SQL Project Report: Music Store Data Exploration Using SQL

Overview

This project explores the **Chinook Database**, a sample database representing a digital music store. It includes tables like Customer, Invoice, Track, Artist, Album, and more. The goal is to analyze sales, customer behavior, and music product insights using **SQL**.

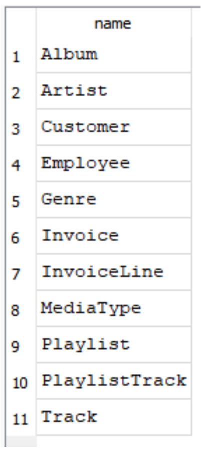
Database Link: https://github.com/lerocha/chinook-database

Software used DB Browser for SQlite, download link: https://sqlitebrowser.org/dl/

Table Exploration

List All Tables in the Database

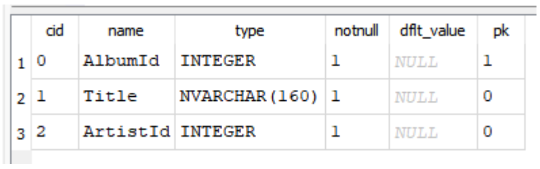
SELECT name FROM sqlite_master WHERE type='table';



	name
1	Album
2	Artist
3	Customer
4	Employee
5	Genre
6	Invoice
7	InvoiceLine
8	MediaType
9	Playlist
10	PlaylistTrack
11	Track

Table Structure (Schema)

PRAGMA table_info(Album);



	cid	name	type	notnull	dflt_value	pk
1	0	AlbumId	INTEGER	1	NULL	1
2	1	Title	NVARCHAR(160)	1	NULL	0
3	2	ArtistId	INTEGER	1	NULL	0

**Similar output will be displayed for every table.

PRAGMA table_info(Artist);

PRAGMA table_info(Customer);

PRAGMA table_info(Employee);

PRAGMA table_info(Genre);

PRAGMA table_info(Invoice);

PRAGMA table_info(InvoiceLine);

PRAGMA table_info(MediaType);

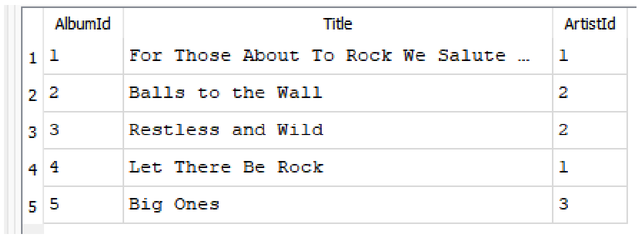
PRAGMA table_info(Playlist);

PRAGMA table_info(PlaylistTrack);

PRAGMA table_info(Track);

Sample Data Preview

SELECT * FROM Album LIMIT 5;



	AlbumId	Title	ArtistId
1	1	For Those About To Rock We Salute …	1
2	2	Balls to the Wall	2
3	3	Restless and Wild	2
4	4	Let There Be Rock	1
5	5	Big Ones	3

**Similar output will be displayed for every table.

SELECT * FROM Customer LIMIT 5;

SELECT * FROM Artist LIMIT 5;

SELECT * FROM Employee LIMIT 5;

SELECT * FROM Genre LIMIT 5;

SELECT * FROM Invoice LIMIT 5;

SELECT * FROM InvoiceLine LIMIT 5;

SELECT * FROM MediaType LIMIT 5;

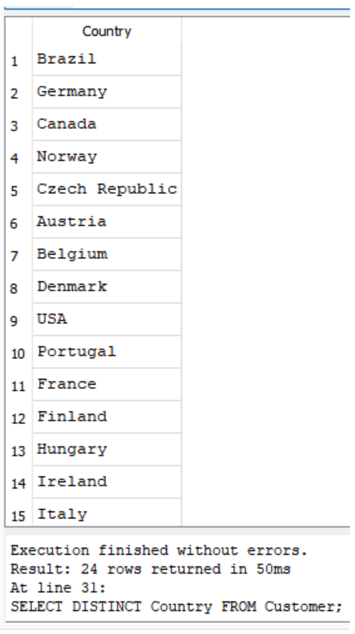
SELECT * FROM Playlist LIMIT 5;

SELECT * FROM Track LIMIT 5;

Basic Data Analysis Queries:

List of Countries (Customers)

SELECT DISTINCT Country FROM Customer;



	Country
1	Brazil
2	Germany
3	Canada
4	Norway
5	Czech Republic
6	Austria
7	Belgium
8	Denmark
9	USA
10	Portugal
11	France
12	Finland
13	Hungary
14	Ireland
15	Italy

```
Execution finished without errors.
Result: 24 rows returned in 50ms
At line 31:
SELECT DISTINCT Country FROM Customer;
```

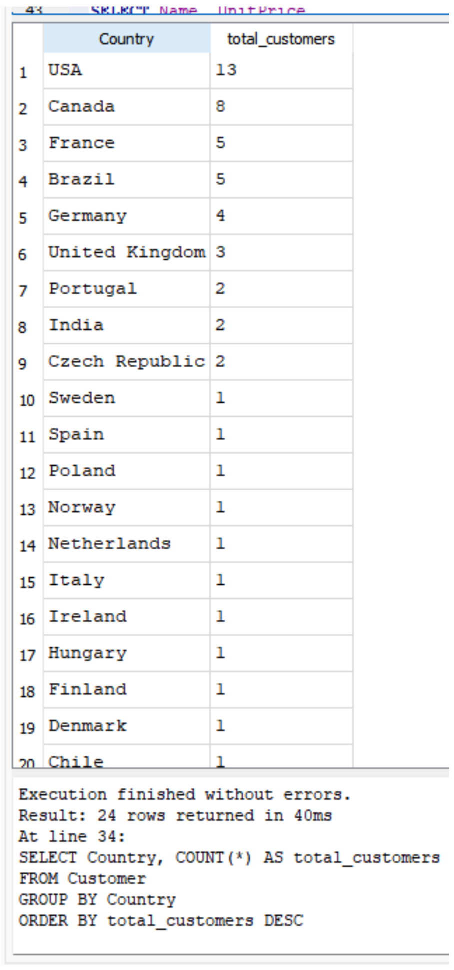
Number of Customers per Country

SELECT Country, COUNT(*) AS total_customers

FROM Customer

GROUP BY Country

ORDER BY total_customers DESC;

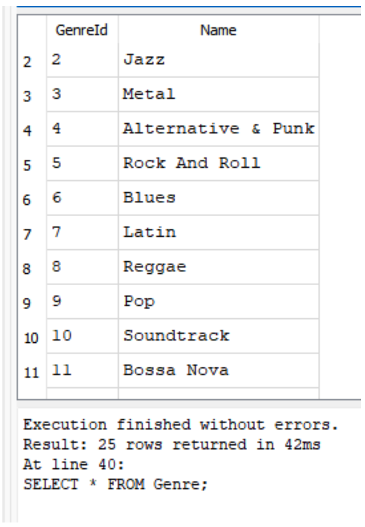


	Country	total_customers
1	USA	13
2	Canada	8
3	France	5
4	Brazil	5
5	Germany	4
6	United Kingdom	3
7	Portugal	2
8	India	2
9	Czech Republic	2
10	Sweden	1
11	Spain	1
12	Poland	1
13	Norway	1
14	Netherlands	1
15	Italy	1
16	Ireland	1
17	Hungary	1
18	Finland	1
19	Denmark	1
20	Chile	1

```
Execution finished without errors.
Result: 24 rows returned in 40ms
At line 34:
SELECT Country, COUNT(*) AS total_customers
FROM Customer
GROUP BY Country
ORDER BY total_customers DESC
```

List of Genres Available

SELECT * FROM Genre;



	GenreId	Name
2	2	Jazz
3	3	Metal
4	4	Alternative & Punk
5	5	Rock And Roll
6	6	Blues
7	7	Latin
8	8	Reggae
9	9	Pop
10	10	Soundtrack
11	11	Bossa Nova

```
Execution finished without errors.
Result: 25 rows returned in 42ms
At line 40:
SELECT * FROM Genre;
```

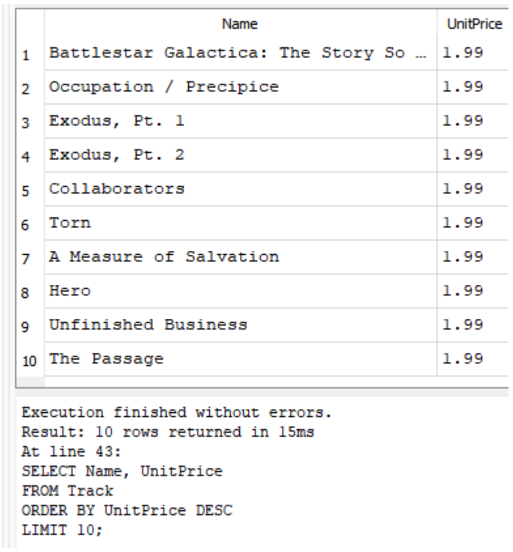
Top 10 Most Expensive Tracks

SELECT Name, UnitPrice

FROM Track

ORDER BY UnitPrice DESC

LIMIT 10;



	Name	UnitPrice
1	Battlestar Galactica: The Story So ...	1.99
2	Occupation / Precipice	1.99
3	Exodus, Pt. 1	1.99
4	Exodus, Pt. 2	1.99
5	Collaborators	1.99
6	Torn	1.99
7	A Measure of Salvation	1.99
8	Hero	1.99
9	Unfinished Business	1.99
10	The Passage	1.99

```
Execution finished without errors.
Result: 10 rows returned in 15ms
At line 43:
SELECT Name, UnitPrice
FROM Track
ORDER BY UnitPrice DESC
LIMIT 10;
```

Join-Based Business Insights

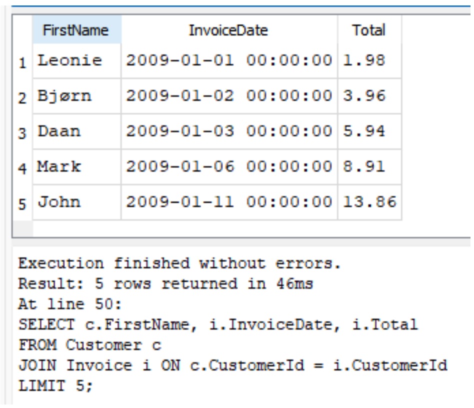
Customer Invoice Relationship (Sample Join)

SELECT c.FirstName, i.InvoiceDate, i.Total

FROM Customer c

JOIN Invoice i ON c.CustomerId = i.CustomerId

LIMIT 5;

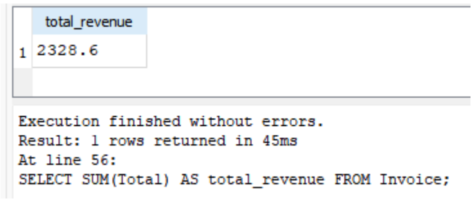


	FirstName	InvoiceDate	Total
1	Leonie	2009-01-01 00:00:00	1.98
2	Bjørn	2009-01-02 00:00:00	3.96
3	Daan	2009-01-03 00:00:00	5.94
4	Mark	2009-01-06 00:00:00	8.91
5	John	2009-01-11 00:00:00	13.86

```
Execution finished without errors.
Result: 5 rows returned in 46ms
At line 50:
SELECT c.FirstName, i.InvoiceDate, i.Total
FROM Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
LIMIT 5;
```

Total Revenue Generated

SELECT SUM(Total) AS total_revenue FROM Invoice;



	total_revenue
1	2328.6

```
Execution finished without errors.
Result: 1 rows returned in 45ms
At line 56:
SELECT SUM(Total) AS total_revenue FROM Invoice;
```

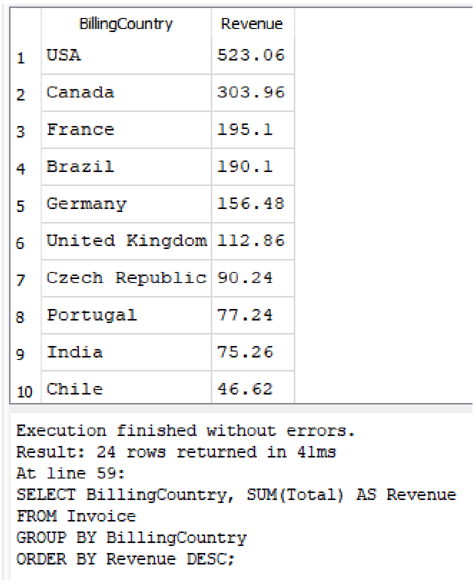
Total Revenue Per Country

SELECT BillingCountry, SUM(Total) AS Revenue

FROM Invoice

GROUP BY BillingCountry

ORDER BY Revenue DESC;



	BillingCountry	Revenue
1	USA	523.06
2	Canada	303.96
3	France	195.1
4	Brazil	190.1
5	Germany	156.48
6	United Kingdom	112.86
7	Czech Republic	90.24
8	Portugal	77.24
9	India	75.26
10	Chile	46.62

```
Execution finished without errors.
Result: 24 rows returned in 41ms
At line 59:
SELECT BillingCountry, SUM(Total) AS Revenue
FROM Invoice
GROUP BY BillingCountry
ORDER BY Revenue DESC;
```

Top 5 Best-Selling Tracks

SELECT t.Name, COUNT(il.TrackId) AS TimesSold

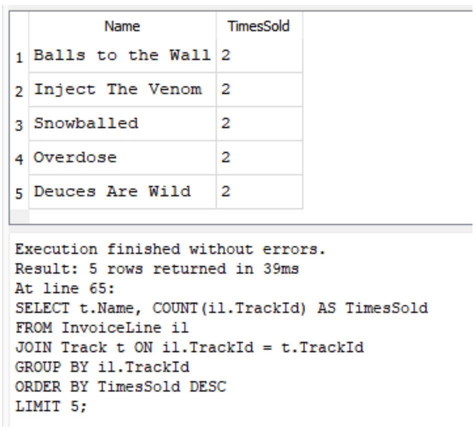
FROM InvoiceLine il

JOIN Track t ON il.TrackId = t.TrackId

GROUP BY il.TrackId

ORDER BY TimesSold DESC

LIMIT 5;



	Name	TimesSold
1	Balls to the Wall	2
2	Inject The Venom	2
3	Snowballed	2
4	Overdose	2
5	Deuces Are Wild	2

```
Execution finished without errors.
Result: 5 rows returned in 39ms
At line 65:
SELECT t.Name, COUNT(il.TrackId) AS TimesSold
FROM InvoiceLine il
JOIN Track t ON il.TrackId = t.TrackId
GROUP BY il.TrackId
ORDER BY TimesSold DESC
LIMIT 5;
```

Average Order Value per Customer

SELECT c.FirstName || ' ' || c.LastName AS CustomerName,

AVG(i.Total) AS AvgInvoiceAmount

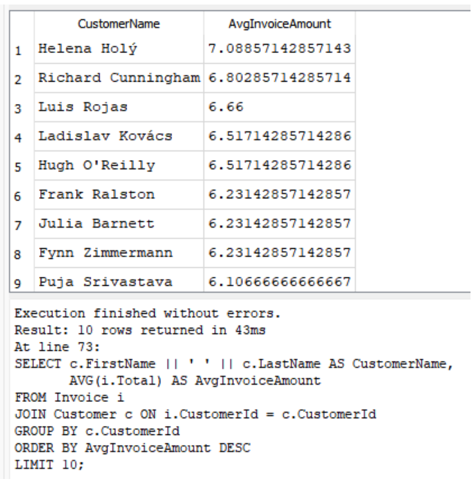
FROM Invoice i

JOIN Customer c ON i.CustomerId = c.CustomerId

GROUP BY c.CustomerId

ORDER BY AvgInvoiceAmount DESC

LIMIT 10;



	CustomerName	AvgInvoiceAmount
1	Helena Holý	7.08857142857143
2	Richard Cunningham	6.80285714285714
3	Luis Rojas	6.66
4	Ladislav Kovács	6.51714285714286
5	Hugh O'Reilly	6.51714285714286
6	Frank Ralston	6.23142857142857
7	Julia Barnett	6.23142857142857
8	Fynn Zimmermann	6.23142857142857
9	Puja Srivastava	6.10666666666667

```
Execution finished without errors.
Result: 10 rows returned in 43ms
At line 73:
SELECT c.FirstName || ' ' || c.LastName AS CustomerName,
       AVG(i.Total) AS AvgInvoiceAmount
FROM Invoice i
JOIN Customer c ON i.CustomerId = c.CustomerId
GROUP BY c.CustomerId
ORDER BY AvgInvoiceAmount DESC
LIMIT 10;
```

Top 5 Customers by Spending

SELECT Customer.FirstName || ' ' || Customer.LastName AS CustomerName, SUM(Invoice.Total) AS AmountSpent

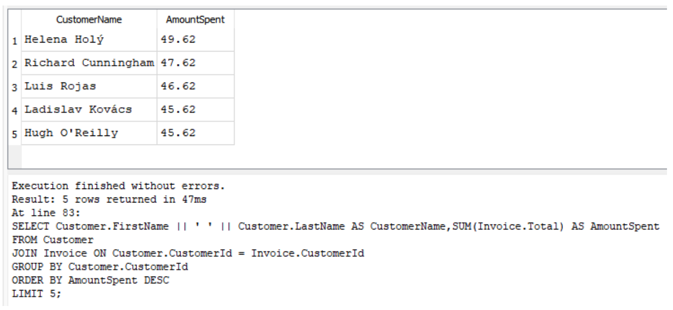
FROM Customer

JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId

GROUP BY Customer.CustomerId

ORDER BY AmountSpent DESC

LIMIT 5;



	CustomerName	AmountSpent
1	Helena Holý	49.62
2	Richard Cunningham	47.62
3	Luis Rojas	46.62
4	Ladislav Kovács	45.62
5	Hugh O'Reilly	45.62

```
Execution finished without errors.
Result: 5 rows returned in 47ms
At line 83:
SELECT Customer.FirstName || ' ' || Customer.LastName AS CustomerName,SUM(Invoice.Total) AS AmountSpent
FROM Customer
JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY Customer.CustomerId
ORDER BY AmountSpent DESC
LIMIT 5;
```

Time-Based and Advanced Queries

Monthly Revenue Report

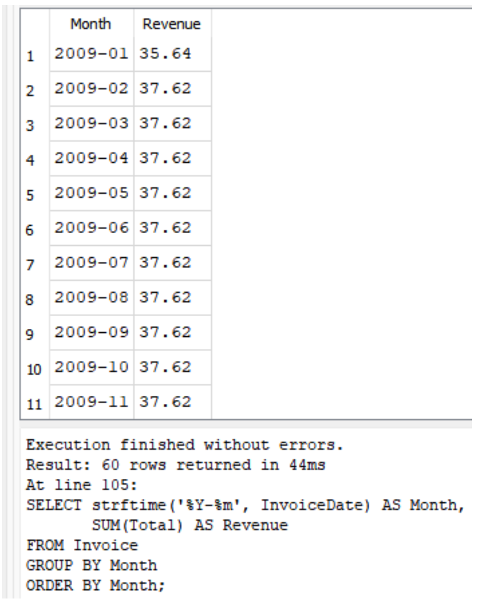
SELECT strftime('%Y-%m', InvoiceDate) AS Month,

SUM(Total) AS Revenue

FROM Invoice

GROUP BY Month

ORDER BY Month;



	Month	Revenue
1	2009-01	35.64
2	2009-02	37.62
3	2009-03	37.62
4	2009-04	37.62
5	2009-05	37.62
6	2009-06	37.62
7	2009-07	37.62
8	2009-08	37.62
9	2009-09	37.62
10	2009-10	37.62
11	2009-11	37.62

```
Execution finished without errors.
Result: 60 rows returned in 44ms
At line 105:
SELECT strftime('%Y-%m', InvoiceDate) AS Month,
       SUM(Total) AS Revenue
FROM Invoice
GROUP BY Month
ORDER BY Month;
```

Revenue Classification by Country

SELECT BillingCountry,

SUM(Total) AS Revenue,

CASE

WHEN SUM(Total) > 100 THEN 'High'

WHEN SUM(Total) > 50 THEN 'Medium'

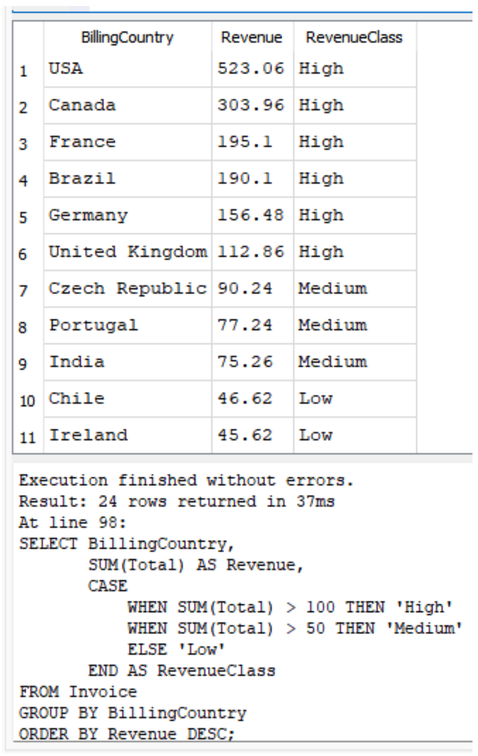
ELSE 'Low'

END AS RevenueClass

FROM Invoice

GROUP BY BillingCountry

ORDER BY Revenue DESC;



	BillingCountry	Revenue	RevenueClass
1	USA	523.06	High
2	Canada	303.96	High
3	France	195.1	High
4	Brazil	190.1	High
5	Germany	156.48	High
6	United Kingdom	112.86	High
7	Czech Republic	90.24	Medium
8	Portugal	77.24	Medium
9	India	75.26	Medium
10	Chile	46.62	Low
11	Ireland	45.62	Low

```
Execution finished without errors.
Result: 24 rows returned in 37ms
At line 98:
SELECT BillingCountry,
       SUM(Total) AS Revenue,
       CASE
           WHEN SUM(Total) > 100 THEN 'High'
           WHEN SUM(Total) > 50 THEN 'Medium'
           ELSE 'Low'
       END AS RevenueClass
FROM Invoice
GROUP BY BillingCountry
ORDER BY Revenue DESC;
```