinct customers handled by each employee.

SQL Query :



```sql
-- 15. Find employee with maximum customers handled
SELECT e.first_name, e.last_name, COUNT(DISTINCT i.customer_id) AS customers_handled
FROM employee e
JOIN customer c ON e.employee_id = (c.customer_id % 5) + 2  -- dummy mapping
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY e.employee_id, e.first_name, e.last_name
ORDER BY customers_handled DESC
LIMIT 1;
```

Output :



	first_name character varying (50)	last_name character varying (50)	customers_handled bigint
1	Lisa	Taylor	2

Analysis:

Highlights the most engaged or efficient employee.

Recommendations:

- Recognize and reward the employee’s performance.
- Share their best practices with other team members.
- Consider workload balancing if one employee manages too many accounts

3.3 Find Top 3 Customers by Invoices and Spending

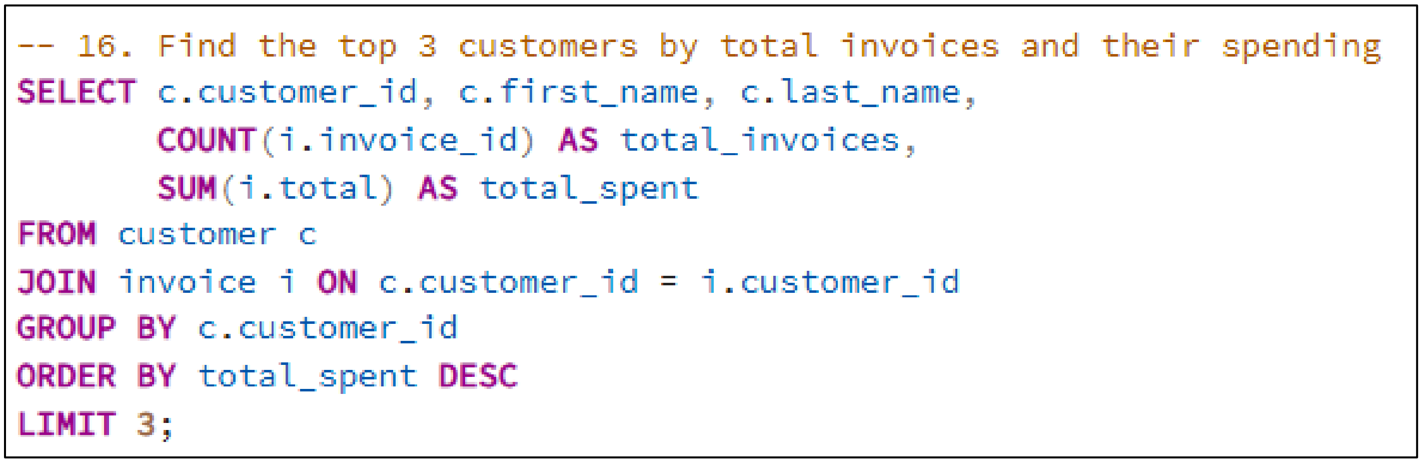
Problem Statement:

Identify customers with the highest invoice counts and spending amounts.

Approach:

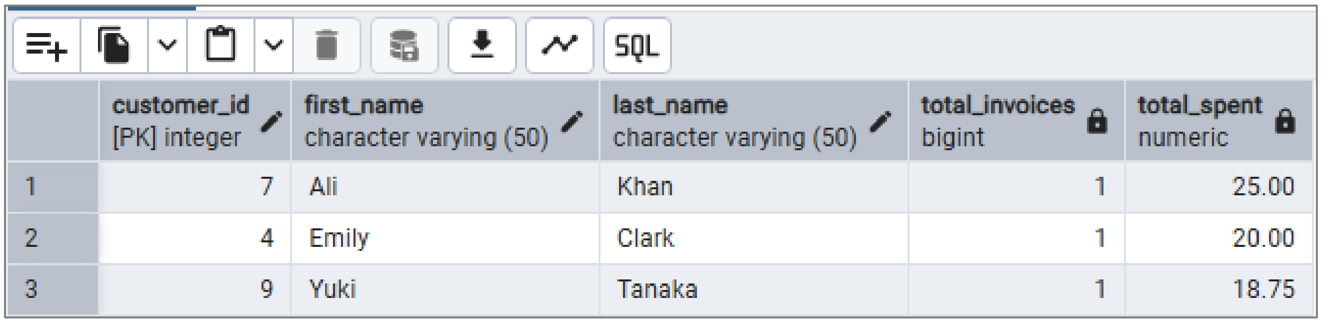
Join customer with invoice, count invoices, sum totals, and rank by spending.

SQL Query :



```sql
-- 16. Find the top 3 customers by total invoices and their spending
SELECT c.customer_id, c.first_name, c.last_name,
       COUNT(i.invoice_id) AS total_invoices,
       SUM(i.total) AS total_spent
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id
ORDER BY total_spent DESC
LIMIT 3;
```

Output :



	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)	total_invoices bigint	total_spent numeric
1	7	Ali	Khan	1	25.00
2	4	Emily	Clark	1	20.00
3	9	Yuki	Tanaka	1	18.75

Analysis:

Shows both frequent and high-value buyers.

Recommendations:

- Design loyalty rewards tailored to these top customers.
- Invite them to beta-test new features.
- Keep strong retention strategies for them.

3.4 Average Track Length Per Genre

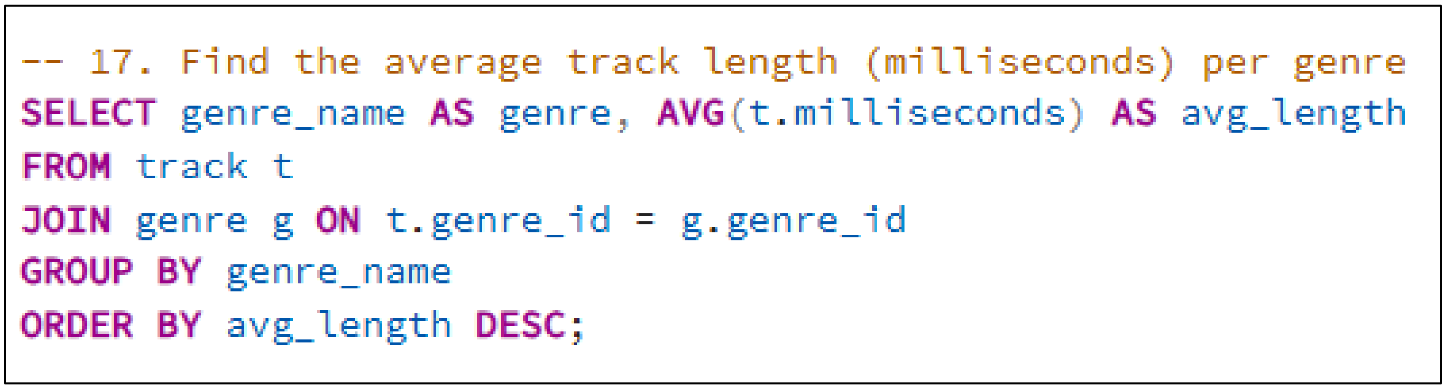
Problem Statement:

Analyze the average track length across genres.

Approach:

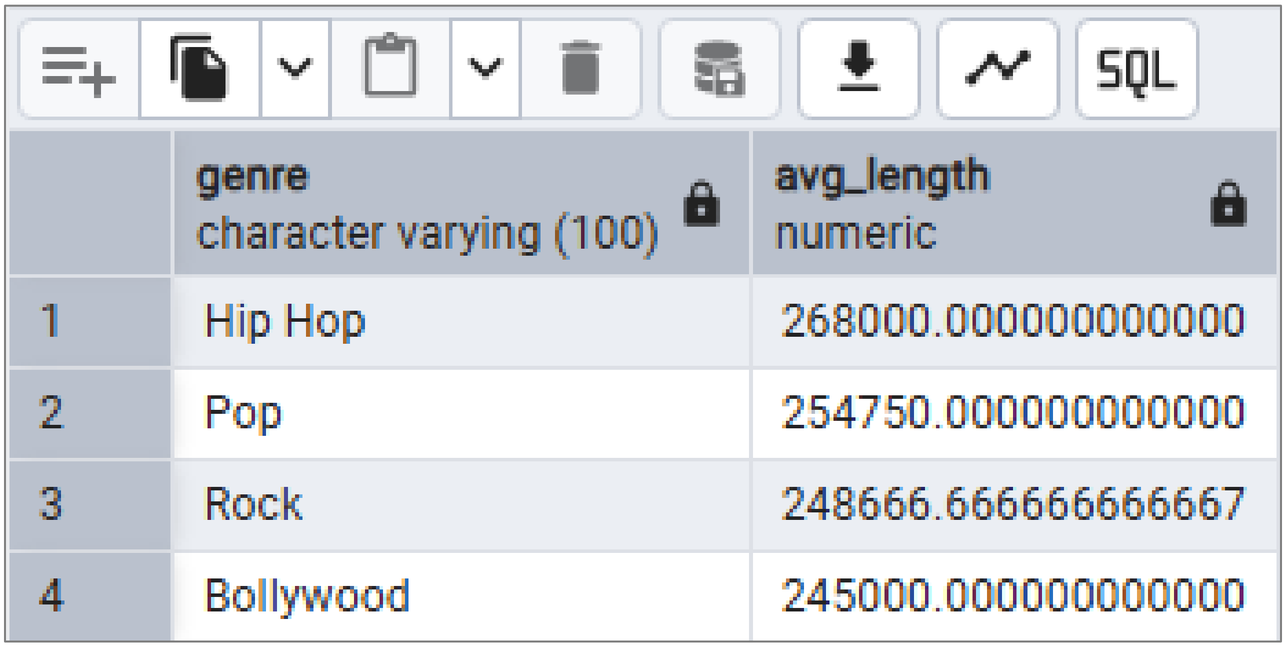
Join track with genre, calculate average milliseconds per genre.

SQL Query :



```sql
-- 17. Find the average track length (milliseconds) per genre
SELECT genre_name AS genre, AVG(t.milliseconds) AS avg_length
FROM track t
JOIN genre g ON t.genre_id = g.genre_id
GROUP BY genre_name
ORDER BY avg_length DESC;
```

Output :



	genre character varying (100)	avg_length numeric
1	Hip Hop	268000.000000000000
2	Pop	254750.000000000000
3	Rock	248666.666666666667
4	Bollywood	245000.000000000000

Analysis:

Identifies whether some genres trend toward longer or shorter tracks.

Recommendations:

- Optimize playlists based on listening duration preferences.
- Inform production teams about average successful track lengths.
- Promote genres that align with user consumption habits (e.g., shorter tracks for workouts).

3.5 Rank Artists by Total Revenue

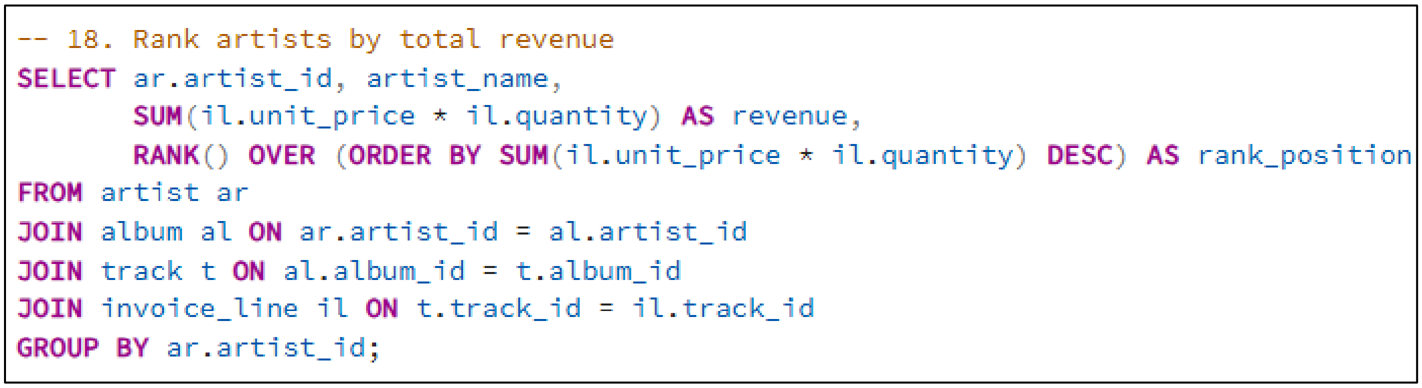
Problem Statement:

Rank artists based on their total contribution to revenue.

Approach:

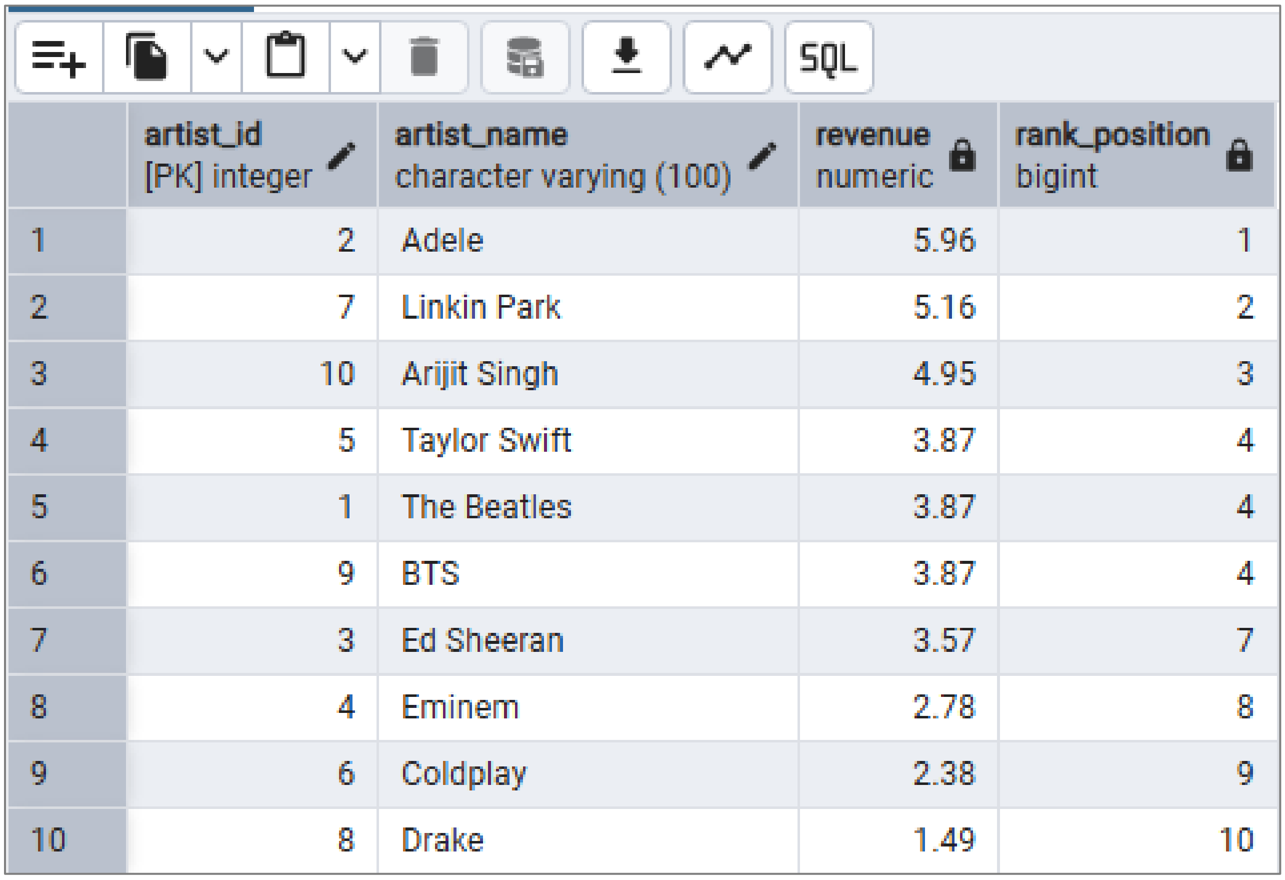
Join artist, album, track, invoice_line; sum sales revenue and apply RANK().

SQL Query :



```sql
-- 18. Rank artists by total revenue
SELECT ar.artist_id, artist_name,
       SUM(il.unit_price * il.quantity) AS revenue,
       RANK() OVER (ORDER BY SUM(il.unit_price * il.quantity) DESC) AS rank_position
FROM artist ar
JOIN album al ON ar.artist_id = al.artist_id
JOIN track t ON al.album_id = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY ar.artist_id;
```

Output :



	artist_id [PK] integer	artist_name character varying (100)	revenue numeric	rank_position bigint
1	2	Adele	5.96	1
2	7	Linkin Park	5.16	2
3	10	Arijit Singh	4.95	3
4	5	Taylor Swift	3.87	4
5	1	The Beatles	3.87	4
6	9	BTS	3.87	4
7	3	Ed Sheeran	3.57	7
8	4	Eminem	2.78	8
9	6	Coldplay	2.38	9
10	8	Drake	1.49	10

Analysis:

Produces a leaderboard of artists by revenue impact.

Recommendations:

- Increase collaborations with top artists.
- Use rankings to negotiate licensing and contracts.
- Track mid-tier artists for growth opportunities.