Music Store SQL Project

Table of Contents

1.Setup & Imports

2.Load CSVs and Prepare Database

3.Data Preview (Optional)

4.Easy Level Queries

5.Moderate Level Queries

6.Advanced Level Queries

7.Final Report & SQL Bundle

1) Setup & Imports

In [1]:
```
import os, sqlite3
import pandas as pd
from datetime import datetime
DATA_DIR = "/mnt/data"
```

2) Load CSVs and Prepare Database

- Reads all CSVs into DataFrames.
- Normalizes column names.
- Creates an in-memory SQLite DB.
- Derives a `levels` column for employee (seniority).

In [5]:
```
import os, sqlite3
import pandas as pd

# since notebook and CSVs are in the same folder
DATA_DIR = "."

def load_csv(filename):
    path = os.path.join(DATA_DIR, filename)
    df = pd.read_csv(path)
    df.columns = (
        df.columns.str.strip()
        .str.lower()
        .str.replace(" ", "_", regex=False)
        .str.replace("-", "_", regex=False)
    )
    return df

tables = {
    "employee": "employee.csv",
    "customer": "customer.csv",
    "invoice": "invoice.csv",
    "invoice_line": "invoice_line.csv",
    "track": "track.csv",
    "album": "album2.csv",
    "artist": "artist.csv",
    "genre": "genre.csv",
    "playlist": "playlist.csv",
    "playlist_track": "playlist_track.csv",
    "media_type": "media_type.csv",
}

dfs = {t: load_csv(f) for t, f in tables.items()}

con = sqlite3.connect(":memory:")
for t, df in dfs.items():
    for col in df.columns:
        if col in {"total","unit_price","quantity","milliseconds","bytes"
            df[col] = pd.to_numeric(df[col], errors="coerce")
    df.to_sql(t, con, index=False, if_exists="replace")

# Add 'levels' column for seniority if not present
emp_cols = pd.read_sql("PRAGMA table_info(employee);", con)["name"].str.l
if "levels" not in emp_cols:
    cur = con.cursor()
    cur.execute("ALTER TABLE employee ADD COLUMN levels INTEGER;")
    cur.execute("""
        UPDATE employee
        SET levels = CASE
            WHEN reports_to IS NULL THEN 3
            WHEN LOWER(COALESCE(title,'')) LIKE '%manager%' THEN 2
            ELSE 1
        END;
    """)
    con.commit()

print("Tables loaded:", ", ".join(dfs.keys()))
```

```
Tables loaded: employee, customer, invoice, invoice_line, track, album, artis
genre, playlist, playlist_track, media_type
```

3) Data Preview

```
In [6]: for t in ["employee", "customer", "invoice", "invoice_line", "track", "all
            print(f"\n=== {t} (first 5 rows) ===")
            display(pd.read_sql(f"SELECT * FROM {t} LIMIT 5;", con))
```

```
=== employee (first 5 rows) ===
```

	employee_id	last_name	first_name	title	reports_to	levels	birthdate	hire_date
0	1	Adams	Andrew	General Manager	9.0	L6	18-02-1962 00:00	14-08-2016 00:00
1	2	Edwards	Nancy	Sales Manager	1.0	L4	08-12-1958 00:00	01-05-2016 00:00
2	3	Peacock	Jane	Sales Support Agent	2.0	L1	29-08-1973 00:00	01-04-2017 00:00
3	4	Park	Margaret	Sales Support Agent	2.0	L1	19-09-1947 00:00	03-05-2017 00:00
4	5	Johnson	Steve	Sales Support Agent	2.0	L1	03-03-1965 00:00	17-10-2017 00:00

```
=== customer (first 5 rows) ===
```

	customer_id	first_name	last_name	company	address	city	state	country
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil
1	2	Leonie	Köhler	None	Theodor-Heuss-Straße 34	Stuttgart	None	Germany
2	3	François	Tremblay	None	1498 rue Bélanger	Montréal	QC	Canada
3	4	Bjørn	Hansen	None	Ullevålsveien 14	Oslo	None	Norway
4	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	None	Czech Republic

```
=== invoice (first 5 rows) ===
```

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_
0	1	18	2017-01-03 00:00:00	627 Broadway	New York	NY	
1	2	30	2017-01-03 00:00:00	230 Elgin Street	Ottawa	ON	
2	3	40	2017-01-05 00:00:00	8, Rue Hanovre	Paris	None	
3	4	18	2017-01-06 00:00:00	627 Broadway	New York	NY	
4	5	27	2017-01-07 00:00:00	1033 N Park Ave	Tucson	AZ	

```
=== invoice_line (first 5 rows) ===
```

	invoice_line_id	invoice_id	track_id	unit_price	quantity
0	1	1	1158	0.99	1
1	2	1	1159	0.99	1
2	3	1	1160	0.99	1
3	4	1	1161	0.99	1
4	5	1	1162	0.99	1

```
=== track (first 5 rows) ===
```

	track_id	name	album_id	media_type_id	genre_id	composer	milliseconds	byt
0	1	For Those About To Rock (We Salute You)	1	1	1	Angus Young, Malcolm Young, Brian Johnson	343719	111703
1	2	Balls to the Wall	2	2	1	None	342562	55104
2	3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U. Dirkscneider & W. Ho...	230619	39909
3	4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. D...	252051	43317
4	5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	62905

```
=== album (first 5 rows) ===
```

	album_id	title	artist_id
0	1	For Those About To Rock We Salute You	1
1	2	Balls to the Wall	2
2	3	Restless and Wild	2
3	4	Let There Be Rock	1
4	5	Big Ones	3

```
=== artist (first 5 rows) ===
```

	artist_id	name
0	1	AC/DC
1	2	Accept
2	3	Aerosmith
3	4	Alanis Morissette
4	5	Alice In Chains

```
=== genre (first 5 rows) ===
```

	genre_id	name
0	1	Rock
1	2	Jazz
2	3	Metal
3	4	Alternative & Punk
4	5	Rock And Roll

4) Easy Level Queries

Easy Q1 — Most senior employee (by `levels`)

In [7]:
```
sql = '''
SELECT employee_id, first_name, last_name, title, levels
FROM employee
ORDER BY levels DESC, employee_id ASC
LIMIT 1;
'''
pd.read_sql(sql, con)
```

Out[7]:

	employee_id	first_name	last_name	title	levels
0	9	Mohan	Madan	Senior General Manager	L7

Easy Q2 — Countries with the most invoices

In [8]:
```
sql = '''
SELECT billing_country AS country, COUNT(*) AS invoice_count
FROM invoice
GROUP BY billing_country
ORDER BY invoice_count DESC, country ASC;
'''
pd.read_sql(sql, con)
```

Out[8]:

	country	invoice_count
0	USA	131
1	Canada	76
2	Brazil	61
3	France	50
4	Germany	41
5	Czech Republic	30
6	Portugal	29
7	United Kingdom	28
8	India	21
9	Chile	13
10	Ireland	13
11	Finland	11
12	Spain	11
13	Australia	10
14	Denmark	10
15	Hungary	10
16	Netherlands	10
17	Poland	10
18	Sweden	10
19	Austria	9
20	Italy	9
21	Norway	9
22	Belgium	7
23	Argentina	5

Easy Q3 — Top 3 invoice totals

In [9]:
```
sql = '''
SELECT invoice_id, customer_id, total
FROM invoice
ORDER BY total DESC, invoice_id ASC
LIMIT 3;
'''
pd.read_sql(sql, con)
```

Out[9]:

	invoice_id	customer_id	total
0	183	42	23.76
1	31	3	19.80
2	92	32	19.80

Easy Q4 — City with highest total invoice amount

In [10]:
```
sql = '''
SELECT billing_city AS city, SUM(total) AS total_amount
FROM invoice
GROUP BY billing_city
ORDER BY total_amount DESC, city ASC
LIMIT 1;
'''
pd.read_sql(sql, con)
```

Out[10]:

	city	total_amount
0	Prague	273.24

Easy Q5 — Customer who has spent the most

In [11]:
```
sql = '''
SELECT c.customer_id,
       c.first_name || ' ' || c.last_name AS customer_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, customer_name
ORDER BY total_spent DESC, customer_name ASC
LIMIT 1;
'''
pd.read_sql(sql, con)
```

Out[11]:

	customer_id	customer_name	total_spent
0	5	František Wichterlová	144.54

5) Moderate Level Queries

Moderate Q1 — Customers who listen to Rock

In [12]:
```
sql = '''
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 LOWER(g.name) = 'rock'
ORDER BY c.email;
'''
pd.read_sql(sql, con)
```

Out[12]:

	email	first_name	last_name
0	aaronmitchell@yahoo.ca	Aaron	Mitchell
1	alero@uol.com.br	Alexandre	Rocha
2	astrid.gruber@apple.at	Astrid	Gruber
3	bjorn.hansen@yahoo.no	Bjørn	Hansen
4	camille.bernard@yahoo.fr	Camille	Bernard
5	daan_peeters@apple.be	Daan	Peeters
6	diego.gutierrez@yahoo.ar	Diego	Gutiérrez
7	dmiller@comcast.com	Dan	Miller
8	dominiquelefebvre@gmail.com	Dominique	Lefebvre
9	edfrancis@yachoo.ca	Edward	Francis
10	eduardo@woodstock.com.br	Eduardo	Martins
11	ellie.sullivan@shaw.ca	Ellie	Sullivan
12	emma_jones@hotmail.com	Emma	Jones
13	enrique_munoz@yahoo.es	Enrique	Muñoz
14	fernadaramos4@uol.com.br	Fernanda	Ramos
15	fharris@google.com	Frank	Harris
16	fralston@gmail.com	Frank	Ralston
17	frantisekw@jetbrains.com	František	Wichterlová
18	ftremblay@gmail.com	François	Tremblay
19	fzimmermann@yahoo.de	Fynn	Zimmermann
20	hannah.schneider@yahoo.de	Hannah	Schneider
21	hholy@gmail.com	Helena	Holý
22	hleacock@gmail.com	Heather	Leacock
23	hughoreilly@apple.ie	Hugh	O'Reilly
24	isabelle_mercier@apple.fr	Isabelle	Mercier
25	jacksmith@microsoft.com	Jack	Smith
26	jenniferp@rogers.ca	Jennifer	Peterson
27	jfernandes@yahoo.pt	João	Fernandes
28	joakim.johansson@yahoo.se	Joakim	Johansson
29	johavanderberg@yahoo.nl	Johannes	Van der Berg
30	johngordon22@yahoo.com	John	Gordon
31	jubarnett@gmail.com	Julia	Barnett
32	kachase@hotmail.com	Kathy	Chase
33	kara.nielsen@jubii.dk	Kara	Nielsen
34	ladislav_kovacs@apple.hu	Ladislav	Kovács
35	leonekohler@surfeu.de	Leonie	Köhler

Moderate Q2 — Top 10 Rock artists by track count

In [13]:
```
sql = '''
SELECT a.artist_id, a.name AS artist_name, COUNT(*) AS rock_track_count
FROM artist a
JOIN album al ON a.artist_id = al.artist_id
JOIN track t  ON al.album_id = t.album_id
JOIN genre g  ON t.genre_id = g.genre_id
WHERE LOWER(g.name) = 'rock'
GROUP BY a.artist_id, artist_name
ORDER BY rock_track_count DESC, artist_name ASC
LIMIT 10;
'''
pd.read_sql(sql, con)
```

Out[13]:

	artist_id	artist_name	rock_track_count
0	22	Led Zeppelin	114
1	150	U2	112
2	58	Deep Purple	92
3	90	Iron Maiden	81
4	118	Pearl Jam	54
5	152	Van Halen	52
6	51	Queen	45
7	142	The Rolling Stones	41
8	76	Creedence Clearwater Revival	40
9	52	Kiss	35

Moderate Q3 — Tracks longer than the average length

In [14]:
```
sql = '''
WITH avg_len AS (
    SELECT AVG(milliseconds) AS avg_ms FROM track
)
SELECT t.track_id, t.name, t.milliseconds
FROM track t, avg_len
WHERE t.milliseconds > avg_len.avg_ms
ORDER BY t.milliseconds DESC, t.name ASC;
'''
pd.read_sql(sql, con)
```

Out[14]:

	track_id	name	milliseconds
0	2820	Occupation / Precipice	5286953
1	3224	Through a Looking Glass	5088838
2	3244	Greetings from Earth, Pt. 1	2960293
3	3242	The Man With Nine Lives	2956998
4	3227	Battlestar Galactica, Pt. 2	2956081
...	...	...	...
489	1387	22 Acacia Avenue	395572
490	1864	The Unforgiven II	395520
491	1897	The Shortest Straw	395389
492	3413	Concerto for Clarinet in A Major, K. 622: II. ...	394482
493	806	Wicked Ways	393691

494 rows × 3 columns

6) Advanced Level Queries

Advanced Q1 — Amount each customer has spent on each artist

In [15]:
```
sql = '''
WITH per_line AS (
    SELECT i.customer_id,
           a.artist_id,
           (il.unit_price * il.quantity) AS line_revenue
    FROM invoice_line il
    JOIN invoice i ON il.invoice_id = i.invoice_id
    JOIN track t   ON il.track_id = t.track_id
    JOIN album al  ON t.album_id = al.album_id
    JOIN artist a  ON al.artist_id = a.artist_id
),
agg AS (
    SELECT customer_id, artist_id, ROUND(SUM(line_revenue), 2) AS amount_s
    FROM per_line
    GROUP BY customer_id, artist_id
)
SELECT c.customer_id,
       c.first_name || ' ' || c.last_name AS customer_name,
       a.name AS artist_name,
       amount_spent
FROM agg
JOIN customer c ON agg.customer_id = c.customer_id
JOIN artist a   ON agg.artist_id = a.artist_id
ORDER BY customer_name ASC, amount_spent DESC, artist_name ASC;
'''
pd.read_sql(sql, con)
```

Out[15]:

	customer_id	customer_name	artist_name	amount_spent
0	32	Aaron Mitchell	James Brown	19.80
1	32	Aaron Mitchell	Chris Cornell	13.86
2	32	Aaron Mitchell	Creedence Clearwater Revival	1.98
3	32	Aaron Mitchell	Men At Work	1.98
4	32	Aaron Mitchell	Nirvana	1.98
...	...	...	...	...
2184	42	Wyatt Girard	The Doors	0.99
2185	42	Wyatt Girard	The Rolling Stones	0.99
2186	42	Wyatt Girard	U2	0.99
2187	42	Wyatt Girard	UB40	0.99
2188	42	Wyatt Girard	Van Halen	0.99

2189 rows × 4 columns

Advanced Q2 — Most popular music genre for each country

In [16]:
```
sql = '''
WITH genre_counts AS (
    SELECT c.country,
           g.name AS genre_name,
           COUNT(*) AS purchase_count
    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, genre_name
),
ranked AS (
    SELECT country, genre_name, purchase_count,
           ROW_NUMBER() OVER (PARTITION BY country ORDER BY purchase_coun
    FROM genre_counts
)
SELECT country, genre_name AS top_genre, purchase_count
FROM ranked
WHERE rn = 1
ORDER BY country ASC;
'''
pd.read_sql(sql, con)
```

Out[16]:

	country	top_genre	purchase_count
0	Argentina	Alternative & Punk	17
1	Australia	Rock	34
2	Austria	Rock	40
3	Belgium	Rock	26
4	Brazil	Rock	205
5	Canada	Rock	333
6	Chile	Rock	61
7	Czech Republic	Rock	143
8	Denmark	Rock	24
9	Finland	Rock	46
10	France	Rock	211
11	Germany	Rock	194
12	Hungary	Rock	44
13	India	Rock	102
14	Ireland	Rock	72
15	Italy	Rock	35
16	Netherlands	Rock	33
17	Norway	Rock	40
18	Poland	Rock	40
19	Portugal	Rock	108
20	Spain	Rock	46
21	Sweden	Rock	60
22	USA	Rock	561
23	United Kingdom	Rock	166

Advanced Q3 — Top-spending customer for each country

In [17]:
```
sql = '''
WITH spend AS (
    SELECT c.country,
           c.customer_id,
           c.first_name || ' ' || c.last_name AS customer_name,
           SUM(i.total) AS total_spent
    FROM customer c
    JOIN invoice i ON c.customer_id = i.customer_id
    GROUP BY c.country, c.customer_id, customer_name
),
ranked AS (
    SELECT country, customer_id, customer_name, total_spent,
           ROW_NUMBER() OVER (PARTITION BY country ORDER BY total_spent DI
    FROM spend
)
SELECT country, customer_id, customer_name, ROUND(total_spent, 2) AS tota
FROM ranked
WHERE rn = 1
ORDER BY country ASC;
'''
pd.read_sql(sql, con)
```

Out[17]:

	country	customer_id	customer_name	total_spent
0	Argentina	56	Diego Gutiérrez	39.60
1	Australia	55	Mark Taylor	81.18
2	Austria	7	Astrid Gruber	69.30
3	Belgium	8	Daan Peeters	60.39
4	Brazil	1	Luís Gonçalves	108.90
5	Canada	3	François Tremblay	99.99
6	Chile	57	Luis Rojas	97.02
7	Czech Republic	5	František Wichterlová	144.54
8	Denmark	9	Kara Nielsen	37.62
9	Finland	44	Terhi Hämäläinen	79.20
10	France	42	Wyatt Girard	99.99
11	Germany	37	Fynn Zimmermann	94.05
12	Hungary	45	Ladislav Kovács	78.21
13	India	58	Manoj Pareek	111.87
14	Ireland	46	Hugh O'Reilly	114.84
15	Italy	47	Lucas Mancini	50.49
16	Netherlands	48	Johannes Van der Berg	65.34
17	Norway	4	Bjørn Hansen	72.27
18	Poland	49	Stanisław Wójcik	76.23
19	Portugal	34	João Fernandes	102.96
20	Spain	50	Enrique Muñoz	98.01
21	Sweden	51	Joakim Johansson	75.24
22	USA	17	Jack Smith	98.01
23	United Kingdom	53	Phil Hughes	98.01

7) Final Report & SQL Bundle

In [19]:
```
# assumes you already have:
# import os, pandas as pd
# from datetime import datetime
# and a live SQLite connection: con
# and DATA_DIR = "."

# Generate a markdown report and save all queries to a single .sql file.
report_lines = []
ts = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
report_lines.append("# Music Store SQL Project — Summary Report\n")
report_lines.append(f"_Generated on {ts}_\n")

# Highlights
df_q1 = pd.read_sql(
    "SELECT employee_id, first_name, last_name, title, levels "
    "FROM employee ORDER BY levels DESC, employee_id ASC LIMIT 1;", con)
df_q2 = pd.read_sql(
    "SELECT billing_country AS country, COUNT(*) AS invoice_count "
    "FROM invoice GROUP BY billing_country ORDER BY invoice_count DESC, c
df_q3 = pd.read_sql(
    "SELECT invoice_id, total FROM invoice ORDER BY total DESC, invoice_i
df_q4 = pd.read_sql(
    "SELECT billing_city AS city, SUM(total) AS total_amount "
    "FROM invoice GROUP BY billing_city ORDER BY total_amount DESC, city
df_q5 = pd.read_sql(
    "SELECT c.first_name || ' ' || c.last_name AS customer_name, SUM(i.to
    "FROM customer c JOIN invoice i ON c.customer_id = i.customer_id "
    "GROUP BY 1 ORDER BY total_spent DESC, customer_name ASC LIMIT 1;", c

if not df_q1.empty:
    r = df_q1.iloc[0]
    report_lines.append(f"- **Most senior employee:** {r['first_name']} {
if not df_q2.empty:
    r = df_q2.iloc[0]
    report_lines.append(f"- **Country with most invoices:** {r['country']
if not df_q3.empty:
    top3 = ", ".join([f"Invoice {int(i)}: {float(t):.2f}" for i, t in df_
    report_lines.append(f"- **Top 3 invoice totals:** {top3}")
if not df_q4.empty:
    r = df_q4.iloc[0]
    report_lines.append(f"- **Best promo city:** {r['city']} (Total amoun
if not df_q5.empty:
    r = df_q5.iloc[0]
    report_lines.append(f"- **Top spending customer:** {r['customer_name']

# Save report next to your notebook
report_path = os.path.join(DATA_DIR, "SQL_Project_Final_Report.md")
with open(report_path, "w", encoding="utf-8") as f:
    f.write("\n".join(report_lines))

# Save SQL bundle next to your notebook
sql_bundle = '''
-- EASY LEVEL
-- Q1
SELECT employee_id, first_name, last_name, title, levels
FROM employee
ORDER BY levels DESC, employee_id ASC
LIMIT 1;

-- Q2
```

```
Saved:
 - Report: .\SQL_Project_Final_Report.md
 - SQL   : .\SQL_Project_Queries.sql
```