Project Title: Chinook Data Analytics - Unlocking Business Insights from a Digital Media Store

Project Overview:

This project involves an in-depth exploration and analysis of the **Chinook database**, a comprehensive sample dataset representing a digital media store. Leveraging SQL (Structured Query Language), the goal is to extract, transform, and analyze various facets of the store's operations, including sales performance, customer behaviour, employee efficiency, and media catalogue trends. This project will demonstrate proficiency in a wide range of SQL concepts, from basic data retrieval to complex analytical queries, ultimately providing actionable business insights.

Context:

As of July 2025, understanding customer purchasing habits, optimizing inventory, and evaluating sales strategies are more critical than ever for digital media businesses. This project simulates a real-world scenario where a database professional is tasked with providing data-driven recommendations to improve the Chinook store's profitability and operational efficiency.

Project Objectives:

The project will address key business questions and involve the following objectives:

Database Understanding & Schema Navigation:

- Thoroughly understand the Chinook database schema, including all tables (Artists, Albums, Tracks, Customers, Employees, Invoices, Invoice Lines, Genres, Media Types, Playlists, Playlist Track) and their relationships (Primary Key/Foreign Key constraints).
- Create an Entity-Relationship Diagram (ERD) or describe the key relationships within the database.

Core Data Retrieval & Manipulation:

- Practice basic SELECT statements with WHERE, ORDER BY, and LIMIT/TOP clauses.
- Demonstrate proficiency in INSERT, UPDATE, and DELETE operations for data management (e.g., adding a new track, updating customer details).

Advanced Querying & Data Aggregation:

- **Joining Tables:** Utilize various types of JOIN clauses (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN) to combine data from multiple tables to answer complex questions (e.g., "List all tracks by a specific artist and their genre").
- **Aggregation and Grouping:** Employ GROUP BY and aggregate functions (COUNT, SUM, AVG, MIN, MAX) with HAVING clauses to perform statistical analysis (e.g., "Calculate total sales per country," "Find the average invoice amount for each customer").
- **Subqueries & Common Table Expressions (CTEs):** Apply subqueries and/or CTEs for multi-step data processing and improved query readability (e.g., "Find customers who have spent more than the average customer spending").
- **Window Functions:** Explore the use of window functions for more sophisticated analytical tasks (e.g., "Rank employees by sales performance within their respective departments").

Business Insights & Reporting:

- **Sales Analysis:**
  - Identify top-performing artists and genres by revenue.
  - Analyze monthly/quarterly sales trends over the four-year period.
  - Determine the average value of an invoice and the average number of items per invoice.
- **Customer Behaviour Analysis:**
  - Segment customers based on their total spending.
  - Identify the top 10 most valuable customers.
  - Determine the most popular genres or artists among specific customer groups.
- **Employee Performance:**
  - Evaluate sales performance of individual employees and their contribution to overall revenue.
  - Analyze employee hierarchy and reporting structures.

- **Media Catalogue Analysis:**
    - Identify the distribution of tracks by genre and media type.
    - Analyze the average duration of tracks per album or genre.

Technology Used:

- **Database:** Chinook Database
- **Language:** SQL (MySQL)
- **Tools:** MySQL Workbench

Deliverables:

- A documented SQL script file (.sql) containing all queries used to achieve the project objectives, clearly commented and organized by objective.
- A brief project report summarizing the key findings, insights derived from the SQL queries, and potential business recommendations.

Data Cleaning

Replace Null value in each table →

Customer Table --

```
-- Removing null value with "Not available"
set sql_safe_updates = 0;

UPDATE customer
SET
    company = COALESCE(Company, 'Not Available'),
    State = COALESCE(State, 'Not Available'),
    Fax = COALESCE(Fax, 'Not Available'),
    Phone = COALESCE(Phone, 'Not Available'),
    PostalCode = COALESCE(PostalCode, 'N/A');
```

Invoice Table –

```
UPDATE invoice
SET
    BillingState = COALESCE(BillingState, 'Not Availeble'),
    BillingPostalCode = COALESCE(BillingPostalCode, 'N/A');
```

Table Track –

```
select * from track;
update track
set
Composer = coalesce(Composer,'Not Available');
```

Data Analysis

Q.1 Total Sales by Genre:

→

```
SELECT
    g.name, SUM(i.unitprice * i.quantity) AS Total_Sales
FROM
    Genre g
        JOIN
    Track t ON g.GenreId = t.GenreId
        JOIN
    InvoiceLine i ON t.trackId = i.trackId
GROUP BY g.name
ORDER BY Total_Sales DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	Total_Sales
Rock	826.65
Latin	382.14
Metal	261.36
Alternative & Punk	241.56
TV Shows	93.53
Jazz	79.20
Blues	60.39
Drama	57.71

Q.2 Top 10 Selling Tracks:

-- Identify the top 10 Track Names that have generated the highest total revenue. Include the track name and total revenue.

→

```
SELECT
    t.name, SUM(iv.unitprice * iv.quantity) AS Total_revenue
FROM
    track t
        JOIN
    invoiceline iv ON t.TrackId = iv.TrackId
GROUP BY t.name
ORDER BY Total_revenue DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch

name	Total_revenue
Dazed and Confused	4.95
The Trooper	4.95
The Fix	3.98
Pilot	3.98
Branch Closing	3.98
Gay Witch Hunt	3.98
Walkabout	3.98
The Woman King	3.98
How to Stop an Exploding Man	3.98
Phyllis's Wedding	3.98

Q.3 Average Track Length by Genre:

-- For each Genre Name, calculate the average Milliseconds of its tracks. Convert milliseconds to minutes.

→

```
SELECT
    g.name,
    ROUND(AVG(t.milliseconds / 60000.0), 2) AS Track_len_minutes
FROM
    track t
        JOIN
    genre g ON t.genreId = g.genreId
GROUP BY g.name
ORDER BY Track_len_minutes DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	Track_len_minutes
Sci Fi & Fantasy	48.53
Science Fiction	43.76
Drama	42.92
TV Shows	35.75
Comedy	26.42
Metal	5.16
Electronica/Dance	5.05
Heavy Metal	4.96
Classical	4.90
Jazz	4.86
Rock	4.73
Blues	4.51

Q.4 Tracks Sold in a Specific Year:

-- Count the total number of Invoice Line items that occurred in the year 2022.

→

```
SELECT
    YEAR(i.invoicedate) AS s_year,
    COUNT(t.trackid) AS total_sold_track
FROM
    track t
        JOIN
    invoiceline iv ON t.trackid = iv.trackid
        JOIN
    invoice i ON iv.invoiceid = i.invoiceid
GROUP BY s_year
HAVING s_year = '2022'
ORDER BY total_sold_track DESC;
```

Result Grid | Filter Rows: | Export: | Wrap

s_year	total_sold_track
2022	455

Q.5 Customers Who Purchased 'Blues' Genre:

-- List FirstName, LastName of Customers who have purchased at least one Track from the 'Blues' Genre. Do not list duplicates.

→

```
SELECT DISTINCT
    c.firstname, c.lastname
FROM
    customer c
        JOIN
    invoice i ON c.customerid = i.customerid
        JOIN
    invoiceline iv ON i.invoiceid = iv.invoiceid
        JOIN
    track t ON iv.trackid = t.trackid
        JOIN
    genre g ON t.genreid = g.genreid
WHERE
    g.name = 'Blues';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

firstname	lastname
Leonie	Köhler
Ladislav	Kovács
John	Gordon
Helena	Holý
Marc	Dubois
Luis	Rojas
Tim	Goyer
Lucas	Mancini

Q.6 Employees and Their Direct Manager:

-- List Employee FirstName, LastName (as 'Employee Name') and their manager's FirstName, LastName (as 'Manager Name').

→

```
SELECT
    CONCAT(e.firstname, ' ', e.lastname) AS employee_name,
    CONCAT(m.firstname, ' ', m.lastname) AS manager_name
FROM
    employee e
        LEFT JOIN
    employee m ON e.reportsto = m.employeeid;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_name	manager_name
Andrew Adams	NULL
Nancy Edwards	Andrew Adams
Jane Peacock	Nancy Edwards
Margaret Park	Nancy Edwards
Steve Johnson	Nancy Edwards
Michael Mitchell	Andrew Adams
Robert King	Michael Mitchell
Laura Callahan	Michael Mitchell

Q.7 Top 3 Support Representatives by Sales:

-- Identify the top 3 Employees (FirstName, LastName, Title) who are 'Sales Support Agent's and have generated the most total sales (Invoice, Total) from their supported customers.

→

```
SELECT
    e.firstname,
    e.lastname,
    e.title,
    SUM(i.total) AS total_sales
FROM
    employee e
        JOIN
    customer c ON e.employeeid = c.supportrepid
        JOIN
    invoice i ON c.customerid = i.customerid
WHERE
    e.title = 'Sales Support Agent'
GROUP BY e.firstname , e.lastname , e.title
ORDER BY total_sales DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

firstname	lastname	title	total_sales
Jane	Peacock	Sales Support Agent	833.04
Margaret	Park	Sales Support Agent	775.40
Steve	Johnson	Sales Support Agent	720.16

Q.8 Average Sales per Employee (as Support Rep):

-- For each Employee who acts as a SupportRep, calculate the average Total of the Invoices associated with their customers

→

```
SELECT
    CONCAT(e.firstname, '  ', e.lastname) AS Sales_Support_Agent,
    round(AVG(i.total),3) AS avg_total_sales
FROM
    employee e
        JOIN
    customer c ON e.employeeid = c.supportrepid
        JOIN
    invoice i ON c.customerid = i.customerid
WHERE
    e.title = 'sales support agent'
GROUP BY Sales_Support_Agent
ORDER BY avg_total_sales DESC;

-- Q.9 Customers from Cities with an Employee:
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Sales_Support_Agent	avg_total_sales
Steve Johnson	5.716
Jane Peacock	5.706
Margaret Park	5.539

Q.9 Customers from Cities with an Employee:

-- Find the FirstName, LastName of Customers who live in the same City as at least one Employee. Do not list duplicates.

→

```
SELECT
    distinct c.firstname, c.lastname
FROM
    customer c
WHERE
    c.city in (select distinct city from employee);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conter

firstname	lastname
Mark	Philips

Q.10 Invoices with More Than 5 Tracks:

-- List InvoiceId, CustomerId, and Total for invoices that contain more than 5 distinct Tracks.

→

```
with Track_count as (
        select distinct(count(trackid)) as distinct_track,invoiceid
        from invoiceline
        group by invoiceid
        having distinct_track >5)

    select i.invoiceid,i.customerid,i.total
    from invoice i
    join track_count tc
    on i.invoiceid = tc.invoiceid
    order by i.total desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

invoiceid	customerid	total
404	6	25.86
299	26	23.86
96	45	21.86
194	46	21.86
89	7	18.86
201	25	18.86
88	57	17.91

Q.11 Customers Who Purchased a Specific Track:

-- List FirstName, LastName of Customers who purchased the track 'Bohemian Rhapsody'.

→

```
SELECT
    c.firstname, c.lastname
FROM
    customer c
        JOIN
    invoice i ON c.customerid = i.customerid
        JOIN
    invoiceline iv ON iv.invoiceid = i.invoiceid
        JOIN
    track t ON iv.trackid = t.trackid
WHERE
    t.name = 'Bohemian Rhapsody';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conter

firstname	lastname
Enrique	Muñoz

Q.12 Monthly Sales Trend:

-- Calculate the total sales (Invoice.Total) for each month of each year. Display InvoiceYear, InvoiceMonth, and TotalSales, ordered chronologically.

→

```
SELECT
    MONTH(invoicedate) AS Invoice_month,
    YEAR(invoicedate) AS Invoice_year,
    SUM(total) AS total_sales
FROM
    invoice
GROUP BY YEAR(invoicedate) , MONTH(invoicedate)
ORDER BY Invoice_year , Invoice_month;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

Invoice_month	Invoice_year	total_sales
1	2021	35.64
2	2021	37.62
3	2021	37.62
4	2021	37.62
5	2021	37.62
6	2021	37.62
7	2021	37.62
8	2021	37.62
9	2021	37.62

Q.13 Artists Whose Tracks Have Never Been Sold:

-- List Artist Names whose Albums have Tracks that have never appeared in any Invoice Line.

→

```
SELECT DISTINCT
    ar.name
FROM
    album a
        JOIN
    artist ar ON a.artistid = ar.artistid
        JOIN
    track t ON a.albumid = t.albumid
        LEFT JOIN
    invoiceline iv ON iv.trackid = t.trackid
WHERE
    iv.trackid IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap C

name
AC/DC
Aerosmith
Alanis Morissette
Alice In Chains
Antônio Carlos Jobim
Apocalyptica
Audioslave
BackBeat
Billy Cobham
Black Label Society

Q.14. Customers Who Purchased Every Genre:

-- Find the FirstName and LastName of Customers who have purchased at least one Track from every single Genre available in the store.

→

```
with customer_genre as (
    select c.customerid,c.firstname,c.lastname,
    count( distinct t.genreid) as genre_count
    from customer c
    join invoice i
    on c.customerid = i.customerid
    join invoiceline iv
    on i.invoiceid = iv.invoiceid
    join track t
    on iv.trackid = t.trackid
    group by c.customerid,c.firstname,c.lastname),
    total_genre as (
        select  count(distinct genreid) as total_genre_count
        from genre)
    select cg.firstname,cg.lastname
    from customer_genre cg,total_genre tg
    where cg.genre_count = tg.total_genre_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

firstname	lastname

Q.15 Tracks Available in Playlists but Not Sold:

-- Find Track Names that exist in at least one Playlist but have never been sold in an Invoice Line.

→

```
SELECT DISTINCT
    t.name
FROM
    playlisttrack pt
        JOIN
    track t ON pt.trackid = t.trackid
        LEFT JOIN
    invoiceline iv ON t.trackid = iv.trackid
WHERE
    iv.trackid IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap

name
Let's Get It Up
C.O.D.
Let There Be Rock
Bad Boy Boogie
Whole Lotta Rosie

Q.16 Employee Tenure (in years):

-- For each Employee (FirstName, LastName), calculate their tenure in years from their Hire Date to the current date.

→

```
SELECT
    firstname,
    lastname,
    DATE(hiredate) AS Hire_Date,
    CURDATE() AS Today,
    TIMESTAMPDIFF(YEAR, hiredate, CURDATE()) AS tenure_in_years
FROM
    employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

firstname	lastname	Hire_Date	Today	tenure_in_years
Andrew	Adams	2002-08-14	2025-07-09	22
Nancy	Edwards	2002-05-01	2025-07-09	23
Jane	Peacock	2002-04-01	2025-07-09	23
Margaret	Park	2003-05-03	2025-07-09	22
Steve	Johnson	2003-10-17	2025-07-09	21
Michael	Mitchell	2003-10-17	2025-07-09	21
Robert	King	2004-01-02	2025-07-09	21
Laura	Callahan	2004-03-04	2025-07-09	21

Q.17 Customers Who Are Managers:

-- Identify if any Customer (FirstName, LastName) is also an Employee and holds a Title like 'Manager' or 'General Manager'.

→

```
SELECT
    c.firstname, c.lastname
FROM
    customer c
        JOIN
    employee e ON c.firstname = e.firstname
        AND c.lastname = e.lastname
WHERE
    e.title LIKE '%Manager%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

firstname	lastname

Q.18 Customer Acquisition by Support Rep:

-- For each SupportRep (FirstName, LastName), count the number of new Customers they acquired in each year based on Customer.SupportRepId and Customer.

→

```
SELECT
    e.firstname,
    e.lastname,
    YEAR(e.hiredate) AS aquire_year,
    COUNT(DISTINCT c.customerid) AS new_customer
FROM
    customer c
        left JOIN
    employee e ON c.supportrepid = e.employeeid
WHERE
    e.hiredate IS NOT NULL
GROUP BY e.firstname , e.lastname , aquire_year
ORDER BY e.firstname , e.lastname , aquire_year;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

firstname	lastname	aquire_year	new_customer
Jane	Peacock	2002	21
Margaret	Park	2003	20
Steve	Johnson	2003	18

Q.19 Albums with Tracks from Multiple Composers:

-- List Album Titles that contain tracks from more than one unique Composer.

→

```
SELECT DISTINCT
    a.title
FROM
    album a
        JOIN
    track t ON a.albumid = t.albumid
GROUP BY a.title
HAVING COUNT(DISTINCT t.composer) > 1;
```

Result Grid | Filter Rows: | Export: | W

title
...And Justice For All
A Real Dead One
A Real Live One
A TempestadeTempestade Ou O Livro Dos Dias
A-Sides
Ace Of Spades
Acústico MTV
Acústico MTV [Live]

Q.20 Top 5 Busiest Invoice Dates:

-- Find the top 5 Invoice Dates (just the date part) that had the highest number of Invoices.

→

```
SELECT
    DATE(invoicedate) AS i_date,
    COUNT(invoiceid) AS total_invoice
FROM
    invoice
GROUP BY i_date
ORDER BY total_invoice DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export:

i_date	total_invoice
2021-04-04	2
2021-05-05	2
2021-03-04	2
2021-06-05	2
2021-02-01	2

Q.21 Top Spenders by Genre:

-- Find the top 3 customers in each country who have spent the most money on "Rock" music. For each customer, list their full name, country, and total amount spent on Rock music.

→

```
with total_spent_by_ct as (
    select c.customerid,c.firstname,c.lastname,c.country,sum(i.total) as total_spent
    from customer c
    join invoice i
    on c.customerID = i.customerID
    join invoiceline iv
    on i.invoiceId = iv.invoiceId
    join track t
    on iv.trackid = t.trackId
    join genre g
    on t.genreId = g.genreId
    where g.name = 'rock'
    group by c.customerid
    ),
    ranked_customers as(
        select concat(ts.firstname, " ",ts.lastname) as cust_name ,ts.country,total_spent,
        rank() over (partition by ts.country order by ts.total_spent desc) as rank_no
        from total_spent_by_ct ts
        )
    SELECT
    cust_name, country, total_spent, rank_no
FROM
    ranked_customers
WHERE
    rank_no <= 3
ORDER BY country , rank_no;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cust_name	country	total_spent	rank_no
Diego Gutiérrez	Argentina	81.18	1
Mark Taylor	Australia	170.28	1
Astrid Gruber	Austria	203.40	1
Daan Peeters	Belgium	209.88	1
Eduardo Martins	Brazil	244.53	1
Alexandre Rocha	Brazil	142.56	2
Roberto Almeida	Brazil	113.85	3

Q.22 Artist Popularity by Track Count and Revenue:

-- Identify artists who have more than 10 tracks and whose total revenue generated from sales is above the average revenue generated by all artists. Display the artist's name, the number of tracks, and their total revenue, ordered by revenue in descending order.

→

```sql
with total_revenue as (
    select ar.artistid,ar.name as Artist_Name , sum(iv.unitprice * iv.quantity) as total_revenue,
            count(t.trackid) as Track_Count
    from album a
    join artist ar
    on a.artistId =ar.artistId
    join track t
    on a.albumid = t.albumid
    join invoiceline iv
    on t.trackid = iv.trackid
    group by ar.artistid , ar.name
    order by total_revenue desc
    ),
    avg_revenue as (
        select avg(total_revenue) as avg_revenue
        from total_revenue
        )

    SELECT
    tr.Artist_Name, tr.total_revenue, tr.track_count
FROM
    total_revenue tr
        CROSS JOIN
    avg_revenue ar
WHERE
    tr.track_count > 10
        AND tr.total_revenue > ar.avg_revenue
ORDER BY tr.total_revenue DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Artist_Name	total_revenue	track_count
Iron Maiden	138.60	140
U2	105.93	107
Metallica	90.09	91
Led Zeppelin	86.13	87
Lost	81.59	41

Q.23 Customer Retention Analysis (Monthly):

-- For each month, calculate the number of new customers acquired and the number of existing customers who made a purchase in that month.

→

```sql
with cust_first_purchase as (
    select c.customerId , min(date(i.invoicedate)) as first_purchase
    from customer c
    join invoice i
    on c.customerId = i.customerID
    group by c.customerId
    )
    select month(i.invoicedate) as month_sales,
        count(distinct case
                when month(i.invoicedate) = month(cfp.first_purchase) then i.customerID
                else null
                end) as new_customer,
        count(distinct case
                when month(i.invoicedate) > month(cfp.first_purchase)
                or year(i.invoicedate) > year(cfp.first_purchase) then i.customerID
                else null
                end) as ext_customer
            from invoice i
            join cust_first_purchase cfp
            on i.customerid = cfp.customerid
            group by month_sales
            order by month_sales;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month_sales	new_customer	ext_customer
1	8	26
2	8	25
3	8	27
4	7	26
5	6	29
6	6	29
7	5	30
8	3	32
9	2	31
10	2	33

Q.24 Employee Performance vs. Sales Targets:

-- Assume a sales target of $800 for each invoice line item. For each sales support agent, calculate their total sales, the number of invoice line items they processed, and the percentage by which they exceeded or fell short of their hypothetical sales target.

→

```sql
with total_sales_by_emoloyee as (
        select concat(e.firstname, ' ' , e.lastname) as emp_name , sum(iv.unitprice * iv.quantity) as total_sale,count(iv.quantity) as items
        from employee e
        join customer c
        on e.employeeid = c.supportrepid
        join invoice i
        on c.customerid = i.customerid
        join invoiceline iv
        on i.invoiceid = iv.invoiceid
        where e.title = 'Sales support Agent'
        group by emp_name
        )

        select tse.emp_name,tse.total_sale,tse.items,
            case when total_sale =  '800' then 'Close to Achieve'
                when total_sale > '800' then ' Target Acieved'
                else ' Target Not Achieved'
                end as Employee_Performance
            from total_sales_by_emoloyee tse;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

emp_name	total_sale	items	Employee_Performance
Jane Peacock	833.04	796	Target Acieved
Margaret Park	775.40	760	Target Not Achieved
Steve Johnson	720.16	684	Target Not Achieved

Q.25 Playlist Content Overlap:

-- Find pairs of playlists that share at least 5 common tracks. For each pair, display the names of both playlists and the count of shared tracks.

→

```sql
SELECT
    p1.name play_1,
    p2.name play_2,
    COUNT(pl1.trackid) AS track_count
FROM
    playlisttrack pl1
        JOIN
    playlisttrack pl2 ON pl1.trackid = pl2.trackid
        AND pl1.playlistid < pl2.playlistid
        JOIN
    playlist p1 ON pl1.playlistid = p1.playlistid
        JOIN
    playlist p2 ON pl2.playlistid = p2.playlistid
GROUP BY p1.name , p2.name
HAVING track_count >= 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

play_1	play_2	track_count
Music	Music	3290
Music	Heavy Metal Classic	52
Music	90's Music	1477
Music	Grunge	30
Music	Brazilian Music	78
Music	Classical	150
Music	Classical 101 - The Basics	50
Music	Classical 101 - Next Steps	50
Music	Classical 101 - Deep Cuts	50
TV Shows	TV Shows	213
90's Music	Music	1477
90's Music	Heavy Metal Classic	5

Q.26 Cumulative Sales by Country and Month:

-- Calculate the cumulative total sales for each country, month by month.

→

```
select i.billingcountry,month(i.invoicedate) as invoice_month,sum(iv.unitprice * iv.quantity) as sales,
sum(sum(iv.unitprice * iv.quantity)) over (partition by i.billingcountry order by month(i.invoicedate)) as cumulative_sales
from customer c
join invoice i
on c.customerid = i.customerid
join invoiceline iv
on i.invoiceid = iv.invoiceid
group by i.billingcountry,month(i.invoicedate)
order by i.billingcountry,invoice_month;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

billingcountry	invoice_month	sales	cumulative_sales
Argentina	1	1.98	1.98
Argentina	3	13.86	15.84
Argentina	6	1.98	17.82
Argentina	8	0.99	18.81
Argentina	9	3.96	22.77
Argentina	11	8.91	31.68
Argentina	12	5.94	37.62
Australia	1	13.86	13.86
Australia	4	1.98	15.84
Australia	5	0.99	16.83
Australia	7	3.96	20.79
Australia	8	8.91	29.70
Australia	10	5.94	35.64
Australia	11	1.98	37.62
Austria	1	18.86	18.86
Austria	4	1.98	20.84
Austria	6	0.99	21.83

Q.27 Median Track Length by Genre:

-- For each genre, find the median track length (in milliseconds).

→

```
with track_rank as (
  select g.name as Genrename , t.milliseconds as Trk_lngth,
  row_number() over (partition by g.name order by t.milliseconds) as row_n,
  count(t.trackId) over (partition by g.name) as total_track
  from track t
  join genre g
  on t.genreid = g.genreid
  )
 SELECT
  Genrename, round(AVG(Trk_lngth),3) AS medianTrk
FROM
    track_rank
WHERE
    row_n IN ((total_track + 1) / 2 , (total_track + 2) / 2)
GROUP BY Genrename
ORDER BY Genrename;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Genrename	medianTrk
Alternative	240255.000
Alternative & Punk	230974.000
Blues	251219.000
Bossa Nova	204956.000
Classical	274504.000
Comedy	1302093.000
Drama	2610250.000
Easy Listening	188499.000
Electronica/Dance	304143.000
Heavy Metal	300956.000
Hip Hop/Rap	185103.000

Q.28 Customers Who Haven't Purchased a Specific Genre:

-- Find all customers (full name and email) who have never purchased a "Classical" music track.

→

```
SELECT
    CONCAT(c.firstname, ' ', c.lastname) AS cust_name, c.email
FROM
    customer c
        LEFT JOIN
    (SELECT DISTINCT
        i.customerid
    FROM
        invoice i
    JOIN invoiceline iv ON i.invoiceid = iv.invoiceid
    JOIN track t ON iv.trackid = t.trackid
    JOIN genre g ON t.genreid = g.genreid
    WHERE
        g.name = 'Classical') AS classic_cust
        ON c.customerid = classic_cust.customerid
WHERE
    classic_cust.customerid IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cust_name	email
Leonie Köhler	leonekohler@surfeu.de
František Wichterlová	frantisekw@jetbrains.com
Helena Holý	hholy@gmail.com
Daan Peeters	daan_peeters@apple.be
Kara Nielsen	kara.nielsen@jubii.dk
Eduardo Martins	eduardo@woodstock.com.br
Alexandre Rocha	alero@uol.com.br
Roberto Almeida	roberto.almeida@riotur.gov.br
Mark Philips	mphilips12@shaw.ca
Jennifer Peterson	jenniferp@rogers.ca

Q.29 Most Popular Media Type per Country:

-- Determine the most popular MediaType (based on total quantity sold) for each country.

→

```
select i.billingcountry as Country,m.name as MediaType,sum(iv.quantity) as total_quantity,
rank() over (partition by i.billingcountry order by sum(iv.quantity)desc) as rank_n
from mediatype m
join track t
on m.mediatypeid = t.mediatypeid
join invoiceline iv
on t.trackid = iv.trackid
join invoice i
on iv.invoiceid = i.invoiceid
group by i.billingcountry,m.name
order by i.billingcountry,total_quantity desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Country	MediaType	total_quantity	rank_n
Argentina	MPEG audio file	36	1
Argentina	Protected AAC audio file	2	2
Australia	MPEG audio file	38	1
Austria	MPEG audio file	25	1
Austria	Protected AAC audio file	8	2
Austria	Protected MPEG-4 video file	5	3
Belgium	MPEG audio file	38	1
Brazil	MPEG audio file	170	1
Brazil	Protected AAC audio file	18	2
Brazil	Protected MPEG-4 video file	2	3
Canada	MPEG audio file	289	1
Canada	Protected AAC audio file	12	2
Canada	Protected MPEG-4 video file	3	3
Chile	MPEG audio file	28	1
Chile	Protected MPEG-4 video file	9	2
Chile	Protected AAC audio file	1	3
Czech R...	MPEG audio file	57	1

Q.30 Artist with Most Albums Without Any Tracks Sold:

-- Find the artist(s) who have the most albums in the database, but none of their tracks have ever been sold.

→

```sql
with total_album_by_artist as (
    select ar.artistid,ar.name as Artist_Name,count(a.albumid) as total_album
    from artist ar
    join album a
    on ar.artistid = a.artistid
    group by ar.artistid,ar.name
    order by total_album desc
),
album_sold_track as (
    select distinct ar.artistid
    from artist ar
    join album a
    on ar.artistid = a.artistid
    join track t
    on a.albumid = t.albumid
    join invoiceline iv
    on t.trackid = iv.trackid
)
SELECT
    tab.Artist_Name
FROM
    total_album_by_artist tab
        LEFT JOIN
    album_sold_track ast ON tab.artistid = ast.artistid
WHERE
    ast.artistid IS NULL
ORDER BY tab.total_album DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Artist_Name
Cake
Aisha Duo
Karsh Kale
Aaron Goldberg
Nicolaus Esterhazy Sinfonia

Q.31 Churn Prediction - Customers at Risk:

-- Identify customers who made their first purchase more than 6 months ago, but haven't made any purchases in the last 3 months. For these customers, list their full name, their last purchase date, and the total number of days since their last purchase.

→

```
SELECT
    CONCAT(c.firstname, ' ', c.lastname) AS cust_name,
    DATE(MAX(i.invoicedate)) AS last_purchase,
    DATEDIFF(CURRENT_DATE(), DATE(MAX(i.invoicedate))) AS day_since_lp
FROM
    customer c
        JOIN
    invoice i ON c.customerID = i.customerid
GROUP BY cust_name
HAVING date(min(i.invoicedate)) < DATE_SUB(CURRENT_DATE(),
    INTERVAL 6 MONTH)
    AND MONTH(last_purchase) < DATE_SUB(CURRENT_DATE(),
    INTERVAL 3 MONTH)
ORDER BY day_since_lp DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cust_name	last_purchase	day_since_lp
Puja Srivastava	2024-05-30	406
Niklas Schröder	2024-06-30	375
Leonie Köhler	2024-07-13	362
Jack Smith	2024-07-31	344
Dominique Lefebvre	2024-08-13	331
Mark Taylor	2024-08-31	313
Tim Goyer	2024-09-13	300
João Fernandes	2024-10-01	282
Luis Rojas	2024-10-14	269
Fernanda Ramos	2024-11-01	251
Hannah Schneider	2024-11-14	238
Joakim Johansson	2024-12-02	220
Jennifer Peterson	2024-12-15	207

Q.32 Genre Popularity Shift Over Time (Quarterly Analysis):

-- For each quarter of each year present in the data, determine the top 3 most popular genres by total sales revenue. Display the year, quarter, genre name, and the total revenue for that genre in that quarter.

→

```
with Quarterly_revenue as (
select distinct g.name as Genre_name,year(i.invoicedate) as sales_year,quarter(i.invoicedate) as sales_quarter ,sum(iv.unitprice * iv.quantity) as total_sales_revenue,
rank() over(partition by year(i.invoicedate),quarter(i.invoicedate) order by sum(iv.unitprice * iv.quantity) desc) as rank_no
from genre g
join track t
on g.genreid = t.genreid
join invoiceline iv
on t.trackid = iv.trackid
join invoice i
on iv.invoiceid = i.invoiceid
group by Genre_name, sales_year,sales_quarter
)
SELECT
    Genre_name, sales_year, sales_quarter, total_sales_revenue
FROM
    Quarterly_revenue
WHERE
    rank_no <= 3
ORDER BY sales_year , sales_quarter , total_sales_revenue DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Genre_name	sales_year	sales_quarter	total_sales_revenue
Rock	2021	1	32.67
Latin	2021	1	32.67
Alternative & Punk	2021	1	14.85
Rock	2021	2	45.54
Latin	2021	2	19.80
Alternative & Punk	2021	2	16.83
Rock	2021	3	43.56
Metal	2021	3	28.71

Q.33 Employee Efficiency - Average Time to Close an Invoice:

-- For each sales support agent (employee), calculate the average time (in days) between the Invoice Date and the date the invoice was paid (assume Billing Country is a proxy for "closed" if Invoice Date is the only date field). If no 'paid' date exists, assume its CURRENT_DATE. Also, show the total revenue generated by each agent.

→

```
SELECT
    e.employeeid,
    e.firstname,
    e.lastname,
    AVG(DATEDIFF(CURRENT_DATE(), DATE(i.invoicedate))) AS avg_time_taken,
    SUM(i.total) AS total_revenue
FROM
    employee e
        JOIN
    customer c ON e.employeeid = c.supportrepid
        JOIN
    invoice i ON c.customerid = i.customerid
WHERE
    e.title = 'Sales Support Agent'
GROUP BY e.employeeid , e.firstname , e.lastname;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employeeid	firstname	lastname	avg_time_taken	total_revenue
3	Jane	Peacock	720.4932	833.04
4	Margaret	Park	756.3786	775.40
5	Steve	Johnson	759.6508	720.16

Q.34 Most Diverse Customer Portfolios:

-- Identify the top 5 customers who have purchased tracks from the most unique genres. For each of these customers, list their full name, and the count of unique genres they've purchased from.

→

```sql
SELECT
    CONCAT(c.firstname, ' ', c.lastname) AS cust_name,
    COUNT(DISTINCT g.genreid) AS unique_genre
FROM
    customer c
        JOIN
    invoice i ON c.customerid = i.customerid
        JOIN
    invoiceline iv ON i.invoiceid = iv.invoiceid
        JOIN
    track t ON iv.trackid = t.trackid
        JOIN
    genre g ON t.genreid = g.genreid
GROUP BY cust_name
ORDER BY unique_genre DESC
limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cust_name	unique_genre
Luis Rojas	12
Ladislav Kovács	11
Jack Smith	10
François Tremblay	10
João Fernandes	10

Q.35 Artist-to-Artist Collaboration Network (Indirect Sales Influence):

-- Identify pairs of artists where a customer who purchased a track from Artist A also purchased a track from Artist B, and these two artists are not directly related by having tracks on the same album. Quantify the "collaboration" by the number of unique customers who bought from both. Order the results by the collaboration count in descending order.

→

```sql
with cutomer_artist as (
    select distinct c.customerid,ar.artistid,ar.name as Artist_Name
    from customer c
    join invoice i
    on c.customerid = i.customerid
    join invoiceline iv
    on i.invoiceid = iv.invoiceid
    join track t
    on iv.trackid = t.trackid
    join album a
    on t.albumid = a.albumid
    join artist ar
    on a.artistid = ar.artistid
    ),
artist_album as (
    select distinct a.albumid,a.artistid
    from album a
    join track t
    on a.albumid = t.albumid
    where a.artistid is not null
    ),
artist_share_album as (
    select aa1.artistid as ArtistId1,aa2.artistid as ArtistId2
    from artist_album aa1
    join artist_album aa2
    on aa1.albumid = aa2.albumid
    and aa1.artistid < aa2.artistid
    )
```

```
SELECT
    ar1.name AS ArtistA,
    ar2.name AS ArtistB,
    COUNT(DISTINCT ca1.customerid) AS c_count
FROM
    cutomer_artist ca1
        JOIN
    cutomer_artist ca2 ON ca1.customerid = ca2.customerid
        AND ca1.artistid < ca2.artistid
        JOIN
    artist ar1 ON ca1.artistid = ar1.artistid
        JOIN
    artist ar2 ON ca2.artistid = ar2.artistid
        LEFT JOIN
    artist_share_album asa ON (ca1.artistid = asa.artistid1
        AND ca2.artistid = asa.artistid2)
        OR (ca1.artistid = asa.artistid2
        AND ca2.artistid = asa.artistid1)
WHERE
    asa.artistid1 IS NULL
GROUP BY ar1.name , ar2.name
ORDER BY c_count DESC , ArtistA , ArtistB
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ArtistA	ArtistB	c_count
Led Zeppelin	U2	14
Iron Maiden	U2	12
Metallica	Iron Maiden	12
Iron Maiden	Lenny Kravitz	11
Iron Maiden	R.E.M.	10
Led Zeppelin	Iron Maiden	10
Led Zeppelin	Os Paralamas Do Sucesso	10
Metallica	U2	10

Q.36 Impact of Track Length on Sales:

-- Divide tracks into 5 equal length 'quintiles'. For each quintile, calculate the total sales revenue and the average number of times tracks from that quintile were purchased. `

→

```sql
with Track_len_q as (
    select t.trackid,t.milliseconds,
    ntile(5) over (order by t.milliseconds) as len_quintile
    from track t
    ),
    track_purchase as (
    select iv.trackid,count(iv.invoicelineid) as purchase_count_track
    from invoiceline iv
    group by iv.trackid
    )
    SELECT
    tlq.len_quintile,
    SUM(iv.unitprice * iv.quantity) AS total_revenue,
    AVG(tp.purchase_count_track) AS avg_purchase_count_track
FROM
    Track_len_q tlq
        JOIN
    invoiceline iv ON tlq.trackid = iv.trackid
        LEFT JOIN
    track_purchase tp ON tlq.trackid = tp.trackid
GROUP BY tlq.len_quintile
ORDER BY tlq.len_quintile;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

len_quintile	total_revenue	avg_purchase_count_track
1	459.36	1.2759
2	432.63	1.2059
3	433.62	1.2192
4	454.41	1.2353
5	548.58	1.2036

Report:

"To better understand the operational dynamics of our simulated digital media store, this report utilizes SQL to query the Chinook database. Our objective is to identify top-performing areas and potential opportunities for growth."

Insights:

- **Most Selling Genre**: -

  We can identify Popularity of Genre and most selling genre is “Rock”

- **Top Selling Tracks**: -

  Top selling Tracks are “Dazed and Confused”, “The Trooper”, “The Fix”, etc.

- **Top 3 Employee as Sales representative**: -

  Top 3 Employees are “Jane Peacock”, “Margret Park”, “Steve Johnson”

- **Monthly Trend Analysis**: -

  We can see that Sales are increasing after 1st month.

- **Employee Tenure**: -

  Most senior employee is “Nancy Edwards” who is working for 23 years.

- **Artist with no Track sold**: -

  Artists whose track have not sold are “AC/DC”, “Aerosmith”, “Alice in chains” etc.

- **Most Popular Artists**: -

  Most Popular Artists are “Iron Maiden”, “U2”, “Lost” etc.

- **<u>Most Popular MediaType by Country</u>**: -

  In Argentina “MPEG audio file”, “Protected AAC audio file”,

  In Australia “MPEG audio file”,

  These are the Popular media types in each country.

- **<u>Customer at Risk</u>**: -

  “Pooja Srivastav”, “Niclas Schröder”, “Leonie Köhler”, these are one of the customers who made their last purchase 350 days ago and losing interest in buying.

- **<u>Most valuable Customer</u>**: -

  “Loise Rojas”, “Ladislav Kovacs”, “Jack Smith”

  These are one of the most valuable customers as they have most diverse portfolio.