

Finding the Top 10 Artists by Sales in SQL Server: A Comprehensive Guide

Introduction

When working with sales data in SQL Server, we often need to **rank and filter the top-performing entities**, such as artists, products, or customers. This article will explore **three approaches** to finding the **Top 10 artists by sales** in the Chinook database.

Each approach has strengths and weaknesses, and we'll discuss when to use each. By the end, you'll understand:

- How to **calculate total sales correctly**.
- How to **filter the Top 10 artists accurately**.
- How to use **ranking functions** (**RANK()******, **DENSE_RANK()**) effectively.
- How to handle **ties properly**.

The Problem: Finding the Top 10 Artists by Sales

We want to generate a report that:

1. **Calculates total sales per artist.**
2. **Filters to show only the Top 10 artists.**
3. **Handles ties correctly** (artists with the same sales should be ranked together).
4. **Different SQL techniques are used to achieve the same result.**

We will explore **four different solutions** and discuss their pros and cons.

Dataset Overview & SQL Joins

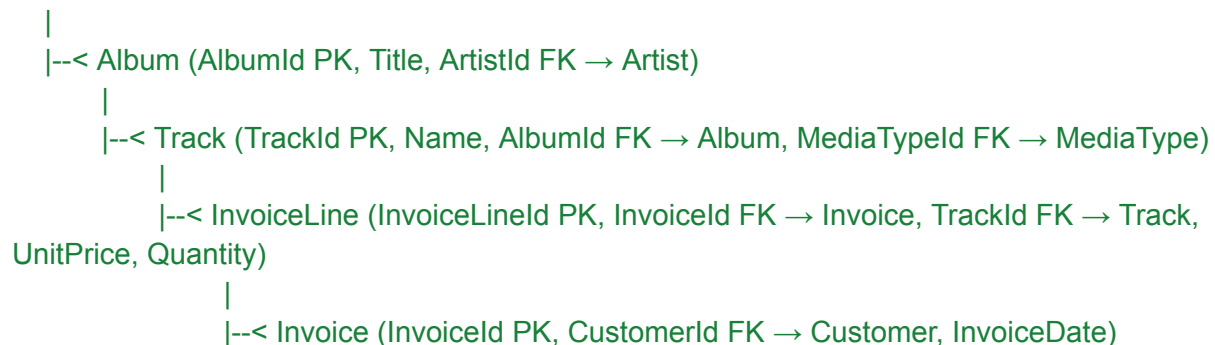
We are using the **Chinook database**, which contains:

- **Artist** (Artist information)
- **Album** (Each album belongs to an artist)
- **Track** (Each track belongs to an album)
- **InvoiceLine** (Sales data, referencing tracks)
- **Invoice** (Details of transactions)

Entity-Relationship Diagram (ERD)

Text-Based ERD with Primary and Foreign Keys

Artist (**ArtistId** PK, Name)



Understanding the Joins

1. **Join **Artist** to ****Album** → Each **artist** has multiple **albums**, so we join on **ArtistId**.
2. **Join **Album** to ****Track** → Each **album** has multiple **tracks**, so we join on **AlbumId**.
3. **Join **Track** to ****InvoiceLine** → Each **track** may be sold multiple times, so we join on **TrackId**.
4. **Join **InvoiceLine** to ****Invoice** → To get **date-based filtering**, we join **InvoiceLine** to **Invoice** using **InvoiceId**.
5. **Join **Track** to ****MediaType** → To **filter out video tracks**, we join on **MediaTypeId**.

Correct SQL Joins to Aggregate Sales

Before ranking, we need to **calculate total sales per artist correctly**. The following query ensures we **properly join** all relevant tables:

```
WITH SalesByArtist AS (
  SELECT
    ar.Name AS Artist,
    SUM(il.UnitPrice * il.Quantity) AS TotalSales
  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
  JOIN Invoice i ON il.InvoiceId = i.InvoiceId
  JOIN MediaType mt ON t.MediaTypeId = mt.MediaTypeId
  WHERE i.InvoiceDate BETWEEN '2011-07-01' AND '2012-06-30'
  AND mt.Name NOT LIKE '%Video%' -- Exclude video tracks
```

```
GROUP BY ar.ArtistId, ar.Name  
)
```

Let's explore **three ways** to filter and rank the Top 10 artists.

****Solution 1: Using `MIN(TotalSales)` Without `**WITH TIES`**

```
WITH SalesByArtist AS (...)  
SELECT Artist, TotalSales  
FROM SalesByArtist  
WHERE TotalSales >= (  
    SELECT MIN(TotalSales)  
    FROM (  
        SELECT DISTINCT TOP 10 TotalSales  
        FROM SalesByArtist  
        ORDER BY TotalSales DESC  
    ) AS Top10  
)  
ORDER BY TotalSales DESC;
```

Strengths:

- More precise than other filtering methods—avoids `WITH TIES` errors.
- Ensures exactly 10 distinct sales values are considered.

Weaknesses:

- Still doesn't explicitly rank artists.
- Could return **more than 10 artists** if many ties exist.

Best Use Case:

When you need a **clean Top 10 filter without explicit ranking numbers**.

****Solution 2: Using `**RANK()`**

```
WITH SalesByArtist AS (...),
RankedArtists AS (
  SELECT
    Artist,
    TotalSales,
    RANK() OVER (ORDER BY TotalSales DESC) AS RankPosition
  FROM SalesByArtist
)
SELECT Artist, TotalSales, RankPosition
FROM RankedArtists
WHERE RankPosition <= 10
ORDER BY RankPosition;
```

✓ Strengths:

- **Explicitly assigns rank numbers.**
- Handles ties properly.
- Easy to modify for Top 5, Top 20, etc.

✗ Weaknesses:

- **RANK()** skips numbers when there are ties (e.g., if three artists are ranked #5, the next rank is #8).
- Could return **fewer than 10 artists** if many ties occur.

🚀 Best Use Case:

When you need a **clear ranking system**, but skipping ranks is acceptable.

Solution 3: Using **DENSE_RANK()** (Recommended)

```
WITH SalesByArtist AS (...),
RankedArtists AS (
  SELECT
    Artist,
    TotalSales,
    DENSE_RANK() OVER (ORDER BY TotalSales DESC) AS RankPosition
  FROM SalesByArtist
)
SELECT Artist, TotalSales, RankPosition
FROM RankedArtists
```

```
WHERE RankPosition <= 10
ORDER BY RankPosition;
```

✅ Strengths:

- Does not skip rank numbers.
- Handles ties correctly.
- More **consistent than ****RANK()** for ensuring 10 artists appear.

❌ Weaknesses:

- Could return **more than 10 artists** if many ties occur at rank 10.

🚀 Best Use Case:

When you need ranking numbers without skipping ranks.

Final Recommendation: Which Query to Use?

Use Case	Best Query
✅ Simple and efficient Top 10 filter	Solution 1 (MIN(TotalSales))
✅ Explicit ranking with gaps	Solution 2 (RANK())
✅ Explicit ranking without gaps	Solution 3 (DENSE_RANK())

🚀 For best accuracy, use Solution 3 (**DENSE_RANK()******).

🎯 Now you're ready to filter and rank your data with confidence!