APPLE ITUNES MUSIC ANALYSIS PROJECT SQL.sql

```
CREATE DATABASE MUSIC_DATABASE;
USE MUSIC_DATABASE;

CREATE TABLE ALBUMS_DATASET(
ALBUM_ID INTEGER PRIMARY KEY,
TITLE TEXT NOT NULL,
ARTIST INTEGER NOT NULL);

CREATE TABLE ARTIST_DATA(
ARTIST_ID INTEGER PRIMARY KEY,
NAME TEXT NOT NULL);

CREATE TABLE employee_data (
    employee_id INT PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    title VARCHAR(100),
    reports_to INT,
    levels VARCHAR(10),
    birthdate DATETIME,
    hire_date DATETIME,
    address VARCHAR(255),
    city VARCHAR(100),
```

```
    state VARCHAR(50),
    country VARCHAR(50),
    postal_code VARCHAR(20),
    phone VARCHAR(50),
    fax VARCHAR(50),
    email VARCHAR(255)
);

CREATE TABLE genre (
    genre_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE invoice (
    invoice_id INT PRIMARY KEY,
    customer_id INT NOT NULL,
    invoice_date DATETIME NOT NULL,
    billing_address VARCHAR(255),
    billing_city VARCHAR(100),
    billing_state VARCHAR(50),
    billing_country VARCHAR(50),
    billing_postal_code VARCHAR(20),
    total DECIMAL(10, 2) NOT NULL
);
```

```
CREATE TABLE invoice_line (
    invoice_line_id INTEGER PRIMARY KEY,
    invoice_id INTEGER,
    track_id INTEGER,
    unit_price DECIMAL(10, 2),
    quantity INTEGER
);

CREATE TABLE media_type (
    media_type_id INTEGER PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE playlist (
    playlist_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE playlist_track (
    playlist_id INT NOT NULL,
    track_id INT NOT NULL,
    PRIMARY KEY (playlist_id, track_id),
    FOREIGN KEY (playlist_id) REFERENCES playlist(playlist_id)
```

```
);

CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(40) NOT NULL,
    last_name VARCHAR(40) NOT NULL,
    company VARCHAR(80),
    address VARCHAR(120),
    city VARCHAR(60),
    state VARCHAR(40),
    country VARCHAR(40),
    postal_code VARCHAR(20),
    phone VARCHAR(30),
    fax VARCHAR(30),
    email VARCHAR(80),
    support_rep_id INT
);

CREATE TABLE track (
    track_id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    album_id INTEGER,
    media_type_id INTEGER,
    genre_id INTEGER,
```

```
    composer VARCHAR(255),
    milliseconds INTEGER,
    bytes INTEGER,
    unit_price NUMERIC(4,2)
);


select * from employee_data;


insert into employee_data (employee_id, last_name, first_name, title,
reports_to, levels,
birthdate, hire_date, address, city, state, country, postal_code, phone, fax, email)
values
(1, 'Adams', 'Andrew','General Manager',9,'L6', '1962-02-18 00:00','2016-08-14
00:00','1120 Jasper Ave NW',
'Edmonton','AB','Canada','T5K 2N1','+1 (780)-428-9482','+1 (780)-428-
3457','andrew@chinookcorp.com'),
(2,'Edwards','Nancy','Sales Manager',1,'L4', '1958-12-08 00:00','2016-05-01
00:00','825 8 Ave SW',
'Calgary','AB','Canada','T2P 2T3', '+1 (403) 262-3443', '+1 (403) 262-3322',
'nancy@chinookcorp.com'),
(3,'Peacock','Jane', 'Sales Support Agent',2, 'L1','1973-08-29 00:00','2017-04-01
00:00',
'1111 6 Ave SW', 'Calgary', 'AB', 'Canada',  'T2P 5M5', '+1 (403) 262-3443','+1
(403) 262-6712', 'jane@chinookcorp.com'),
(4,'Park','Margaret','Sales Support Agent', 2,'L1', '1947-09-19 00:00',  '2017-05-
03 00:00',
'683 10 Street SW','Calgary', 'AB',' Canada', 'T2P 5G3',' +1 (403) 263-4423', '+1
(403) 263-4289', 'margaret@chinookcorp.com'),
```

(5,'Johnson','Steve', 'Sales Support Agent', 2, 'L1', '1965-03-03 00:00', '2017-10-17 00:00',

'7727B 41 Ave',' Calgary', 'AB','Canada', 'T3B 1Y7', '1 (780) 836-9987', '1 (780) 836-9543', 'steve@chinookcorp.com'),

(6,'Mitchell','Michael', 'IT Manager', 1, 'L3', '1973-07-01 00:00', '2016-10-17 00:00',

'5827 Bowness Road NW', 'Calgary', 'AB', 'Canada', 'T3B 0C5', '+1 (403) 246-9887', '+1 (403) 246-9899', 'michael@chinookcorp.com'),

(7, 'King', 'Robert', 'IT Staff',6, 'L2', '1970-05-29 00:00', '2017-01-02 00:00',

'590 Columbia Boulevard West', 'Lethbridge', 'AB', 'Canada', 'T1K 5N8', '+1 (403) 456-9986', '+1 (403) 456-8485', 'robert@chinookcorp.com'),

(8,'Callahan', 'Laura', 'IT Staff', 6, 'L2', '1968-01-09 00:00', '2017-03-04 00:00',

'923 7 ST NW', 'Lethbridge', 'AB', 'Canada', 'T1H 1Y8', '+1 (403) 467-3351', '+1 (403) 467-8772', 'laura@chinookcorp.com'),

(9, 'Madan', 'Mohan', 'Senior General Manager', 5,'L7', '1961-01-26 00:00', '2016-01-14 00:00',

'1008 Vrinda Ave MT', 'Edmonton', 'AB', 'Canada', 'T5K 2N1', '+1 (780) 428-9482', '+1 (780) 428-3457', 'madan.mohan@chinookcorp.com');

```
select * from employee_data;

select * from albums_dataset;

select * from customer;

select * from genre;

select * from invoice;

select * from invoice_line;

select * from media_type;

select * from playlist;
```

```
select * from playlist_track;

select * from track;
```

-- 1. Customer Overview

```
SELECT
    country,
    COUNT(*) as total_customers,
    ROUND(AVG(total), 2) as avg_invoice_value
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY country
ORDER BY total_customers DESC;
```

-- 2. Revenue Trends by Month

```
SELECT
    YEAR(invoice_date) as year,
    MONTH(invoice_date) as month,
    SUM(total) as monthly_revenue,
    ROUND(SUM(total) / SUM(SUM(total)) OVER (PARTITION BY
YEAR(invoice_date)) * 100, 2) as revenue_percentage
FROM invoice
GROUP BY YEAR(invoice_date), MONTH(invoice_date)
ORDER BY year, month;
```

-- 3. Top Selling Artists

```
SELECT
    a.name as artist_name,
    COUNT(il.track_id) as tracks_sold,
    SUM(il.unit_price * il.quantity) as total_revenue
FROM ARTIST_DATA a
JOIN ALBUMS_DATASET al ON a.ARTIST_ID = al.ARTIST   -- Changed to al.ARTIST
JOIN track t ON al.ALBUM_ID = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY a.name
ORDER BY total_revenue DESC
LIMIT 10;
```

-- 4. Customer Engagement by Support Rep

```
SELECT
    e.first_name,
    e.last_name,
    COUNT(DISTINCT c.customer_id) as customers_supported,
    SUM(i.total) as total_sales,
    ROUND(AVG(i.total), 2) as avg_sale_value
FROM employee_data e
JOIN customer c ON e.employee_id = c.support_rep_id
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY e.employee_id
ORDER BY total_sales DESC;
```

-- 5. Playlist Popularity

```
SELECT
    p.name as playlist_name,
    COUNT(pt.track_id) as total_tracks,
    COUNT(DISTINCT il.invoice_id) as times_purchased
FROM playlist p
JOIN playlist_track pt ON p.playlist_id = pt.playlist_id
LEFT JOIN invoice_line il ON pt.track_id = il.track_id
GROUP BY p.playlist_id
ORDER BY times_purchased DESC;
```

--- ADVANCED ANALYTICS ---

```
-- 1. Customer Segmentation using RFM Analysis
WITH customer_rfm AS (
    SELECT
        c.customer_id,
        c.first_name,
        c.last_name,
        DATEDIFF(MAX(i.invoice_date), CURRENT_DATE()) as recency,
        COUNT(i.invoice_id) as frequency,
        SUM(i.total) as monetary,
        NTILE(4) OVER (ORDER BY DATEDIFF(MAX(i.invoice_date),
CURRENT_DATE()) DESC) as r_score,
        NTILE(4) OVER (ORDER BY COUNT(i.invoice_id)) as f_score,
```

```
        NTILE(4) OVER (ORDER BY SUM(i.total)) as m_score
    FROM customer c
    JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.customer_id
)
SELECT
    customer_id,
    first_name,
    last_name,
    recency,
    frequency,
    monetary,
    r_score,
    f_score,
    m_score,
    CASE
        WHEN r_score = 4 AND f_score >= 3 AND m_score >= 3 THEN
'Champions'
        WHEN r_score >= 3 AND f_score >= 3 THEN 'Loyal Customers'
        WHEN r_score >= 3 AND m_score >= 3 THEN 'Potential Loyalists'
        WHEN r_score = 2 THEN 'Recent Customers'
        WHEN r_score = 1 THEN 'At Risk'
        ELSE 'Need Attention'
    END as customer_segment
FROM customer_rfm
```

```
ORDER BY monetary DESC;
```

-- 2. Top Performing Tracks with Window Functions

```
WITH track_performance AS (
    SELECT
        t.track_id,
        t.name as track_name,
        a.name as artist_name,
        al.title as album_name,
        COUNT(il.invoice_line_id) as times_purchased,
        SUM(il.unit_price * il.quantity) as total_revenue,
        RANK() OVER (ORDER BY SUM(il.unit_price * il.quantity) DESC) as
revenue_rank,
        DENSE_RANK() OVER (PARTITION BY g.genre_id ORDER BY
SUM(il.unit_price * il.quantity) DESC) as genre_rank
    FROM track t
    JOIN ALBUMS_DATASET al ON t.album_id = al.ALBUM_ID
    JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID  -- Changed to
al.ARTIST
    JOIN genre g ON t.genre_id = g.genre_id
    JOIN invoice_line il ON t.track_id = il.track_id
    GROUP BY t.track_id
)
SELECT
    track_name,
```

```
    artist_name,
    album_name,
    times_purchased,
    total_revenue,
    revenue_rank,
    genre_rank
FROM track_performance
WHERE revenue_rank <= 20
ORDER BY revenue_rank;

-- 3. Monthly Sales Growth Analysis
WITH monthly_sales AS (
    SELECT
        YEAR(invoice_date) as year,
        MONTH(invoice_date) as month,
        SUM(total) as monthly_revenue,
        LAG(SUM(total)) OVER (ORDER BY YEAR(invoice_date),
MONTH(invoice_date)) as prev_month_revenue
    FROM invoice
    GROUP BY YEAR(invoice_date), MONTH(invoice_date)
)
SELECT
    year,
    month,
    monthly_revenue,
```

```
    prev_month_revenue,
    ROUND(((monthly_revenue - prev_month_revenue) / prev_month_revenue)
* 100, 2) as growth_percentage,
    CASE
        WHEN monthly_revenue > prev_month_revenue THEN 'Growth'
        WHEN monthly_revenue < prev_month_revenue THEN 'Decline'
        ELSE 'Stable'
    END as trend
FROM monthly_sales
ORDER BY year, month;
```

-- 4. Customer Lifetime Value (CLV) Analysis

```
WITH customer_purchases AS (
    SELECT
        c.customer_id,
        c.first_name,
        c.last_name,
        COUNT(i.invoice_id) as total_purchases,
        SUM(i.total) as total_spent,
        DATEDIFF(MAX(i.invoice_date), MIN(i.invoice_date)) as
customer_tenure_days,
        CASE
            WHEN DATEDIFF(MAX(i.invoice_date), MIN(i.invoice_date)) = 0
THEN SUM(i.total)
            ELSE SUM(i.total) / (DATEDIFF(MAX(i.invoice_date),
MIN(i.invoice_date)) / 30.0)
```

```
        END as monthly_value
    FROM customer c
    JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.customer_id
)
SELECT
    customer_id,
    first_name,
    last_name,
    total_purchases,
    total_spent,
    customer_tenure_days,
    monthly_value,
    NTILE(5) OVER (ORDER BY monthly_value DESC) as value_segment
FROM customer_purchases
ORDER BY monthly_value DESC;
```

-- 5. Genre Popularity Over Time

```
SELECT
    g.name as genre_name,
    YEAR(i.invoice_date) as year,
    QUARTER(i.invoice_date) as quarter,
    COUNT(il.invoice_line_id) as tracks_sold,
    SUM(il.unit_price * il.quantity) as genre_revenue,
```

```
    ROUND(SUM(il.unit_price * il.quantity) / SUM(SUM(il.unit_price *
il.quantity))
        OVER (PARTITION BY YEAR(i.invoice_date),
QUARTER(i.invoice_date)) * 100, 2) as market_share
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
JOIN invoice_line il ON t.track_id = il.track_id
JOIN invoice i ON il.invoice_id = i.invoice_id
GROUP BY g.genre_id, YEAR(i.invoice_date), QUARTER(i.invoice_date)
ORDER BY year, quarter, genre_revenue DESC;
```

APPLE ITUNES MUSIC ANALYSIS PROJECT SQL 1.sql

-- CUSTOMER ANALYTICS --

-- 1.1 Which customers have spent the most money on music?

```
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) as customer_name,
    c.country,
    c.email,
    SUM(i.total) as total_spent,
    COUNT(i.invoice_id) as total_purchases
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id
ORDER BY total_spent DESC
```

```
LIMIT 10;


-- 1.2 What is the average customer lifetime value?
SELECT
    ROUND(AVG(total_spent), 2) as avg_lifetime_value,
    ROUND(AVG(purchase_count), 2) as avg_purchases_per_customer
FROM (
    SELECT
        c.customer_id,
        SUM(i.total) as total_spent,
        COUNT(i.invoice_id) as purchase_count
    FROM customer c
    JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.customer_id
) customer_stats;


-- 1.3 How many customers have made repeat purchases versus one-time purchases?
SELECT
    purchase_type,
    COUNT(*) as customer_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customer), 2) as percentage
FROM (
    SELECT
```

```
        c.customer_id,
        CASE
            WHEN COUNT(i.invoice_id) > 1 THEN 'Repeat Customer'
            WHEN COUNT(i.invoice_id) = 1 THEN 'One-time Customer'
            ELSE 'No Purchase'
        END as purchase_type
    FROM customer c
    LEFT JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.customer_id
) purchase_categories
GROUP BY purchase_type;
```

-- 1.4 Which country generates the most revenue per customer?

```
SELECT
    country,
    COUNT(DISTINCT c.customer_id) as total_customers,
    SUM(i.total) as total_revenue,
    ROUND(SUM(i.total) / COUNT(DISTINCT c.customer_id), 2) as
revenue_per_customer
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY country
HAVING total_customers > 5
ORDER BY revenue_per_customer DESC;
```

-- 1.5 Which customers haven't made a purchase in the last 6 months?

```
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) as customer_name,
    c.email,
    c.country,
    MAX(i.invoice_date) as last_purchase_date,
    DATEDIFF(CURDATE(), MAX(i.invoice_date)) as
days_since_last_purchase
FROM customer c
LEFT JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id
HAVING last_purchase_date IS NULL OR last_purchase_date <
DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
ORDER BY days_since_last_purchase DESC;
```

<u>-- SALES AND REVENUE ANALYSIS --</u>

-- 2.1 Monthly revenue trends for the last two years

```
SELECT
    YEAR(invoice_date) as year,
    MONTH(invoice_date) as month,
    MONTHNAME(invoice_date) as month_name,
    COUNT(invoice_id) as total_invoices,
    SUM(total) as monthly_revenue,
    ROUND(AVG(total), 2) as avg_invoice_value
```

```
FROM invoice
WHERE invoice_date >= DATE_SUB((SELECT MAX(invoice_date) FROM invoice), INTERVAL 2 YEAR)
GROUP BY YEAR(invoice_date), MONTH(invoice_date), MONTHNAME(invoice_date)
ORDER BY year DESC, month DESC;
```

-- 2.2 Average value of an invoice

```
SELECT
    ROUND(AVG(total), 2) as avg_invoice_value,
    MIN(total) as min_invoice_value,
    MAX(total) as max_invoice_value,
    COUNT(*) as total_invoices
FROM invoice;
```

-- 2.3 Payment methods analysis (assuming payment method is in invoice table)

-- If you have a payment_method column, use this:

```
SELECT
    DAYNAME(invoice_date) as day_of_week,
    COUNT(*) as transaction_count,
    SUM(total) as total_revenue,
    ROUND(AVG(total), 2) as avg_transaction_value,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM invoice), 2) as percentage_of_total
FROM invoice
```

```
GROUP BY DAYNAME(invoice_date), DAYOFWEEK(invoice_date)
ORDER BY DAYOFWEEK(invoice_date);

SELECT
    HOUR(invoice_date) as hour_of_day,
    COUNT(*) as transaction_count,
    SUM(total) as total_revenue,
    ROUND(AVG(total), 2) as avg_transaction_value
FROM invoice
GROUP BY HOUR(invoice_date)
ORDER BY hour_of_day;

-- 2.4 Revenue contribution by sales representative
SELECT
    e.employee_id,
    CONCAT(e.first_name, ' ', e.last_name) as sales_rep,
    e.title,
    COUNT(DISTINCT c.customer_id) as customers_managed,
    COUNT(i.invoice_id) as total_invoices,
    SUM(i.total) as total_revenue,
    ROUND(SUM(i.total) / COUNT(i.invoice_id), 2) as avg_sale_value
FROM employee_data e
JOIN customer c ON e.employee_id = c.support_rep_id
JOIN invoice i ON c.customer_id = i.customer_id
```

```
GROUP BY e.employee_id
ORDER BY total_revenue DESC;

-- 2.5 Peak sales months or quarters
SELECT
    YEAR(invoice_date) as year,
    QUARTER(invoice_date) as quarter,
    COUNT(invoice_id) as total_invoices,
    SUM(total) as quarterly_revenue,
    ROUND(SUM(total) / COUNT(DISTINCT MONTH(invoice_date)), 2) as
avg_monthly_revenue
FROM invoice
GROUP BY YEAR(invoice_date), QUARTER(invoice_date)
ORDER BY quarterly_revenue DESC;

-- PRODUCT & CONTENT ANALYSIS --
-- 3.1 Tracks that generated the most revenue
SELECT
    t.track_id,
    t.name as track_name,
    a.name as artist_name,
    al.title as album_name,
    g.name as genre,
    COUNT(il.invoice_line_id) as times_purchased,
    SUM(il.unit_price * il.quantity) as total_revenue
```

```
FROM track t
JOIN ALBUMS_DATASET al ON t.album_id = al.ALBUM_ID
JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID
JOIN genre g ON t.genre_id = g.genre_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY t.track_id
ORDER BY total_revenue DESC
LIMIT 20;

-- 3.2 Most frequently purchased albums
SELECT
    al.ALBUM_ID,
    al.title as album_name,
    a.name as artist_name,
    COUNT(DISTINCT il.invoice_id) as times_purchased,
    COUNT(il.track_id) as total_tracks_sold,
    SUM(il.unit_price * il.quantity) as total_revenue
FROM ALBUMS_DATASET al
JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID
JOIN track t ON al.ALBUM_ID = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY al.ALBUM_ID
ORDER BY times_purchased DESC
LIMIT 15;
```

```
-- 3.3 Tracks or albums that have never been purchased
-- Tracks never purchased
SELECT
    t.track_id,
    t.name as track_name,
    a.name as artist_name,
    al.title as album_name
FROM track t
JOIN ALBUMS_DATASET al ON t.album_id = al.ALBUM_ID
JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID
LEFT JOIN invoice_line il ON t.track_id = il.track_id
WHERE il.track_id IS NULL;

-- 3.4 Average price per track across different genres
SELECT
    g.genre_id,
    g.name as genre_name,
    COUNT(t.track_id) as total_tracks,
    ROUND(AVG(t.unit_price), 2) as avg_track_price,
    SUM(il.unit_price * il.quantity) as total_genre_revenue,
    COUNT(il.invoice_line_id) as total_tracks_sold
FROM genre g
LEFT JOIN track t ON g.genre_id = t.genre_id
```

```
LEFT JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY g.genre_id
ORDER BY total_genre_revenue DESC;

-- 3.5 Tracks per genre vs sales correlation
SELECT
    g.genre_id,
    g.name as genre_name,
    COUNT(DISTINCT t.track_id) as available_tracks,
    COUNT(il.invoice_line_id) as tracks_sold,
    ROUND(COUNT(il.invoice_line_id) * 100.0 / COUNT(DISTINCT t.track_id), 2) as sales_ratio,
    SUM(il.unit_price * il.quantity) as total_revenue
FROM genre g
LEFT JOIN track t ON g.genre_id = t.genre_id
LEFT JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY g.genre_id
ORDER BY sales_ratio DESC;
```

APPLE ITUNES MUSIC ANALYSIS PROJECT SQL 2.sql

-- ARTIST & GENRE PERFORMANCE --

-- 4.1 Top 5 highest-grossing artists

```
SELECT
    a.ARTIST_ID,
    a.name as artist_name,
    COUNT(DISTINCT al.ALBUM_ID) as total_albums,
    COUNT(DISTINCT t.track_id) as total_tracks,
    COUNT(il.invoice_line_id) as tracks_sold,
    SUM(il.unit_price * il.quantity) as total_revenue
FROM ARTIST_DATA a
JOIN ALBUMS_DATASET al ON a.ARTIST_ID = al.ARTIST
JOIN track t ON al.ALBUM_ID = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY a.ARTIST_ID
ORDER BY total_revenue DESC
LIMIT 5;
```

-- 4.2 Genre popularity by tracks sold

```
SELECT
    g.genre_id,
    g.name as genre_name,
    COUNT(il.invoice_line_id) as tracks_sold,
```

```
    SUM(il.unit_price * il.quantity) as total_revenue,
    ROUND(AVG(t.unit_price), 2) as avg_price
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY g.genre_id
ORDER BY tracks_sold DESC;

-- 4.3 Genre popularity by revenue
SELECT
    g.genre_id,
    g.name as genre_name,
    SUM(il.unit_price * il.quantity) as total_revenue,
    COUNT(il.invoice_line_id) as tracks_sold,
    ROUND(SUM(il.unit_price * il.quantity) / COUNT(il.invoice_line_id), 2) as
revenue_per_track
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY g.genre_id
ORDER BY total_revenue DESC;

-- 4.4 Genre popularity by country
SELECT
    c.country,
```

```
    g.name as genre_name,
    COUNT(il.invoice_line_id) as tracks_sold,
    SUM(il.unit_price * il.quantity) as total_revenue
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
JOIN invoice_line il ON i.invoice_id = il.invoice_id
JOIN track t ON il.track_id = t.track_id
JOIN genre g ON t.genre_id = g.genre_id
GROUP BY c.country, g.name
ORDER BY c.country, total_revenue DESC;
```

--- EMPLOYEE & OPERATIONAL EFFICIENCY ---

-- 5.1 Employees managing highest-spending customers

```
SELECT
    e.employee_id,
    CONCAT(e.first_name, ' ', e.last_name) as sales_rep,
    e.title,
    COUNT(DISTINCT c.customer_id) as total_customers,
    SUM(i.total) as total_revenue,
    ROUND(SUM(i.total) / COUNT(DISTINCT c.customer_id), 2) as
revenue_per_customer,
    MAX(i.total) as largest_sale
FROM employee_data e
JOIN customer c ON e.employee_id = c.support_rep_id
JOIN invoice i ON c.customer_id = i.customer_id
```

```
GROUP BY e.employee_id
ORDER BY total_revenue DESC;
```

-- 5.2 Average number of customers per employee

```
SELECT
    ROUND(COUNT(DISTINCT c.customer_id) * 1.0 / COUNT(DISTINCT e.employee_id), 2) as avg_customers_per_rep
FROM employee_data e
LEFT JOIN customer c ON e.employee_id = c.support_rep_id
WHERE e.title LIKE '%Sales%' OR e.title LIKE '%Support%';
```

-- 5.3 Revenue by employee regions

```
SELECT
    e.city as employee_city,
    e.country as employee_country,
    COUNT(DISTINCT e.employee_id) as total_employees,
    COUNT(DISTINCT c.customer_id) as customers_managed,
    SUM(i.total) as total_revenue
FROM employee_data e
JOIN customer c ON e.employee_id = c.support_rep_id
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY e.city, e.country
ORDER BY total_revenue DESC;
```

-- GEOGRAPHIC TRENDS --

-- 6.1 Countries with highest number of customers

```
SELECT
    country,
    COUNT(*) as total_customers,
    SUM(i.total) as total_revenue,
    ROUND(SUM(i.total) / COUNT(*), 2) as revenue_per_customer
FROM customer c
LEFT JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY country
ORDER BY total_customers DESC;
```

-- 6.2 Revenue variation by region

```
SELECT
    country,
    COUNT(DISTINCT c.customer_id) as total_customers,
    SUM(i.total) as total_revenue,
    ROUND(SUM(i.total) / COUNT(DISTINCT c.customer_id), 2) as
avg_revenue_per_customer,
    COUNT(i.invoice_id) as total_transactions
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY country
HAVING total_customers >= 5
ORDER BY total_revenue DESC;
```

-- 6.3 Underserved geographic regions

```
SELECT
    country,
    COUNT(*) as total_customers,
    COALESCE(SUM(i.total), 0) as total_revenue,
    CASE
        WHEN COUNT(*) > 10 AND COALESCE(SUM(i.total), 0) < 100 THEN
'High Potential - Low Revenue'
        WHEN COUNT(*) > 5 AND COALESCE(SUM(i.total), 0) < 50 THEN
'Medium Potential - Low Revenue'
        ELSE 'Adequately Served'
    END as service_status
FROM customer c
LEFT JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY country
ORDER BY total_customers DESC;
```

-- CUSTOMER RETENTION & PURCHASE RETURNS --

-- 7.1 Purchase frequency distribution

```
SELECT
    purchase_frequency,
    COUNT(*) as customer_count,
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customer), 2) as
percentage
FROM (
```

```
    SELECT
        c.customer_id,
        COUNT(i.invoice_id) as purchase_frequency
    FROM customer c
    LEFT JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.customer_id
) freq_table
GROUP BY purchase_frequency
ORDER BY purchase_frequency;

-- 7.2 Average time between customer purchases
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) as customer_name,
    COUNT(i.invoice_id) as total_purchases,
    ROUND(DATEDIFF(MAX(i.invoice_date), MIN(i.invoice_date)) /
NULLIF(COUNT(i.invoice_id) - 1, 0), 2) as avg_days_between_purchases
FROM customer c
JOIN invoice i ON c.customer_id = i.customer_id
GROUP BY c.customer_id
HAVING total_purchases > 1
ORDER BY avg_days_between_purchases;

-- 7.3 Customers purchasing from multiple genres
SELECT
```

```
    multi_genre_customers,
    COUNT(*) as customer_count,
    ROUND(COUNT(*) * 100.0 / MAX(total_customers), 2) as percentage
FROM (
    SELECT
        c.customer_id,
        COUNT(DISTINCT t.genre_id) as genres_purchased,
        CASE
            WHEN COUNT(DISTINCT t.genre_id) > 1 THEN 'Multi-Genre Buyer'
            WHEN COUNT(DISTINCT t.genre_id) = 1 THEN 'Single-Genre
Buyer'
            ELSE 'No Purchase'
        END as multi_genre_customers,
        (SELECT COUNT(*) FROM customer) as total_customers
    FROM customer c
    LEFT JOIN invoice i ON c.customer_id = i.customer_id
    LEFT JOIN invoice_line il ON i.invoice_id = il.invoice_id
    LEFT JOIN track t ON il.track_id = t.track_id
    GROUP BY c.customer_id
) genre_analysis
GROUP BY multi_genre_customers;
```

-- OPERATIONAL OPTIMIZATION --

-- 8.1 Common track combinations (purchased together)

```
SELECT
```

```
    t1.track_id as track1_id,
    t1.name as track1_name,
    t2.track_id as track2_id,
    t2.name as track2_name,
    COUNT(*) as times_purchased_together
FROM invoice_line il1
JOIN invoice_line il2 ON il1.invoice_id = il2.invoice_id AND il1.track_id < il2.track_id
JOIN track t1 ON il1.track_id = t1.track_id
JOIN track t2 ON il2.track_id = t2.track_id
GROUP BY t1.track_id, t2.track_id
HAVING times_purchased_together >= 3
ORDER BY times_purchased_together DESC
LIMIT 20;
```

-- 8.2 Pricing patterns and sales performance

```
SELECT
    price_range,
    COUNT(t.track_id) as total_tracks,
    SUM(il.quantity) as total_units_sold,
    ROUND(SUM(il.unit_price * il.quantity), 2) as total_revenue,
    ROUND(SUM(il.quantity) * 1.0 / COUNT(t.track_id), 2) as avg_units_per_track
FROM (
    SELECT
```

```
        track_id,
        CASE
            WHEN unit_price < 0.50 THEN 'Under $0.50'
            WHEN unit_price < 0.75 THEN '$0.50-$0.74'
            WHEN unit_price < 1.00 THEN '$0.75-$0.99'
            WHEN unit_price < 1.25 THEN '$1.00-$1.24'
            ELSE '$1.25+'
        END as price_range
    FROM track
) price_categories
JOIN track t ON price_categories.track_id = t.track_id
LEFT JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY price_range
ORDER BY total_revenue DESC;

-- 8.3 Media type usage trends
SELECT
    m.media_type_id,
    m.name as media_type,
    YEAR(i.invoice_date) as year,
    COUNT(il.invoice_line_id) as tracks_sold,
    SUM(il.unit_price * il.quantity) as total_revenue,
    ROUND((COUNT(il.invoice_line_id) - LAG(COUNT(il.invoice_line_id))
        OVER (PARTITION BY m.media_type_id ORDER BY
YEAR(i.invoice_date))) * 100.0 /
```

```
    LAG(COUNT(il.invoice_line_id)) OVER (PARTITION BY
m.media_type_id ORDER BY YEAR(i.invoice_date)), 2) as growth_percentage
FROM media_type m
JOIN track t ON m.media_type_id = t.media_type_id
JOIN invoice_line il ON t.track_id = il.track_id
JOIN invoice i ON il.invoice_id = i.invoice_id
GROUP BY m.media_type_id, YEAR(i.invoice_date)
ORDER BY m.media_type_id, year;
```

APPLE ITUNES MUSIC ANALYSIS PROJECT SQL 3sql

-- Who is the senior most employee based on job title? --

```
SELECT * FROM employee_data ORDER BY levels DESC LIMIT 1;
```

-- Which countries have the most Invoices? --

```
SELECT billing_country, COUNT(*) AS invoice_count FROM invoice
GROUP BY billing_country
ORDER BY invoice_count DESC;
```

-- What are top 3 values of total invoice? --

```
SELECT total FROM invoice  ORDER BY total DESC LIMIT 3;
```

-- Which city has the best customers? --

```
SELECT billing_city, SUM(total) AS total_revenue FROM invoice GROUP
BY billing_city
```

```
ORDER BY total_revenue DESC LIMIT 1;
```

-- Who is the best customer? --

```
SELECT c.customer_id, c.first_name, c.last_name, SUM(i.total) AS total_spent
FROM customer c

JOIN invoice i ON c.customer_id = i.customer_id GROUP BY c.customer_id
ORDER BY total_spent DESC

LIMIT 1;
```

-- Rock Music listeners --

```
SELECT DISTINCT c.email, c.first_name, c.last_name FROM customer c

JOIN invoice i ON c.customer_id = i.customer_id

JOIN invoice_line il ON i.invoice_id = il.invoice_id JOIN track t ON il.track_id
= t.track_id

JOIN genre g ON t.genre_id = g.genre_id WHERE g.name = 'Rock' ORDER
BY c.email;
```

-- Top 10 rock artists --

```
SELECT a.name AS artist_name, COUNT(t.track_id) AS track_count FROM
ARTIST_DATA a

JOIN ALBUMS_DATASET al ON a.ARTIST_ID = al.ARTIST JOIN track t
ON al.ALBUM_ID = t.album_id

JOIN genre g ON t.genre_id = g.genre_id WHERE g.name = 'Rock' GROUP
BY a.ARTIST_ID

ORDER BY track_count DESC LIMIT 10;
```

-- Tracks longer than average length --

```
SELECT name, milliseconds FROM track WHERE milliseconds > (SELECT
AVG(milliseconds) FROM track)

ORDER BY milliseconds DESC;
```

-- Amount spent by each customer on artists --

```
SELECT c.first_name, c.last_name, a.name AS artist_name,

SUM(il.unit_price * il.quantity) AS total_spent FROM customer c

JOIN invoice i ON c.customer_id = i.customer_id

JOIN invoice_line il ON i.invoice_id = il.invoice_id

JOIN track t ON il.track_id = t.track_id

JOIN ALBUMS_DATASET al ON t.album_id = al.ALBUM_ID

JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID

GROUP BY c.customer_id, a.ARTIST_ID

ORDER BY total_spent DESC;
```

-- Most popular music genre for each country --

```
WITH country_genre_sales AS (
    SELECT
        i.billing_country AS country,
        g.name AS genre_name,
        COUNT(il.invoice_line_id) AS purchase_count,
        RANK() OVER (PARTITION BY i.billing_country ORDER BY
COUNT(il.invoice_line_id) DESC) AS rank_num
    FROM invoice i
    JOIN invoice_line il ON i.invoice_id = il.invoice_id
```

```
    JOIN track t ON il.track_id = t.track_id
    JOIN genre g ON t.genre_id = g.genre_id
    GROUP BY i.billing_country, g.genre_id
)
SELECT country, genre_name, purchase_count
FROM country_genre_sales
WHERE rank_num = 1
ORDER BY country;
```

-- Top spending customer for each country --

```
WITH customer_spending AS (
    SELECT
        c.country,
        c.customer_id,
        c.first_name,
        c.last_name,
        SUM(i.total) AS total_spent,
        RANK() OVER (PARTITION BY c.country ORDER BY SUM(i.total)
DESC) AS rank_num
    FROM customer c
    JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.country, c.customer_id
)
SELECT country, first_name, last_name, total_spent
FROM customer_spending
```

```
WHERE rank_num = 1
ORDER BY country;
```

-- Most popular artists --

```
SELECT
    a.name AS artist_name,
    COUNT(il.invoice_line_id) AS tracks_sold,
    SUM(il.unit_price * il.quantity) AS total_revenue
FROM ARTIST_DATA a
JOIN ALBUMS_DATASET al ON a.ARTIST_ID = al.ARTIST
JOIN track t ON al.ALBUM_ID = t.album_id
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY a.ARTIST_ID
ORDER BY tracks_sold DESC
LIMIT 10;
```

-- Most popular song --

```
SELECT
    t.name AS track_name,
    a.name AS artist_name,
    COUNT(il.invoice_line_id) AS times_purchased,
    SUM(il.unit_price * il.quantity) AS total_revenue
FROM track t
JOIN ALBUMS_DATASET al ON t.album_id = al.ALBUM_ID
```

```
JOIN ARTIST_DATA a ON al.ARTIST = a.ARTIST_ID
JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY t.track_id
ORDER BY times_purchased DESC
LIMIT 1;
```

-- Average prices of different music types --

```
SELECT
    g.name AS genre,
    ROUND(AVG(t.unit_price), 2) AS avg_price,
    COUNT(t.track_id) AS total_tracks,
    SUM(il.unit_price * il.quantity) AS total_revenue
FROM genre g
JOIN track t ON g.genre_id = t.genre_id
LEFT JOIN invoice_line il ON t.track_id = il.track_id
GROUP BY g.genre_id
ORDER BY avg_price DESC;
```

-- Most popular countries for music purchases --

```
SELECT
    billing_country AS country,
    COUNT(*) AS total_invoices,
    SUM(total) AS total_revenue,
    ROUND(AVG(total), 2) AS avg_invoice_value,
```

```
    COUNT(DISTINCT customer_id) AS unique_customers
FROM invoice
GROUP BY billing_country
ORDER BY total_revenue DESC;
```