[title] NVARCHAR(160) NOT NULL, album [artist_id] INTEGER NOT NULL, FOREIGN KEY ([artist_id]) REFERENCES [artist] ([artist_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [artist] [artist_id] INTEGER PRIMARY KEY NOT NULL, artist [name] NVARCHAR(120) **CREATE TABLE [customer]** [customer_id] INTEGER PRIMARY KEY NOT NULL, [first_name] NVARCHAR(40) NOT NULL, [last_name] NVARCHAR(20) NOT NULL, [company] NVARCHAR(80), [address] NVARCHAR(70), [city] NVARCHAR(40), [state] NVARCHAR(40), customer [country] NVARCHAR(40), [postal_code] NVARCHAR(10), [phone] NVARCHAR(24), [fax] NVARCHAR(24), [email] NVARCHAR(60) NOT NULL [support_rep_id] INTEGER, FOREIGN KEY ([support_rep_id]) REFERENCES [employee] ([employee_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [employee] [employee_id] INTEGER PRIMARY KEY NOT NULL, [last_name] NVARCHAR(20) NOT NULL, [first_name] NVARCHAR(20) NOT NULL, [title] NVARCHAR(30), [reports_to] INTEGER, [birthdate] DATETIME, [hire_date] DATETIME, [address] NVARCHAR(70), employee [city] NVARCHAR(40), [state] NVARCHAR(40), [country] NVARCHAR(40), [postal_code] NVARCHAR(10), [phone] NVARCHAR(24), [fax] NVARCHAR(24), [email] NVARCHAR(60), FOREIGN KEY ([reports_to]) REFERENCES [employee] ([employee_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [genre] [genre_id] INTEGER PRIMARY KEY NOT NULL, [name] NVARCHAR(120) CREATE TABLE [invoice] [invoice_id] INTEGER PRIMARY KEY NOT NULL, [customer_id] INTEGER NOT NULL, [invoice_date] DATETIME NOT NULL, [billing_address] NVARCHAR(70), [billing_city] NVARCHAR(40), invoice [billing_state] NVARCHAR(40), [billing_country] NVARCHAR(40), [billing_postal_code] NVARCHAR(10), [total] NUMERIC(10,2) NOT NULL, FOREIGN KEY ([customer_id]) REFERENCES [customer] ([customer_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [invoice_line] [invoice_line_id] INTEGER PRIMARY KEY NOT NULL, [invoice_id] INTEGER NOT NULL, [track_id] INTEGER NOT NULL, [unit_price] NUMERIC(10,2) NOT NULL, invoice_line [quantity] INTEGER NOT NULL, FOREIGN KEY ([invoice_id]) REFERENCES [invoice] ([invoice_id]) ON DELETE NO ACTION ON UPDATE NO ACTION, FOREIGN KEY ([track_id]) REFERENCES [track] ([track_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [media_type] [media_type_id] INTEGER PRIMARY KEY NOT NULL, media_type [name] NVARCHAR(120) CREATE TABLE [playlist] [playlist_id] INTEGER PRIMARY KEY NOT NULL, playlist [name] NVARCHAR(120) CREATE TABLE [playlist_track] [playlist_id] INTEGER NOT NULL, [track_id] INTEGER NOT NULL, CONSTRAINT [pk_playlist_track] PRIMARY KEY ([playlist_id], [track_id]), playlist_track FOREIGN KEY ([playlist_id]) REFERENCES [playlist] ([playlist_id]) ON DELETE NO ACTION ON UPDATE NO ACTION, FOREIGN KEY ([track_id]) REFERENCES [track] ([track_id]) ON DELETE NO ACTION ON UPDATE NO ACTION CREATE TABLE [track] [track_id] INTEGER PRIMARY KEY NOT NULL, [name] NVARCHAR(200) NOT NULL, [album_id] INTEGER, [media type id] INTEGER NOT NULL, [genre_id] INTEGER, [composer] NVARCHAR(220), [milliseconds] INTEGER NOT NULL, track [bytes] INTEGER, [unit_price] NUMERIC(10,2) NOT NULL, FOREIGN KEY ([album id]) REFERENCES [album] ([album id]) ON DELETE NO ACTION ON UPDATE NO ACTION, FOREIGN KEY ([genre_id]) REFERENCES [genre] ([genre_id]) ON DELETE NO ACTION ON UPDATE NO ACTION, FOREIGN KEY ([media_type_id]) REFERENCES [media_type] ([media_type_id]) ON DELETE NO ACTION ON UPDATE NO ACTION **Overview of the Data** In [14]: | %%**sql** SELECT name, type FROM sqlite_master WHERE type IN ("table", "view"); Done. Out[14]: name type album table artist table customer table employee table genre table invoice table invoice_line table media_type table playlist table playlist_track table track table **Selecting Albums to Purchase** Now, I will write a query to find out which genres sell the most tracks in the USA. In [15]: %%**sql** -- Create a temporary table to store USA track sales CREATE TEMPORARY TABLE IF NOT EXISTS USA_Track_Sales AS SELECT il.invoice_id, il.track_id FROM invoice_line il INNER JOIN invoice i ON il.invoice_id = i.invoice_id INNER JOIN customer c ON i.customer_id = c.customer_id WHERE c.country = 'USA'; -- Main query using the temporary table SELECT g.name AS genre, COUNT(uts.track_id) AS tracks_sold, CAST(COUNT(uts.track_id) AS FLOAT) / (SELECT COUNT(*) FROM USA_Track_Sales) * 100 AS percentage_sold FROM genre g LEFT JOIN track t ON g.genre_id = t.genre_id LEFT JOIN USA Track Sales uts ON t.track id = uts.track id GROUP BY genre ORDER BY tracks sold DESC; Done. Done. Out[15]: genre tracks_sold percentage_sold 53.37773549000951 Rock Alternative & Punk 130 12.369172216936251 124 11.798287345385347 Metal R&B/Soul 5.042816365366318 Blues 3.425309229305423 35 3.3301617507136063 Alternative Latin 22 2.093244529019981 22 2.093244529019981 Hip Hop/Rap 20 1.9029495718363463 Jazz 14 1.3320647002854424 Easy Listening 1.236917221693625 Reggae 6 0.570884871550904 5 0.47573739295908657 Electronica/Dance 4 0.3805899143672693 Classical Heavy Metal 3 0.285442435775452 2 0.19029495718363465 Soundtrack TV Shows 1 0.09514747859181732 Bossa Nova 0 0.0 Comedy 0 0.0 Drama Opera 0 0.0 0.0 Rock And Roll 0 0.0 Sci Fi & Fantasy 0 0.0 Science Fiction 0 0 0.0 World Based on the sales data and the popularity of different music genres among customers in the USA, it's evident that the "Rock" genre dominates the market with a significant 53.38% of tracks sold. Following closely are "Alternative & Punk" at 12.37% and "Metal" at 11.80%, showcasing a strong preference for these energetic and diverse genres. In light of this, it's recommended that the store consider purchasing albums from artists who specialize in these genres. Prioritizing artists who have contributed to the success of these top-selling genres can help the store cater to its audience's preferences, thereby boosting sales and customer satisfaction. By aligning the store's inventory with the prevailing trends, it can create a more engaging shopping experience and ensure that it remains a go-to destination for music enthusiasts seeking the latest and most popular tracks in these genres. **Analyzing Employee Sales Performance** In [16]: %%**sql** e.employee_id, e.first_name || ' ' || e.last_name AS employee_name, e.hire date, e.city, e.state, e.country, SUM(i.total) AS total_sales_amount FROM employee e LEFT JOIN customer c ON e.employee_id = c.support_rep_id LEFT JOIN invoice i ON c.customer_id = i.customer_id WHERE e.title = 'Sales Support Agent' GROUP BY e.employee_id ORDER BY total sales amount DESC; Done. Out[16]: employee_id employee_name hire_date city state country total_sales_amount 3 Jane Peacock 2017-04-01 00:00:00 Calgary AB Canada 1731.510000000004 Margaret Park 2017-05-03 00:00:00 Calgary AB Canada 1584.0000000000032 5 Steve Johnson 2017-10-17 00:00:00 Calgary AB Canada 1393.9200000000028 The query results show the total dollar amount of sales assigned to each sales support agent within the company, along with their relevant attributes. Here's a short interpretation of the results: • Jane Peacock: Jane has the highest total sales amount among the sales support agents, totaling \$1731.51. She was hired on April 1, 2017, and is located in Calgary, Alberta, Canada. Her strong performance could be due to factors like experience, customer relationships, and effective sales strategies. • Margaret Park: Margaret follows closely with a total sales amount of \$1584.00. She was hired on May 3, 2017, and also operates from Calgary, Alberta, Canada. Her performance is impressive, and her slightly lower sales amount compared to Jane might be attributed to factors such as the specific clientele she handles. • Steve Johnson: Steve has a total sales amount of \$1393.92. He was hired on October 17, 2017, and is also based in Calgary, Alberta, Canada. Although his sales amount is slightly lower than Jane and Margaret, it's important to consider his relatively shorter time in the role and its impact on his performance. Overall, the analysis indicates that Jane Peacock has achieved the highest total sales, followed closely by Margaret Park and Steve Johnson. The variations in their sales amounts could be influenced by factors like tenure, customer relationships, sales strategies, and the types of customers they interact with. This data provides valuable insights into the performance of each sales support agent and can guide decisions regarding training, resource allocation, and ongoing performance evaluations. **Analyzing Sales by Country** In [17]: | %%**sql** WITH CountrySales AS (SELECT CASE WHEN (SELECT COUNT(*) FROM customer WHERE c.country = customer.country) = 1 THEN 'Other' ELSE c.country END AS country_group, c.customer_id, i.invoice_id, i.total FROM customer c INNER JOIN invoice i ON c.customer_id = i.customer_id SELECT country_group AS country, COUNT(DISTINCT customer_id) AS total_customers, ROUND(SUM(total), 2) AS total_sales, ROUND(SUM(total) / COUNT(DISTINCT customer_id), 2) AS avg_sales_per_customer, ROUND(SUM(total) / COUNT(DISTINCT invoice_id), 2) AS avg_order_value FROM CountrySales GROUP BY country_group ORDER BY CASE WHEN country_group = 'Other' THEN 1 ELSE 0 END, total_sales DESC; Done. Out[17]: country total_customers total_sales avg_sales_per_customer avg_order_value 1040.49 13 80.04 7.94 Canada 8 535.59 66.95 7.05 427.68 85.54 5 7.01 Brazil 389.07 France 77.81 7.78 334.62 83.66 8.16 Germany 4 273.24 136.62 Czech Republic 9.11 245.52 81.84 8.77 United Kingdom 3 2 185.13 92.57 6.38 Portugal 2 183.15 91.57 8.72 India Other 15 1094.94 73.0 7.45 Now, I will interpret the results and extract insights from the data: USA: The USA has the highest number of customers (13) and the highest total sales (\$1040.49). The average sales per customer are \$80.04, suggesting that customers in the USA tend to make moderate-sized purchases. The average order value is \$7.94, indicating that customers typically spend less per order but make more frequent purchases. Canada: Canada follows with 8 customers and total sales of \$535.59. The average sales per customer are \$66.95, s imilar to the USA, while the average order value is \$7.05. Customers in Canada also exhibit a preference for sma ller, more frequent purchases. Brazil: With 5 customers, Brazil's total sales amount to \$427.68. The average sales per customer are relatively higher at \$85.54, indicating that customers from Brazil tend to make larger individual purchases. The average or der value is \$7.01, similar to other countries with moderate sales figures. France, Germany, Czech Republic, United Kingdom, Portugal, India: These countries share similarities in terms of the number of customers, total sales, and average sales per customer. While their average order values vary slig htly, they all fall within the range of \$6.38 to \$9.11. These countries seem to have a balanced distribution of customers, sales, and purchase habits. Other: The "Other" group consists of countries with only one customer each. Despite the higher number of countries es (15), the total sales amount is \$1094.94. The average sales per customer are lower at \$73.00, suggesting that single-customer countries contribute to lower overall spending. The average order value is \$7.45, aligning with the trend of moderate spending across various countries. In summary, the analysis highlights variations in customer behavior and spending patterns across different count ries. While the USA and Canada lead in terms of customer base and total sales, other countries like Brazil exhib it higher average sales per customer. The average order values remain relatively consistent across countries, in dicating similar purchasing trends. The "Other" category, which groups countries with one customer, showcases lo wer average sales per customer and highlights the impact of smaller markets on overall sales figures. This infor mation can guide marketing strategies and resource allocation based on the preferences and spending habits of cu stomers from various regions. **Albums vs Individual Tracks** In [18]: %%sql WITH invoice first track AS (SELECT il.invoice_id AS invoice_id, MIN(il.track id) AS first track id invoice_line il GROUP BY -- Use a subquery to select the results of the invoice_first_track CTE and determine whether customers made album purc hases SELECT album_purchase, COUNT(invoice id) AS number of invoices, CAST(COUNT(invoice_id) AS FLOAT) / (SELECT COUNT(*) FROM invoice) AS percent FROM SELECT ifs.*, CASE -- Use the EXCEPT operator to compare the tracks in the first invoice with the tracks in subsequent invoices, -- and determine whether any tracks from the album were purchased in subsequent invoices. -- If the result of the EXCEPT is NULL, it means that all tracks from the album were purchased in subsequent i nvoices, -- and the customer made an album purchase. -- If the result of the EXCEPT is not NULL, it means that at least one track from the album was not purchased in subsequent invoices, -- and the customer did not make an album purchase. WHEN (SELECT t.track_id FROM track t WHERE t.album id = (SELECT t2.album_id FROM track t2 WHERE t2.track_id = ifs.first_track_id EXCEPT SELECT il2.track_id FROM invoice_line il2 WHERE il2.invoice_id = ifs.invoice_id) IS NULL AND (SELECT il2.track_id FROM invoice_line il2 WHERE il2.invoice_id = ifs.invoice_id EXCEPT SELECT t.track_id FROM track t WHERE t.album id = (SELECT t2.album_id FROM track t2 WHERE t2.track_id = ifs.first_track_id) IS NULL THEN "yes" ELSE "no" END AS album_purchase invoice_first_track ifs) subquery -- Group by album_purchase to get the counts and percentages for each type of purchase album_purchase; Done. Out[18]: album_purchase number_of_invoices percent 500 0.8143322475570033 114 0.18566775244299674 These results indicate that a significant portion of the invoices consist of non-album purchases (81.43%), where customers have purchased individual tracks that are not part of a full album. On the other hand, a smaller proportion of invoices (18.57%) are categorized as album purchases, where customers have bought a complete album. This insight into the distribution of album and non-album purchases can be valuable for business decisions and marketing strategies. For instance, you might explore ways to encourage more album purchases, such as offering special discounts for complete albums or promoting album bundles. Understanding customer preferences for albums versus individual tracks can help optimize inventory and tailor promotional efforts to maximize revenue.

SQL Project: Finding Opportunities to Increase Sales

CREATE TABLE [album]

[album_id] INTEGER PRIMARY KEY NOT NULL,

Introduction and Schema Diagram

In [12]: | %%capture

In [13]: %%sql

Out[13]:

%load ext sql

SELECT

FROM

WHERE

Done.

name,

name

Out[12]: 'Connected: None@chinook.db'

sqlite_master

type IN ("table","view");

%sql sqlite:///chinook.db