Relationships

Tiny blobs, medium blobs, purge, binary identities, fakers, floaters and other super unserious SQL operators

MINERVA
FRANCO

PROJECT 3
29 OCT 2024

Table of Contents

I. **Introduction:** A Timeline

II. **Complex Joins:** Uniting Your Data

III. **Business Question 1:** Predictive Models

IV. **Business Question 2:** Employee Performance Analysis

V. **Business Question 3:** Track Length Analysis

VI. **Business Question 4:** Sales Prediction and Revenue Optimization

NEXT

I. Timeline

TUESDAY WED THURS FRI SAT SUN MON BOOM HUH HELLO WHAT PROJECT

DATABASE: CHINOOK

11 Tables, 69 Columns

8 Employees, 59 Customers

Data collected from Jan 2009 thru Dec 2013

24 Countries, USA has the most customers (13)

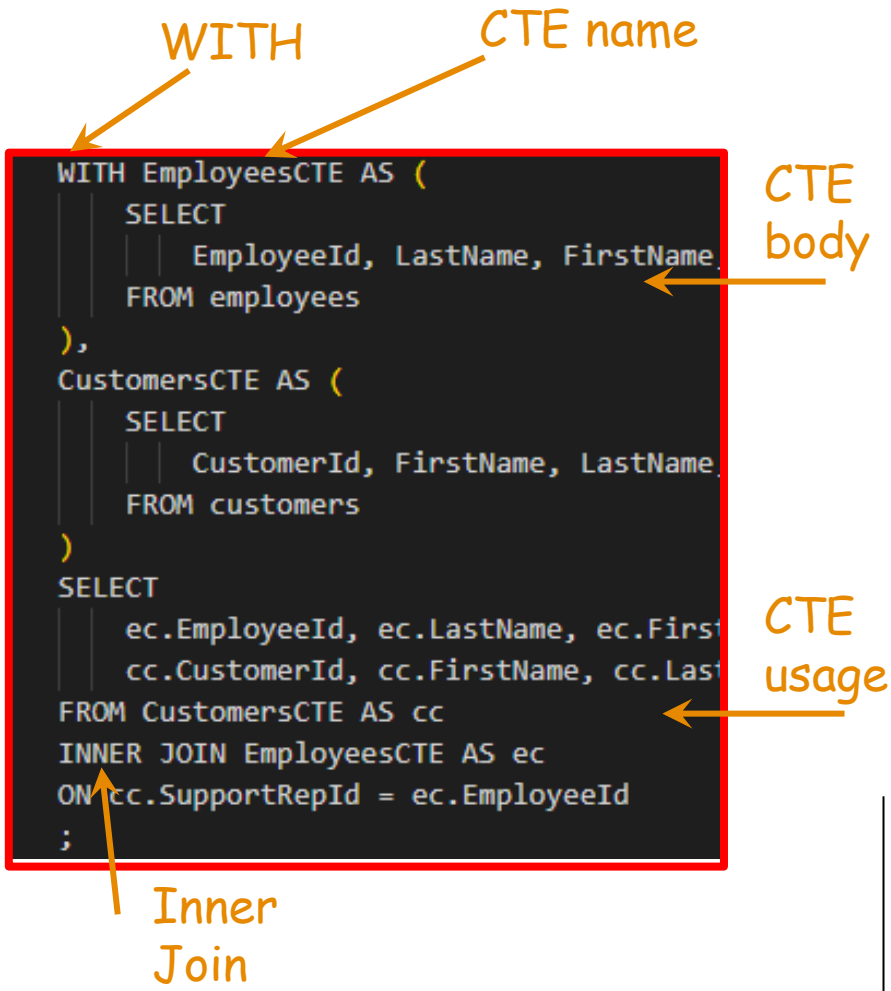
NEXT

II.Complex Joins: Uniting Your Data

Complex joins are essential for combining information from related tables.

CTEs (Common Table Expressions) make complex joins more readable.

INNER JOIN: Returns only matching rows from both tables.




```
WITH EmployeesCTE AS (
    SELECT
        EmployeeId, LastName, FirstName
    FROM employees
),
CustomersCTE AS (
    SELECT
        CustomerId, FirstName, LastName
    FROM customers
)
SELECT
    ec.EmployeeId, ec.LastName, ec.First
    cc.CustomerId, cc.FirstName, cc.Last
FROM CustomersCTE AS cc
INNER JOIN EmployeesCTE AS ec
ON cc.SupportRepId = ec.EmployeeId
;
```

NEXT

III. Predictive Models: Looking Ahead

Business need: Predict which genres and media types will sell best to plan next album launches.

Question: How do I access data on genres and media types & determine if features are predictive?

```
# # Execute the SQL query and load data into a DataFrame
query = """
SELECT
    t.TrackId,
    t.Name AS TrackName,
    g.Name AS Genre,
    mt.Name AS MediaType,
    SUM(ii.Quantity) AS TotalSales
FROM
    tracks AS t
JOIN
    genres AS g ON t.GenreId =
JOIN
    media_types AS mt ON t.Med
JOIN
    invoice_items AS ii ON t.T
GROUP BY
    t.TrackId, g.Name, mt.Name
ORDER BY
    TotalSales DESC
;
"""
df = pd.read_sql_query(query, conn)
conn.close()
```

Data Extraction: Used 3 joins to gather data from 4 tables.

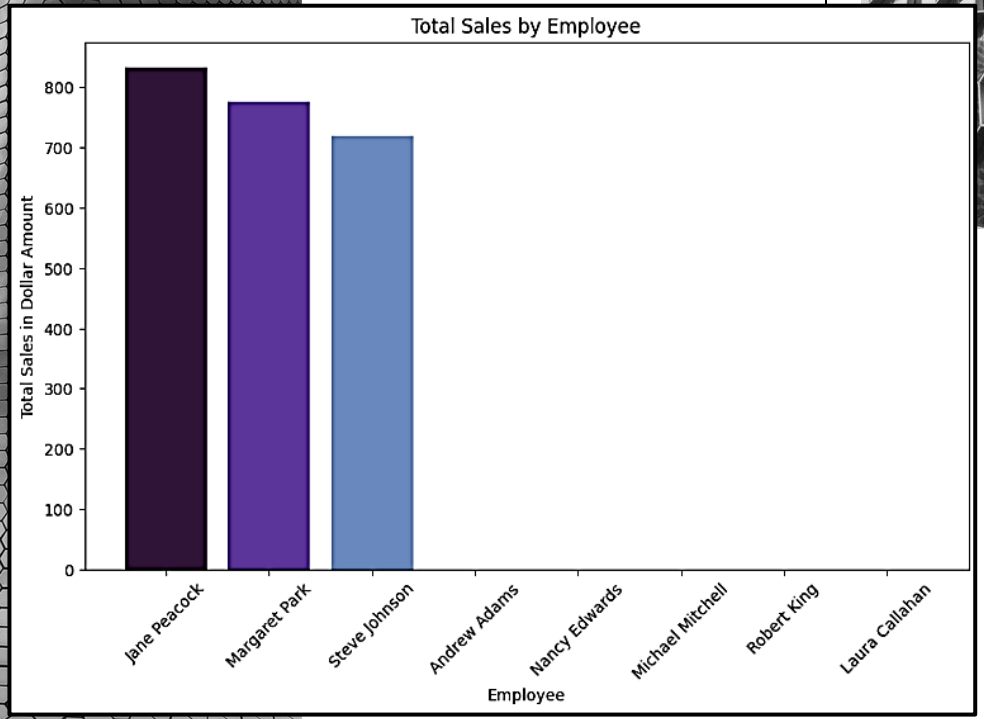
Model: Random Forest Regressor does not require one-hot encoding categorical variables (genre, media type)

Outcome: The model's performance is evaluated using Mean Squared Error (MSE)=0.14 indicating genre & media type can be good indicators of sales.

NEXT

IV. Employee Performance Analysis:

All NPCs , no squad.



Business Need: Understanding which employees generate the most sales to recognize top performers and inform training / incentive programs.

Question: How do I calculate total sales for employees?

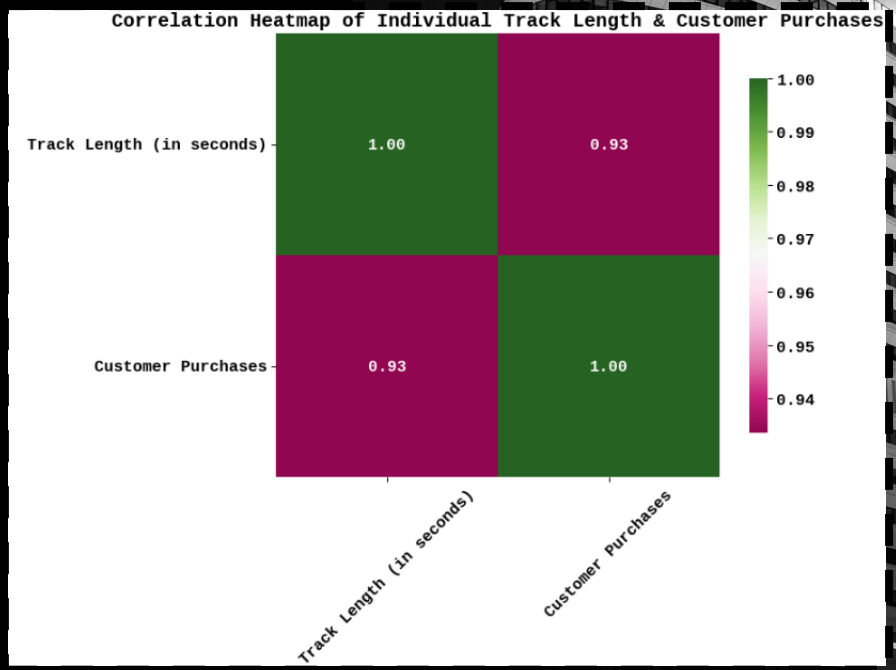
Method: Left joined 3 tables to aggregate sales data by employee, showing how much each employee has sold based on the invoices associated with customers they support.

NEXT

V: Track Length Analysis:
Correlation Station

Business need: Analyze track length preferences to understand customer engagement and guide music production strategies

Question: How do I retrieve a list of tracks along with the distribution of track length across different music genres?



NEXT

Thank you for listening

Questions?

NEXT

Credits

https://www.sqlitetutorial.net/wp-content/uploads/2018/03/chinook.zip

https://www.collidu.com/presentation-sql-joins

Follow me for more dating tips.

NEXT

II.Complex Joins: Uniting Your Data

```
%%script sqlite3 --column --header chinook.db
WITH EmployeesCTE AS (
    SELECT
        EmployeeId, LastName, FirstName, Title, ReportsTo
    FROM employees
),
CustomersCTE AS (
    SELECT
        CustomerId, FirstName, LastName, SupportRepId
    FROM customers
)
SELECT
    ec.EmployeeId, ec.LastName, ec.FirstName, ec.Title, ec.ReportsTo,
    cc.CustomerId, cc.FirstName, cc.LastName
FROM CustomersCTE AS cc
INNER JOIN EmployeesCTE AS ec
ON cc.SupportRepId = ec.EmployeeId
;
```

III. Predictive Models: Looking Ahead

```
# # Execute the SQL query and load data into a DataFrame
query = """
SELECT
    t.TrackId,
    t.Name AS TrackName,
    g.Name AS Genre,
    mt.Name AS MediaType,
    SUM(ii.Quantity) AS TotalSales
FROM
    tracks AS t
JOIN
    genres AS g ON t.GenreId = g.GenreId
JOIN
    media_types AS mt ON t.MediaTypeId = mt.MediaTypeId
JOIN
    invoice_items AS ii ON t.TrackId = ii.TrackId
GROUP BY
    t.TrackId, g.Name, mt.Name
ORDER BY
    TotalSales DESC
;
"""
df = pd.read_sql_query(query, conn)
conn.close()
```

IV. Employee Performance Analysis:

All NPCs , no squad.

```
query = """
SELECT
    Employees.EmployeeId,
    Employees.FirstName,
    Employees.LastName,
    SUM(invoices.Total) AS TotalSales

FROM Employees
LEFT JOIN customers ON EmployeeId = customers
LEFT JOIN invoices ON customers.CustomerId = 
GROUP BY Employees.EmployeeId, Employees.Firs
ORDER BY TotalSales DESC;
"""

# Execute query and store results in a DataFr
df = pd.read_sql_query(query, conn)
```

V: Track Length Analysis: Correlation Station

```
# Query to get track lengths and invoice item costs
query = '''
SELECT
    t.TrackId,
    t.Milliseconds / 1000.0 AS TrackLengthSeconds,
    ii.UnitPrice AS InvoiceItemCost
FROM
    tracks AS t
JOIN
    invoice_items AS ii ON t.TrackId = ii.TrackId
JOIN
    invoices AS i ON ii.InvoiceId = i.InvoiceId;
'''

# Load the results into a DataFrame
df_tracks = pd.read_sql_query(query, conn)

# Close the database connection
conn.close()

# Calculate the correlation between Track Length and Invoice Item Cost
correlation = df_tracks['TrackLengthSeconds'].corr(df_tracks['InvoiceItemCost'])
print("Correlation between Track Length (in seconds) and Invoice Item Cost:",
correlation)
```

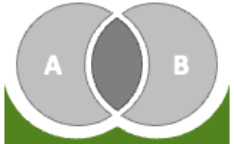
https://www.collidu.com/presentation-sql-joins

SQL JOINS

SELECT * FROM TableA a **LEFT JOIN** TableB b ON a.Key = b.Key

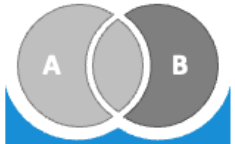
SELECT * FROM TableA a **RIGHT JOIN** TableB b On a.Key= b.Key

SELECT * FROM TableA a **FULL OUTER JOIN** TableB b ON a.Key =b.Key



SELECT *FROM TableA a **INNER JOIN** TableB b ON a.Key= b.Key

SELECT * FROM TableA a **LEFT JOIN** TableB b ON a.Key = b.Key WHERE b.Key IS NULL



SELECT * FROM TableA a **RIGHT JOIN** TableB b ON a.Key= b.Key WHERE a.Key IS NULL

SELECT * FROM TableA a **FULL OUTER JOIN** TableB ON a.Key = b.Key WHERE a.Key IS NULL OR b.Key IS NULL

NEXT

Meet Our Team

NAME SURNAME
Title

NAME SURNAME
Title

NAME SURNAME
Title