Data Analytics Internship

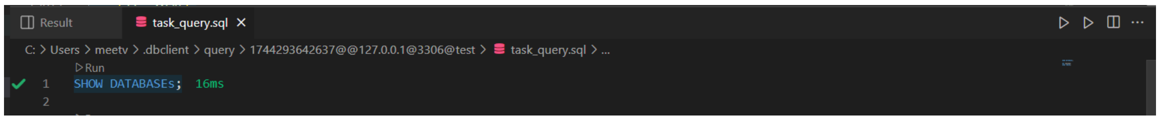
Day3 - SQL for Data Analysis

Dataset Used: Chinook SQL Query

1. Initial Setup / Validation Queries

1. 1. **SHOW DATABASES** to verify Chinook was created

sql:

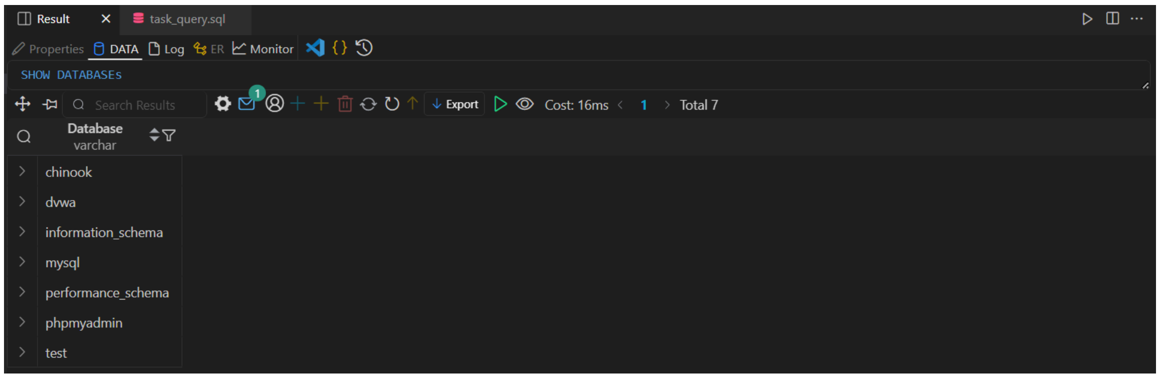


Result | task_query.sql

C: > Users > meetv > .dbclient > query > 1744293642637@@127.0.0.1@3306@test > task_query.sql > ...

```
SHOW DATABASES;  16ms
```

output:



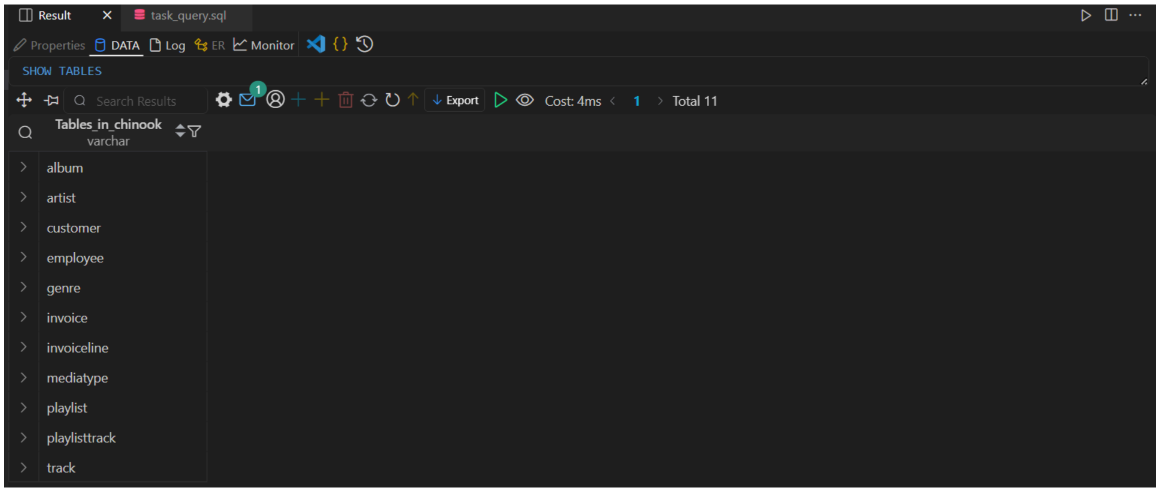
```
SHOW DATABASES
```

Cost: 16ms 1 Total 7

```
Database
varchar
chinook
dvwa
information_schema
mysql
performance_schema
phpmyadmin
test
```

2. 12. **USE Chinook** and **SHOW TABLES** to view all tables

sql:

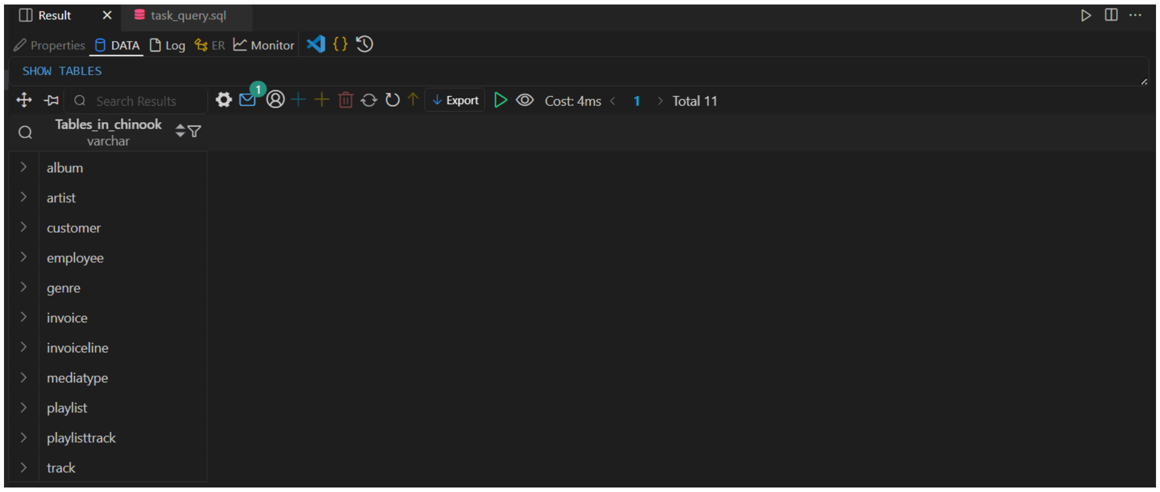


```
SHOW TABLES
```

```
Tables_in_chinook
varchar
album
artist
customer
employee
genre
invoice
invoiceline
mediatype
playlist
playlisttrack
track
```

Cost: 4ms Total 11

output:



```
SHOW TABLES
```

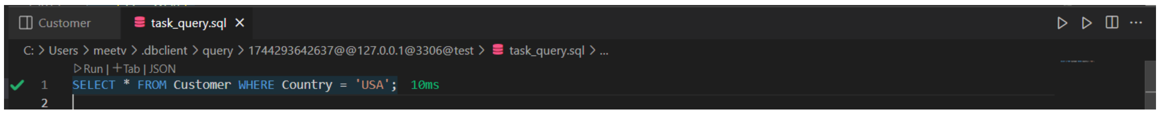
```
Tables_in_chinook
varchar
album
artist
customer
employee
genre
invoice
invoiceline
mediatype
playlist
playlisttrack
track
```

Cost: 4ms Total 11

2. Basic Queries (SELECT, WHERE, ORDER BY, GROUP BY)

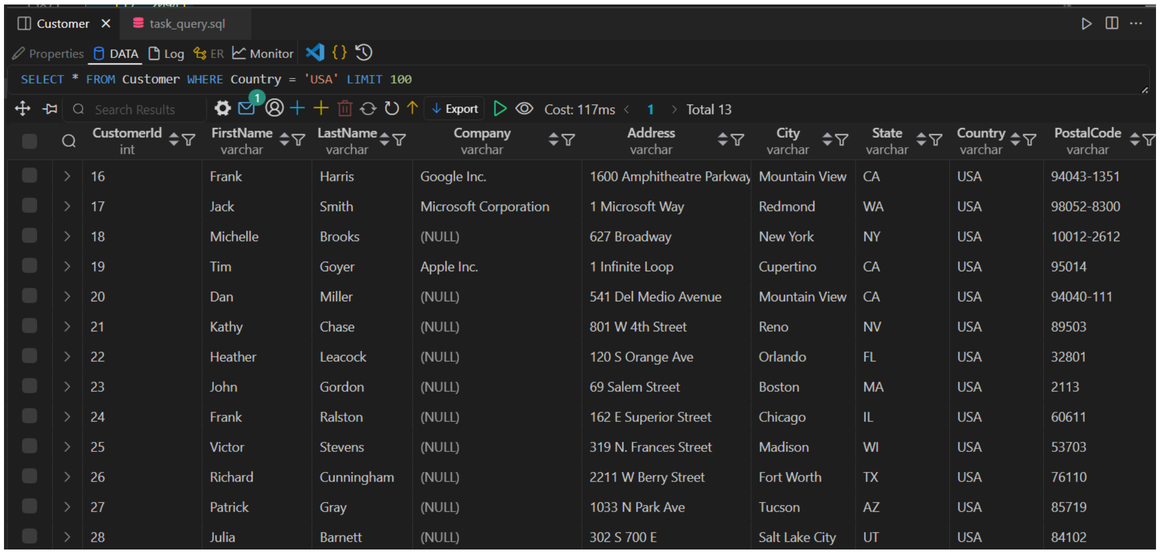
3. 1. List all customers from the USA

sql:


Customer
task_query.sql
C: > Users > meetv > .dbclient > query > 1744293642637@@127.0.0.1@3306@test > task_query.sql > ...
▷ Run | + Tab | JSON
10ms

```
SELECT * FROM Customer WHERE Country = 'USA';
```

output:

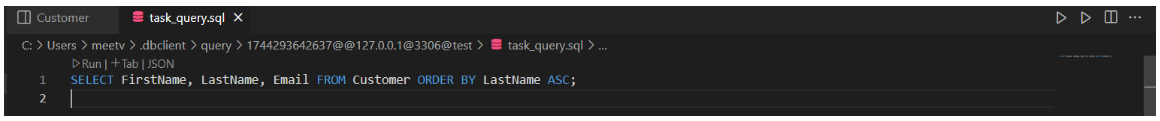


```
SELECT * FROM Customer WHERE Country = 'USA' LIMIT 100
```

CustomerId int	FirstName varchar	LastName varchar	Company varchar	Address varchar	City varchar	State varchar	Country varchar	PostalCode varchar
16	Frank	Harris	Google Inc.	1600 Amphitheatre Parkway	Mountain View	CA	USA	94043-1351
17	Jack	Smith	Microsoft Corporation	1 Microsoft Way	Redmond	WA	USA	98052-8300
18	Michelle	Brooks	(NULL)	627 Broadway	New York	NY	USA	10012-2612
19	Tim	Goyer	Apple Inc.	1 Infinite Loop	Cupertino	CA	USA	95014
20	Dan	Miller	(NULL)	541 Del Medio Avenue	Mountain View	CA	USA	94040-111
21	Kathy	Chase	(NULL)	801 W 4th Street	Reno	NV	USA	89503
22	Heather	Leacock	(NULL)	120 S Orange Ave	Orlando	FL	USA	32801
23	John	Gordon	(NULL)	69 Salem Street	Boston	MA	USA	2113
24	Frank	Ralston	(NULL)	162 E Superior Street	Chicago	IL	USA	60611
25	Victor	Stevens	(NULL)	319 N. Frances Street	Madison	WI	USA	53703
26	Richard	Cunningham	(NULL)	2211 W Berry Street	Fort Worth	TX	USA	76110
27	Patrick	Gray	(NULL)	1033 N Park Ave	Tucson	AZ	USA	85719
28	Julia	Barnett	(NULL)	302 S 700 E	Salt Lake City	UT	USA	84102

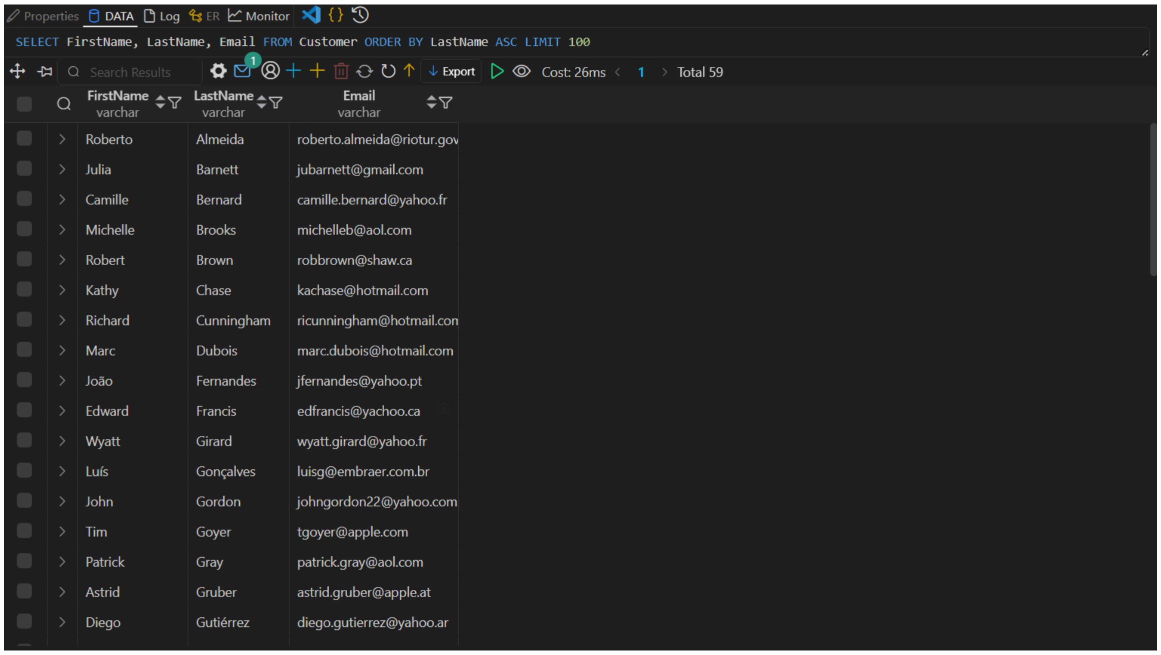
4. 2. Get names and emails of all customers, sorted by last name

sql:


Customer
task_query.sql
C: > Users > meetv > .dbclient > query > 1744293642637@@127.0.0.1@3306@test > task_query.sql > ...
▷ Run | + Tab | JSON

```
SELECT FirstName, LastName, Email FROM Customer ORDER BY LastName ASC;
```

output:



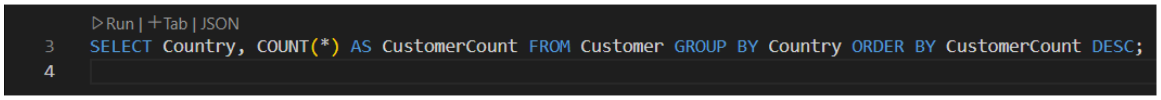
```
SELECT FirstName, LastName, Email FROM Customer ORDER BY LastName ASC LIMIT 100
```

Cost: 26ms 1 Total 59

FirstName varchar	LastName varchar	Email varchar
Roberto	Almeida	roberto.almeida@riotur.gov
Julia	Barnett	jubarnett@gmail.com
Camille	Bernard	camille.bernard@yahoo.fr
Michelle	Brooks	michelleb@aol.com
Robert	Brown	robbrown@shaw.ca
Kathy	Chase	kachase@hotmail.com
Richard	Cunningham	ricunningham@hotmail.con
Marc	Dubois	marc.dubois@hotmail.com
João	Fernandes	jfernandes@yahoo.pt
Edward	Francis	edfrancis@yachoo.ca
Wyatt	Girard	wyatt.girard@yahoo.fr
Luís	Gonçalves	luisg@embraer.com.br
John	Gordon	johngordon22@yahoo.com
Tim	Goyer	tgoyer@apple.com
Patrick	Gray	patrick.gray@aol.com
Astrid	Gruber	astrid.gruber@apple.at
Diego	Gutiérrez	diego.gutierrez@yahoo.ar

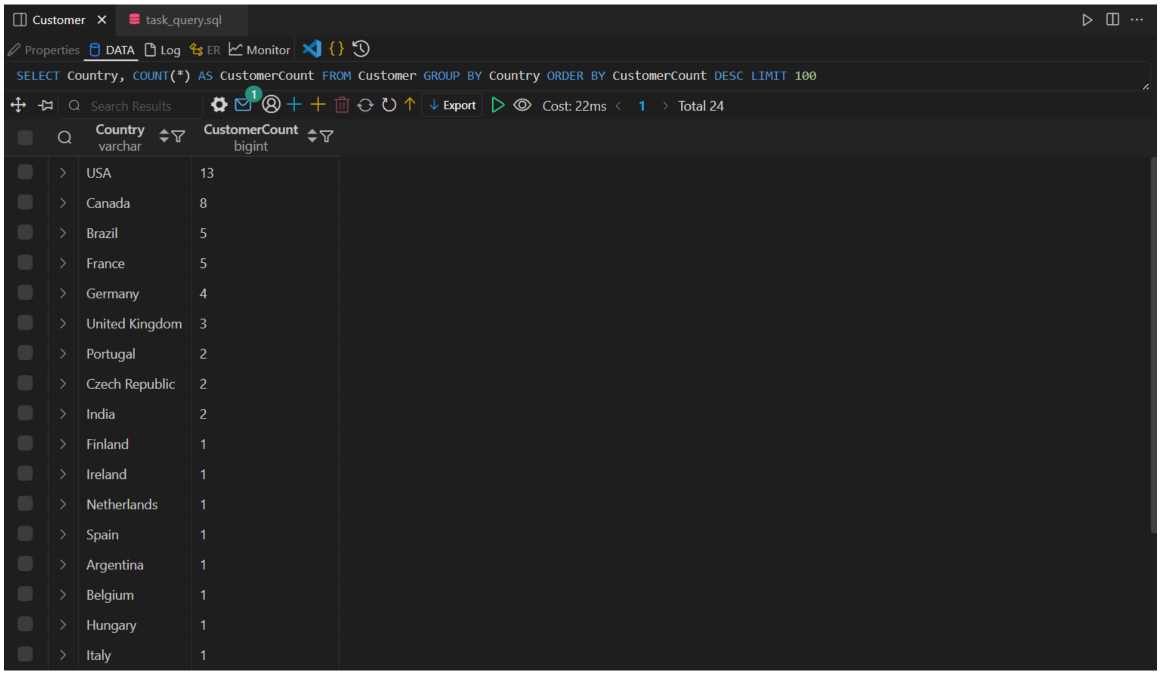
5. 3. Count the number of customers in each country

sql:



```
SELECT Country, COUNT(*) AS CustomerCount FROM Customer GROUP BY Country ORDER BY CustomerCount DESC;
```

output:



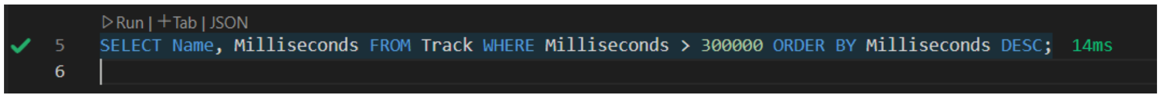
```
SELECT Country, COUNT(*) AS CustomerCount FROM Customer GROUP BY Country ORDER BY CustomerCount DESC LIMIT 100
```

Cost: 22ms 1 Total 24

Country varchar	CustomerCount bigint
USA	13
Canada	8
Brazil	5
France	5
Germany	4
United Kingdom	3
Portugal	2
Czech Republic	2
India	2
Finland	1
Ireland	1
Netherlands	1
Spain	1
Argentina	1
Belgium	1
Hungary	1
Italy	1

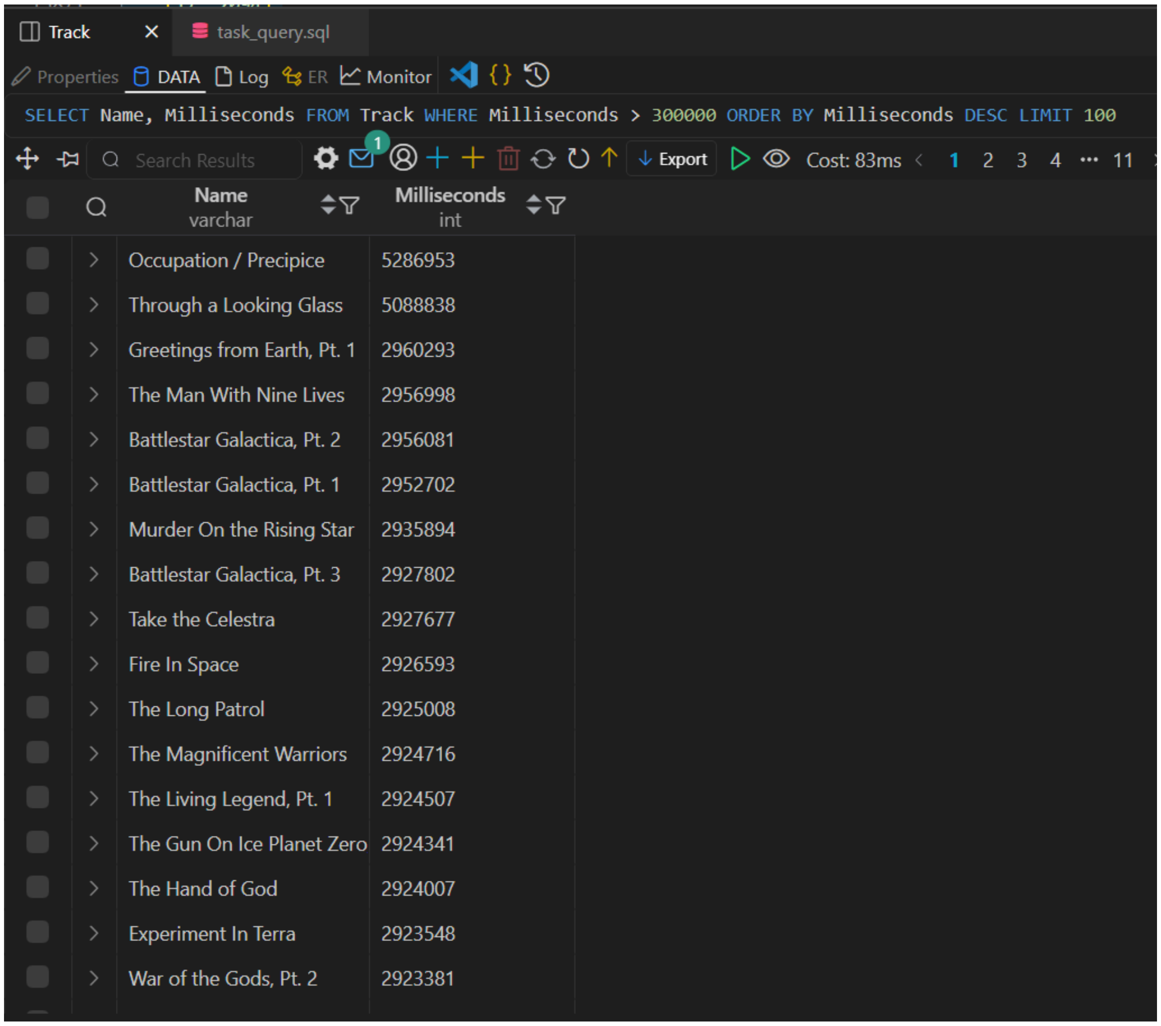
6. 4. List all tracks longer than 5 minutes

sql:



```
SELECT Name, Milliseconds FROM Track WHERE Milliseconds > 300000 ORDER BY Milliseconds DESC;  14ms
```

output:



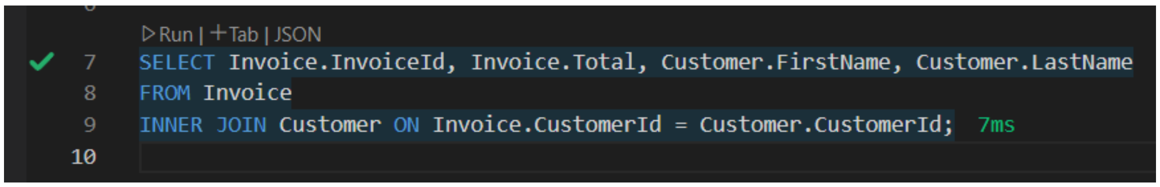
```
SELECT Name, Milliseconds FROM Track WHERE Milliseconds > 300000 ORDER BY Milliseconds DESC LIMIT 100
```

Name varchar	Milliseconds int
Occupation / Precipice	5286953
Through a Looking Glass	5088838
Greetings from Earth, Pt. 1	2960293
The Man With Nine Lives	2956998
Battlestar Galactica, Pt. 2	2956081
Battlestar Galactica, Pt. 1	2952702
Murder On the Rising Star	2935894
Battlestar Galactica, Pt. 3	2927802
Take the Celestra	2927677
Fire In Space	2926593
The Long Patrol	2925008
The Magnificent Warriors	2924716
The Living Legend, Pt. 1	2924507
The Gun On Ice Planet Zero	2924341
The Hand of God	2924007
Experiment In Terra	2923548
War of the Gods, Pt. 2	2923381

3. JOINS (INNER, LEFT, RIGHT)

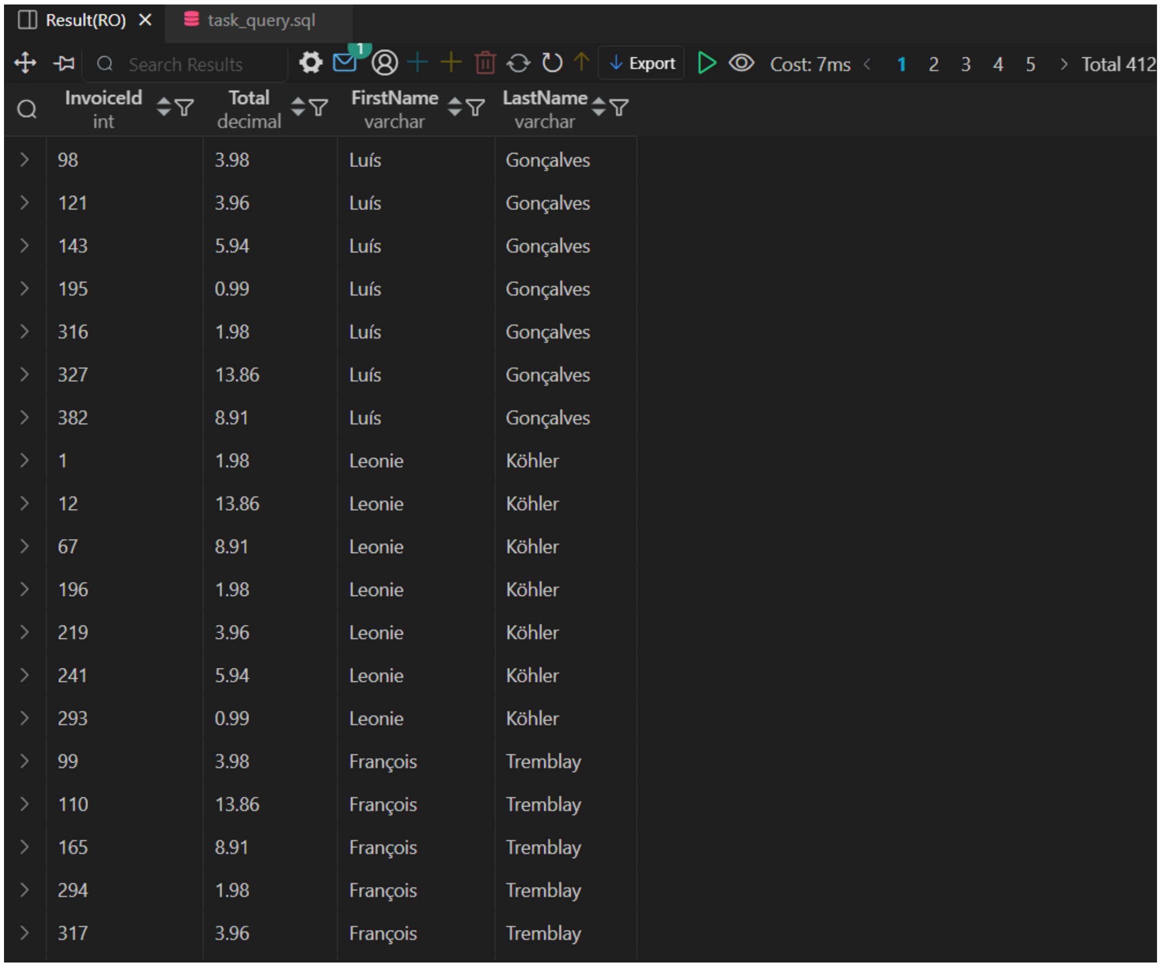
7. 5. List invoice details with customer names

sql:



```
SELECT Invoice.InvoiceId, Invoice.Total, Customer.FirstName, Customer.LastName
FROM Invoice
INNER JOIN Customer ON Invoice.CustomerId = Customer.CustomerId;
```

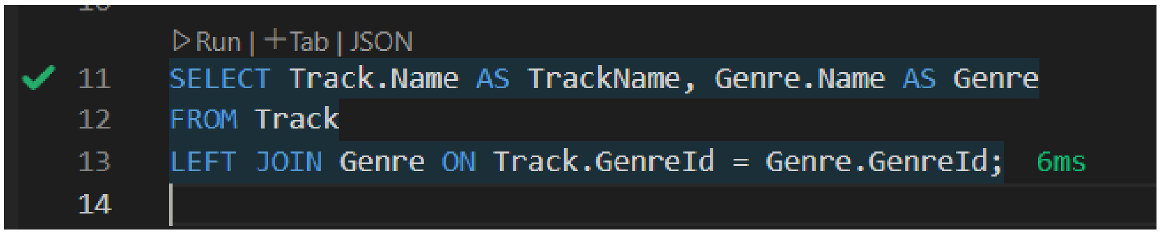
output:



InvoiceId int	Total decimal	FirstName varchar	LastName varchar
98	3.98	Luís	Gonçalves
121	3.96	Luís	Gonçalves
143	5.94	Luís	Gonçalves
195	0.99	Luís	Gonçalves
316	1.98	Luís	Gonçalves
327	13.86	Luís	Gonçalves
382	8.91	Luís	Gonçalves
1	1.98	Leonie	Köhler
12	13.86	Leonie	Köhler
67	8.91	Leonie	Köhler
196	1.98	Leonie	Köhler
219	3.96	Leonie	Köhler
241	5.94	Leonie	Köhler
293	0.99	Leonie	Köhler
99	3.98	François	Tremblay
110	13.86	François	Tremblay
165	8.91	François	Tremblay
294	1.98	François	Tremblay
317	3.96	François	Tremblay

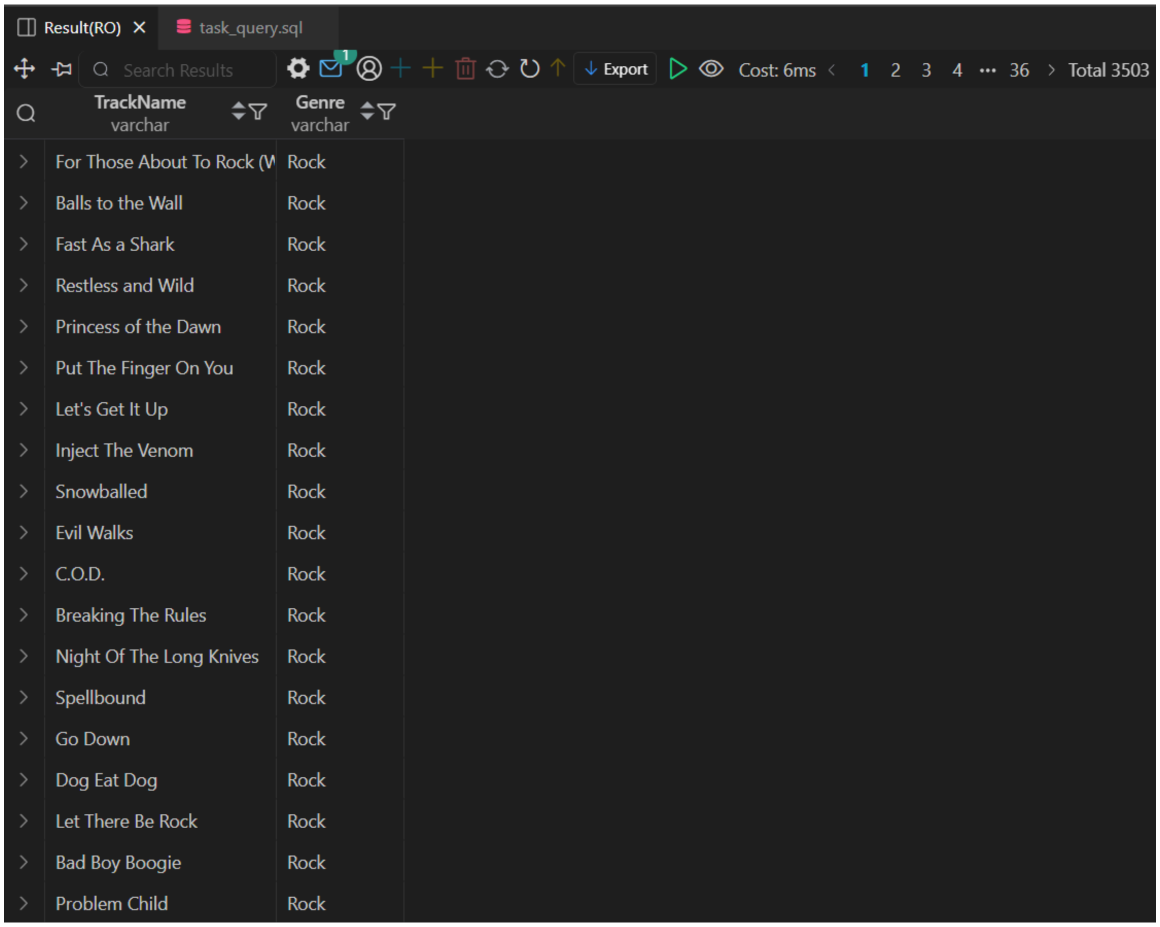
8. 6. Show each track with its genre name (LEFT JOIN)

sql:



```
SELECT Track.Name AS TrackName, Genre.Name AS Genre
FROM Track
LEFT JOIN Genre ON Track.GenreId = Genre.GenreId;
```

output:

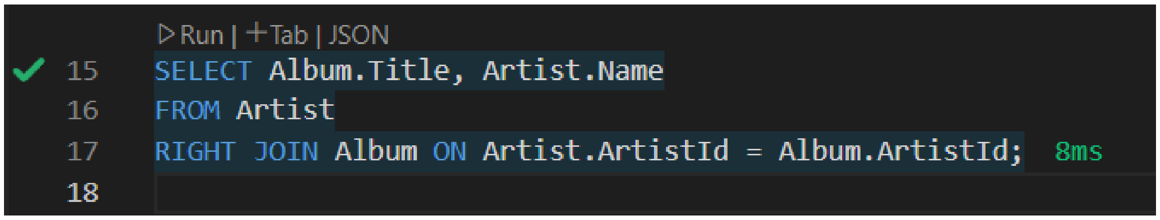

Result(RO) task_query.sql

Search Results Export Cost: 6ms 1 2 3 4 ··· 36 Total 3503

TrackName varchar	Genre varchar
For Those About To Rock (W	Rock
Balls to the Wall	Rock
Fast As a Shark	Rock
Restless and Wild	Rock
Princess of the Dawn	Rock
Put The Finger On You	Rock
Let's Get It Up	Rock
Inject The Venom	Rock
Snowballed	Rock
Evil Walks	Rock
C.O.D.	Rock
Breaking The Rules	Rock
Night Of The Long Knives	Rock
Spellbound	Rock
Go Down	Rock
Dog Eat Dog	Rock
Let There Be Rock	Rock
Bad Boy Boogie	Rock
Problem Child	Rock

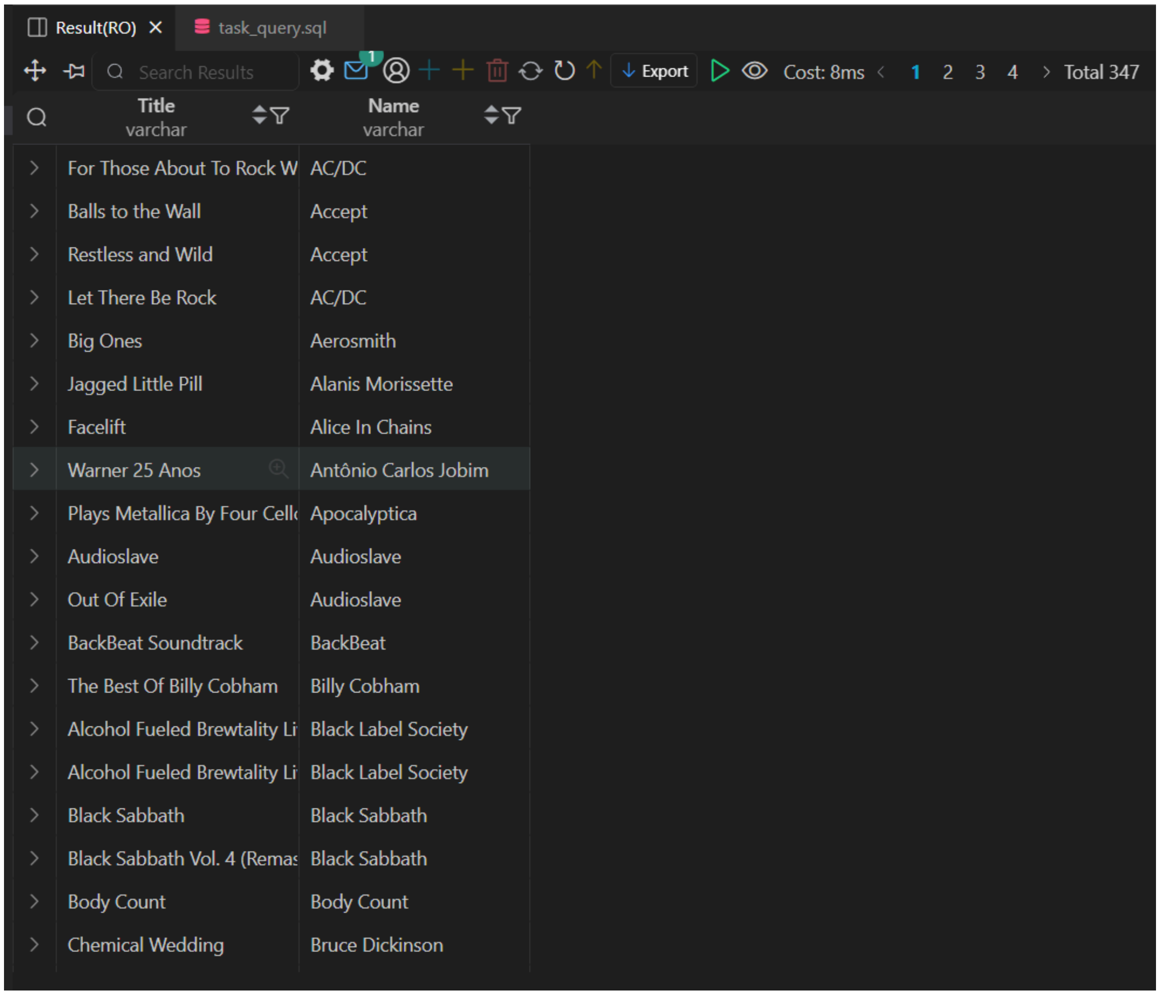
9. 7. Get all albums with their artists (RIGHT JOIN)

sql:


▷ Run | + Tab | JSON

```sql
SELECT Album.Title, Artist.Name
FROM Artist
RIGHT JOIN Album ON Artist.ArtistId = Album.ArtistId;   8ms
```

output:


Result(RO) task_query.sql

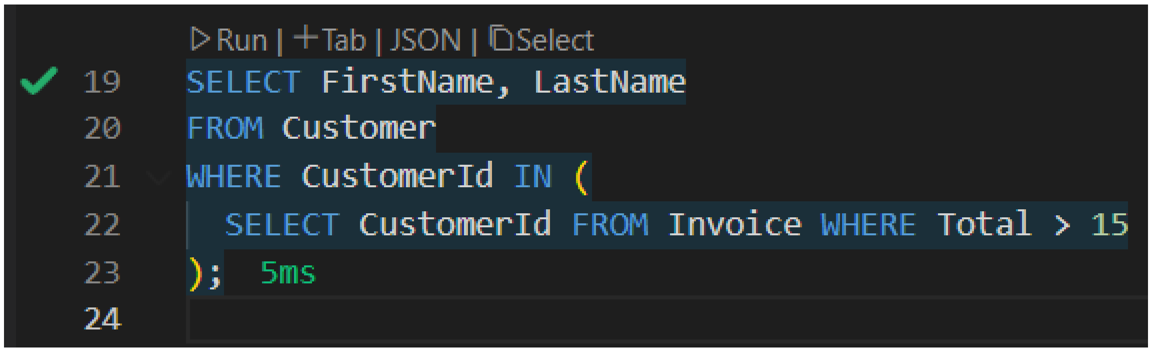
Export Cost: 8ms 1 2 3 4 Total 347

Title varchar	Name varchar
For Those About To Rock W	AC/DC
Balls to the Wall	Accept
Restless and Wild	Accept
Let There Be Rock	AC/DC
Big Ones	Aerosmith
Jagged Little Pill	Alanis Morissette
Facelift	Alice In Chains
Warner 25 Anos	Antônio Carlos Jobim
Plays Metallica By Four Cell	Apocalyptica
Audioslave	Audioslave
Out Of Exile	Audioslave
BackBeat Soundtrack	BackBeat
The Best Of Billy Cobham	Billy Cobham
Alcohol Fueled Brewtality Li	Black Label Society
Alcohol Fueled Brewtality Li	Black Label Society
Black Sabbath	Black Sabbath
Black Sabbath Vol. 4 (Remas	Black Sabbath
Body Count	Body Count
Chemical Wedding	Bruce Dickinson

4. Subqueries

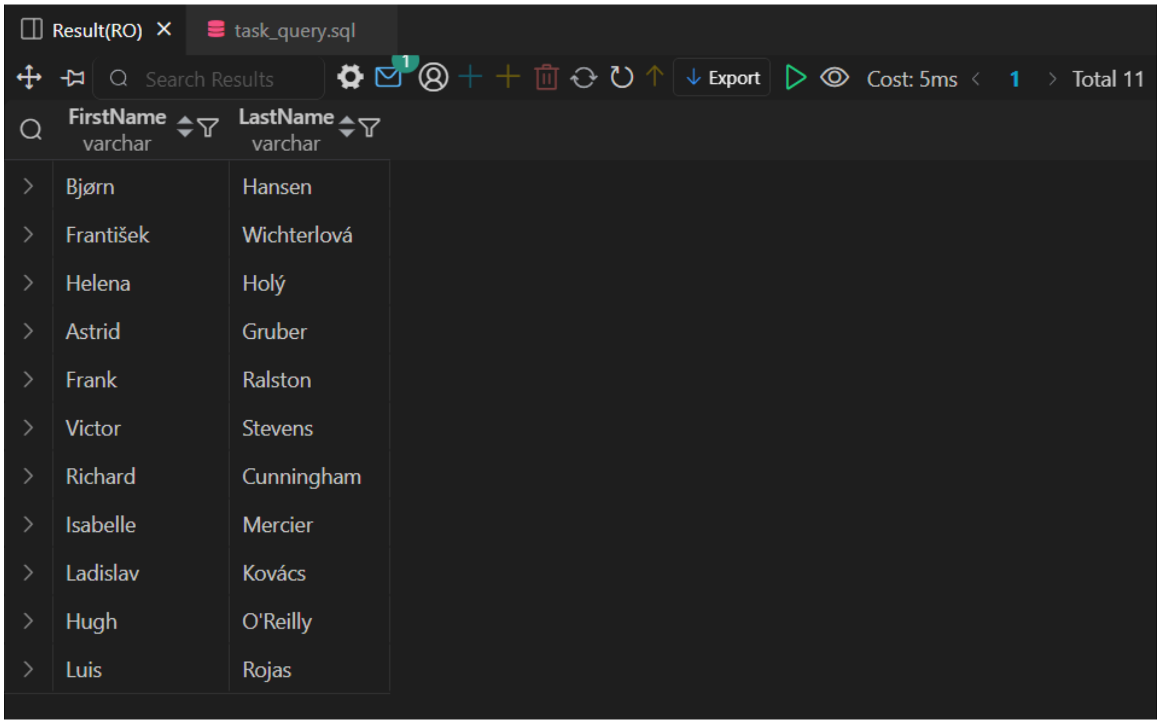
10. 8. Find all customers who made invoices over $15

sql:


▷ Run | + Tab | JSON | Select

```
SELECT FirstName, LastName
FROM Customer
WHERE CustomerId IN (
  SELECT CustomerId FROM Invoice WHERE Total > 15
);  5ms
```

output:



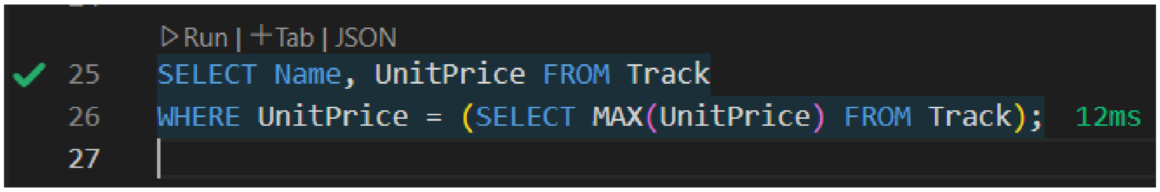
Result(RO) task_query.sql

Search Results Export Cost: 5ms 1 Total 11

FirstName varchar	LastName varchar
Bjørn	Hansen
František	Wichterlová
Helena	Holý
Astrid	Gruber
Frank	Ralston
Victor	Stevens
Richard	Cunningham
Isabelle	Mercier
Ladislav	Kovács
Hugh	O'Reilly
Luis	Rojas

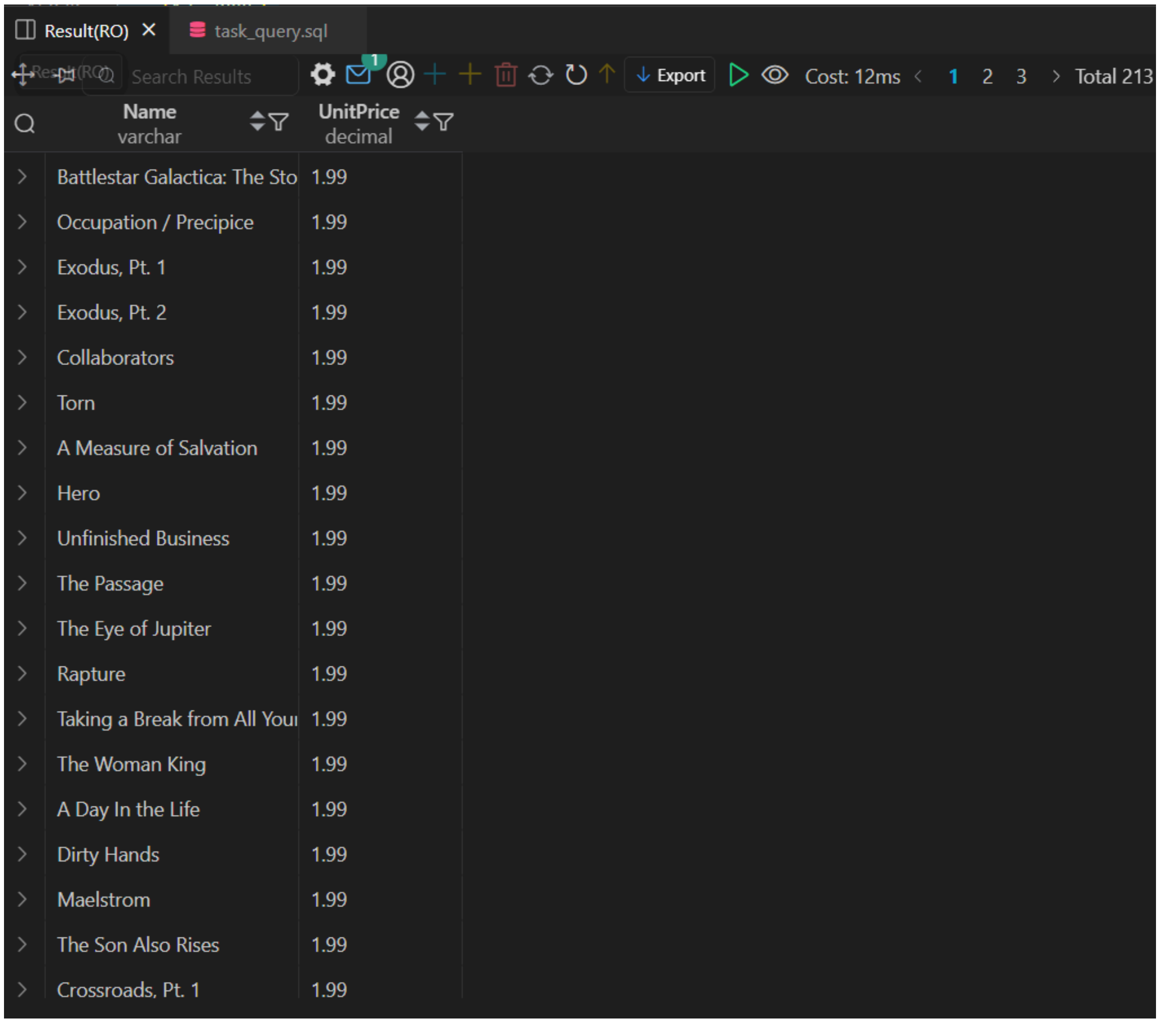
11. 9. Find the track with the highest unit price

sql:



```
SELECT Name, UnitPrice FROM Track
WHERE UnitPrice = (SELECT MAX(UnitPrice) FROM Track);   12ms
```

output:

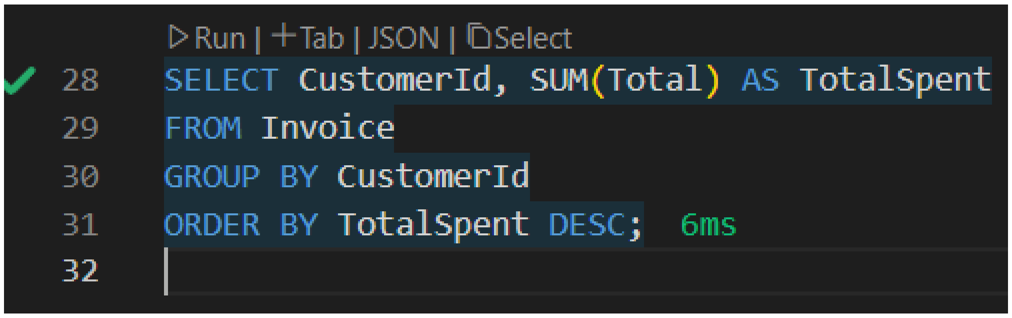


Name varchar	UnitPrice decimal
Battlestar Galactica: The Sto	1.99
Occupation / Precipice	1.99
Exodus, Pt. 1	1.99
Exodus, Pt. 2	1.99
Collaborators	1.99
Torn	1.99
A Measure of Salvation	1.99
Hero	1.99
Unfinished Business	1.99
The Passage	1.99
The Eye of Jupiter	1.99
Rapture	1.99
Taking a Break from All Your	1.99
The Woman King	1.99
A Day In the Life	1.99
Dirty Hands	1.99
Maelstrom	1.99
The Son Also Rises	1.99
Crossroads, Pt. 1	1.99

5. Aggregate Functions (SUM, AVG, COUNT)

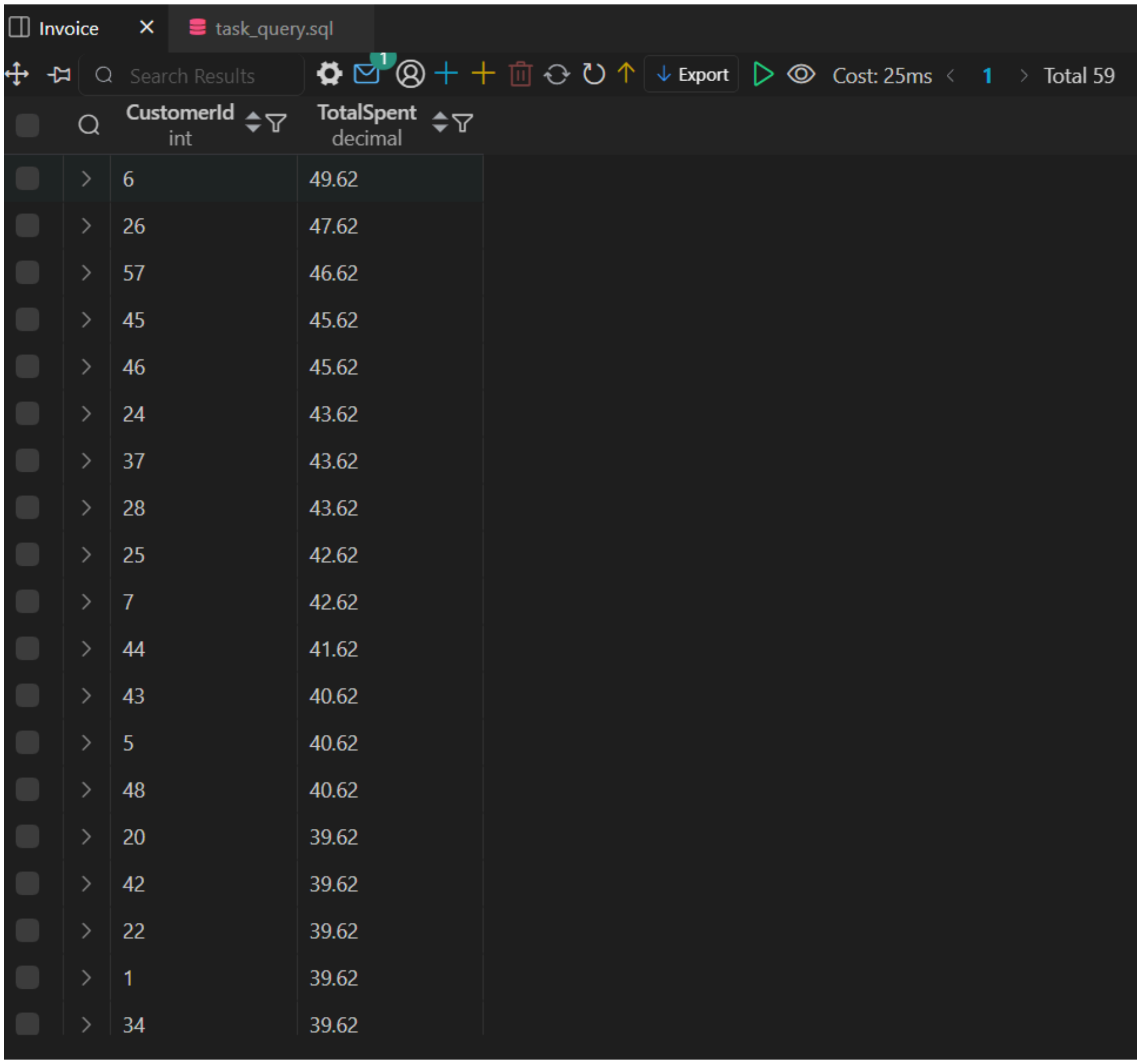
12. 10. Total sales per customer

sql:



```sql
SELECT CustomerId, SUM(Total) AS TotalSpent
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalSpent DESC;
```

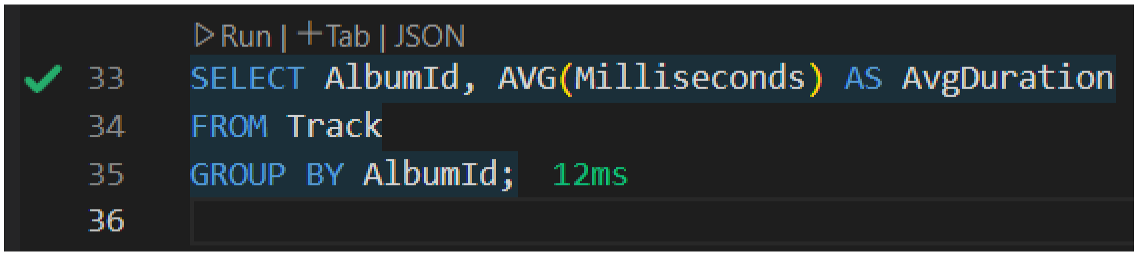
output:



CustomerId int	TotalSpent decimal
6	49.62
26	47.62
57	46.62
45	45.62
46	45.62
24	43.62
37	43.62
28	43.62
25	42.62
7	42.62
44	41.62
43	40.62
5	40.62
48	40.62
20	39.62
42	39.62
22	39.62
1	39.62
34	39.62

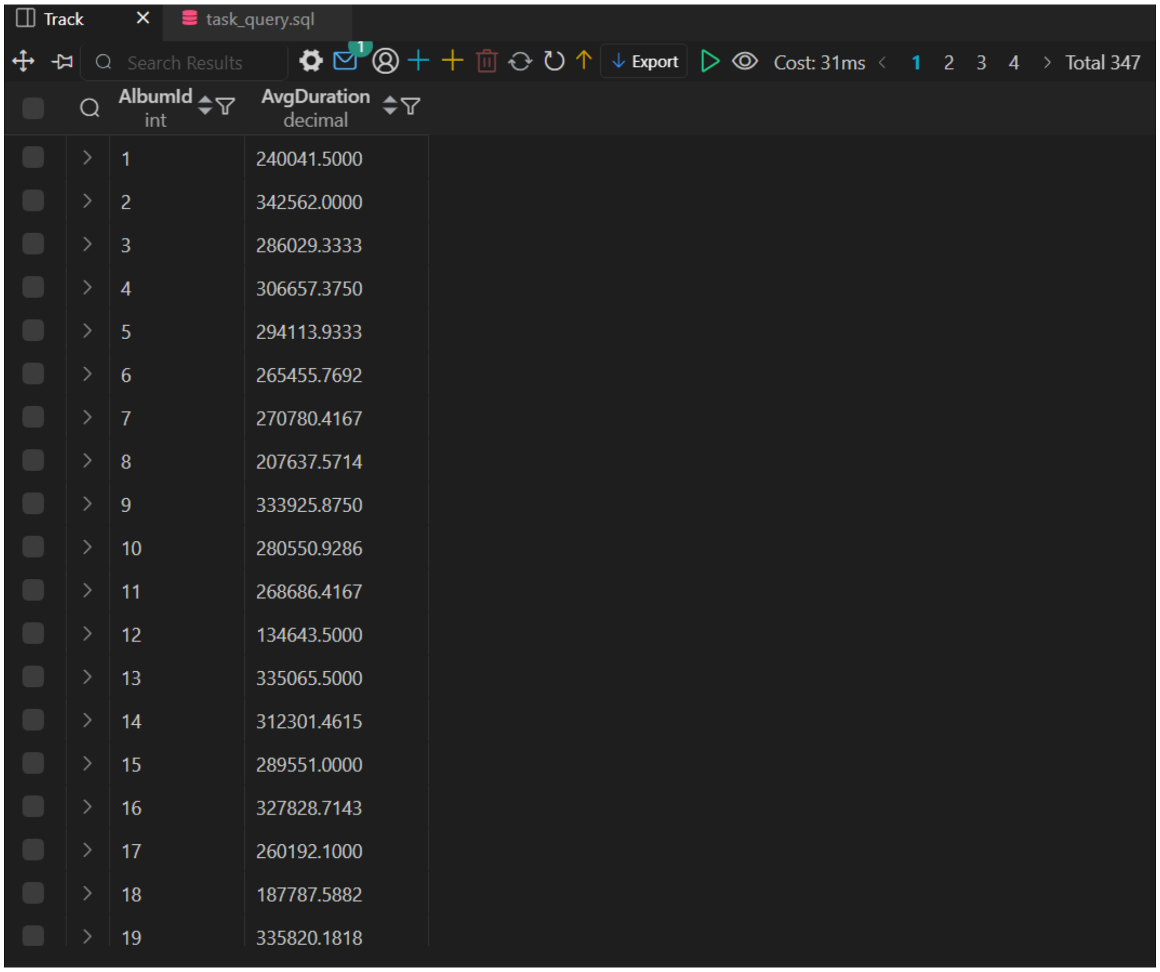
13. 11. Average track duration per album

sql:



```sql
SELECT AlbumId, AVG(Milliseconds) AS AvgDuration
FROM Track
GROUP BY AlbumId;
```

output:

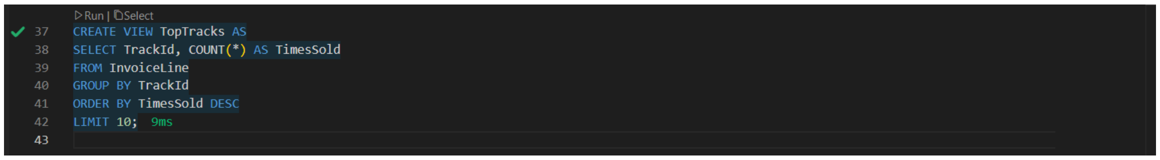


AlbumId int	AvgDuration decimal
1	240041.5000
2	342562.0000
3	286029.3333
4	306657.3750
5	294113.9333
6	265455.7692
7	270780.4167
8	207637.5714
9	333925.8750
10	280550.9286
11	268686.4167
12	134643.5000
13	335065.5000
14	312301.4615
15	289551.0000
16	327828.7143
17	260192.1000
18	187787.5882
19	335820.1818

6. Views for Analysis

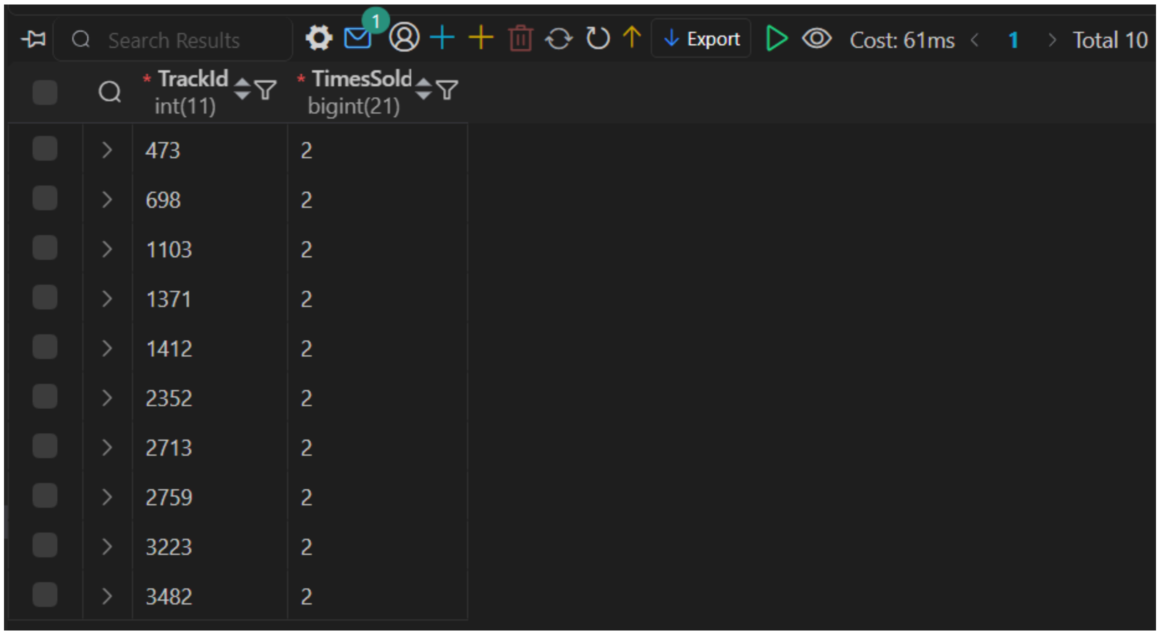
14. 12. Create a view of top 10 best-selling tracks

sql:



```
CREATE VIEW TopTracks AS
SELECT TrackId, COUNT(*) AS TimesSold
FROM InvoiceLine
GROUP BY TrackId
ORDER BY TimesSold DESC
LIMIT 10;
```

output:



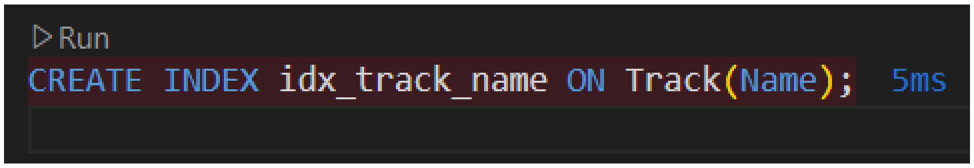
Search Results | Export | Cost: 61ms | 1 | Total 10

TrackId int(11)	TimesSold bigint(21)
473	2
698	2
1103	2
1371	2
1412	2
2352	2
2713	2
2759	2
3223	2
3482	2

7. Indexes & Optimization

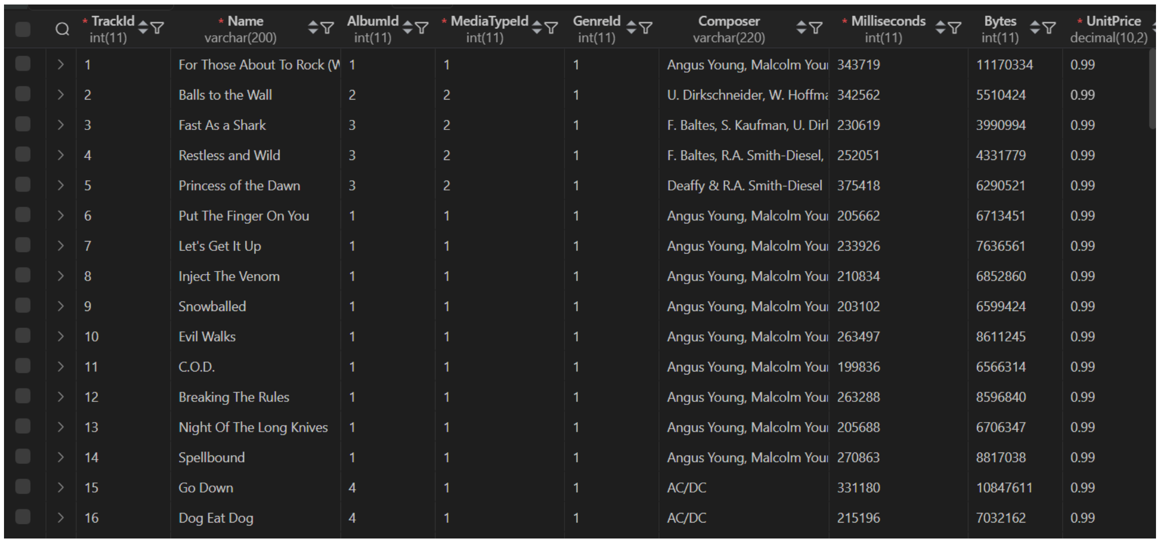
15. 13. Create an index on Track.Name to speed up search

sql:



```
▷ Run
CREATE INDEX idx_track_name ON Track(Name);   5ms
```

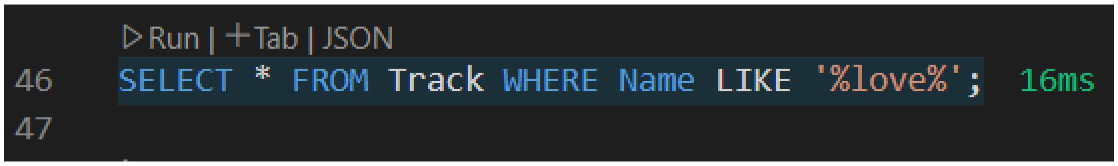
output:



TrackId int(11)	Name varchar(200)	AlbumId int(11)	MediaTypeId int(11)	GenreId int(11)	Composer varchar(220)	Milliseconds int(11)	Bytes int(11)	UnitPrice decimal(10,2)
1	For Those About To Rock (W	1	1	1	Angus Young, Malcolm Your	343719	11170334	0.99
2	Balls to the Wall	2	2	1	U. Dirkschneider, W. Hoffma	342562	5510424	0.99
3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U. Dirl	230619	3990994	0.99
4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel,	252051	4331779	0.99
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99
6	Put The Finger On You	1	1	1	Angus Young, Malcolm Your	205662	6713451	0.99
7	Let's Get It Up	1	1	1	Angus Young, Malcolm Your	233926	7636561	0.99
8	Inject The Venom	1	1	1	Angus Young, Malcolm Your	210834	6852860	0.99
9	Snowballed	1	1	1	Angus Young, Malcolm Your	203102	6599424	0.99
10	Evil Walks	1	1	1	Angus Young, Malcolm Your	263497	8611245	0.99
11	C.O.D.	1	1	1	Angus Young, Malcolm Your	199836	6566314	0.99
12	Breaking The Rules	1	1	1	Angus Young, Malcolm Your	263288	8596840	0.99
13	Night Of The Long Knives	1	1	1	Angus Young, Malcolm Your	205688	6706347	0.99
14	Spellbound	1	1	1	Angus Young, Malcolm Your	270863	8817038	0.99
15	Go Down	4	1	1	AC/DC	331180	10847611	0.99
16	Dog Eat Dog	4	1	1	AC/DC	215196	7032162	0.99

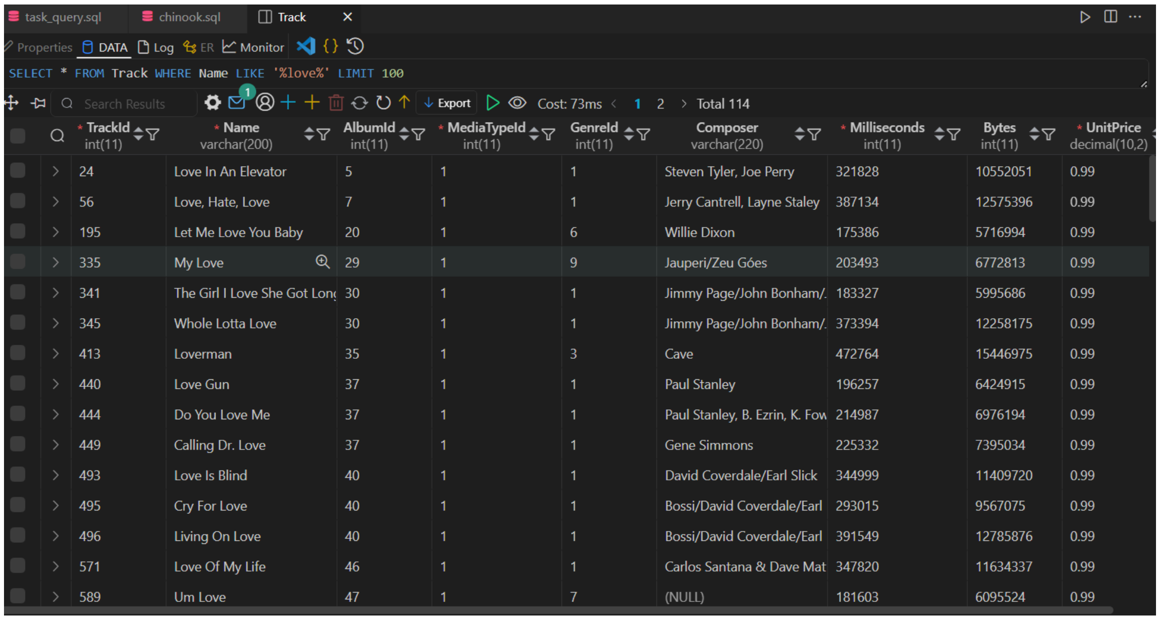
16. 14. Analyze slow query and optimize with index

sql:



```sql
SELECT * FROM Track WHERE Name LIKE '%love%';  16ms
```

output:



```sql
SELECT * FROM Track WHERE Name LIKE '%love%' LIMIT 100
```

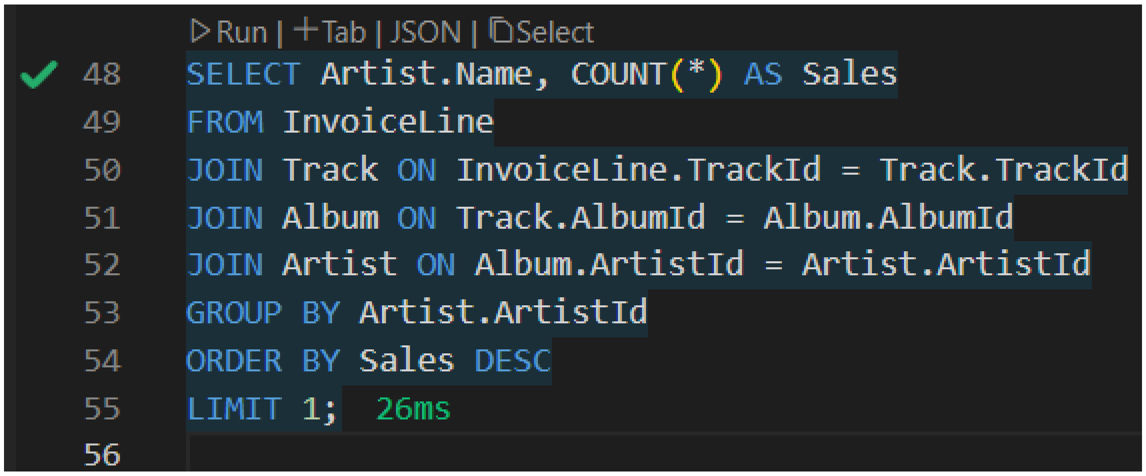
Cost: 73ms 1 2 Total 114

TrackId int(11)	Name varchar(200)	AlbumId int(11)	MediaTypeId int(11)	GenreId int(11)	Composer varchar(220)	Milliseconds int(11)	Bytes int(11)	UnitPrice decimal(10,2)
24	Love In An Elevator	5	1	1	Steven Tyler, Joe Perry	321828	10552051	0.99
56	Love, Hate, Love	7	1	1	Jerry Cantrell, Layne Staley	387134	12575396	0.99
195	Let Me Love You Baby	20	1	6	Willie Dixon	175386	5716994	0.99
335	My Love	29	1	9	Jauperi/Zeu Góes	203493	6772813	0.99
341	The Girl I Love She Got Long	30	1	1	Jimmy Page/John Bonham/.	183327	5995686	0.99
345	Whole Lotta Love	30	1	1	Jimmy Page/John Bonham/.	373394	12258175	0.99
413	Loverman	35	1	3	Cave	472764	15446975	0.99
440	Love Gun	37	1	1	Paul Stanley	196257	6424915	0.99
444	Do You Love Me	37	1	1	Paul Stanley, B. Ezrin, K. Fow	214987	6976194	0.99
449	Calling Dr. Love	37	1	1	Gene Simmons	225332	7395034	0.99
493	Love Is Blind	40	1	1	David Coverdale/Earl Slick	344999	11409720	0.99
495	Cry For Love	40	1	1	Bossi/David Coverdale/Earl	293015	9567075	0.99
496	Living On Love	40	1	1	Bossi/David Coverdale/Earl	391549	12785876	0.99
571	Love Of My Life	46	1	1	Carlos Santana & Dave Mat	347820	11634337	0.99
589	Um Love	47	1	7	(NULL)	181603	6095524	0.99

8. Extra Queries

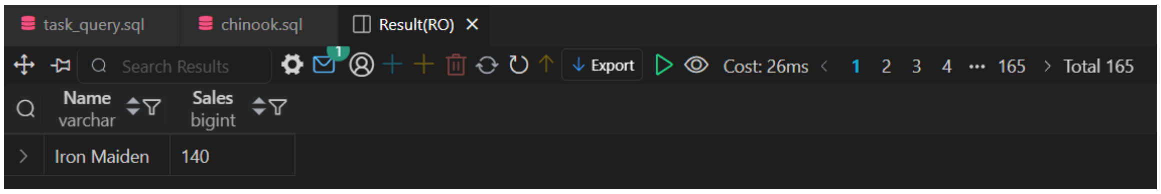
17. 15. Most popular artist based on track sales

sql:



```
SELECT Artist.Name, COUNT(*) AS Sales
FROM InvoiceLine
JOIN Track ON InvoiceLine.TrackId = Track.TrackId
JOIN Album ON Track.AlbumId = Album.AlbumId
JOIN Artist ON Album.ArtistId = Artist.ArtistId
GROUP BY Artist.ArtistId
ORDER BY Sales DESC
LIMIT 1;  26ms
```

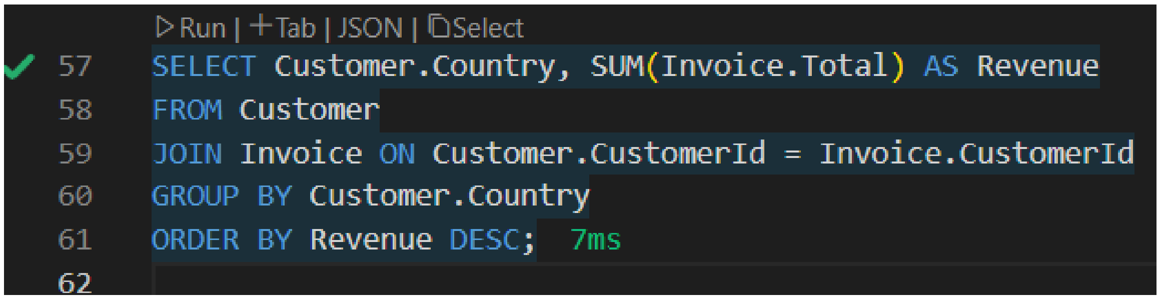
output:



```
task_query.sql   chinook.sql   Result(RO)
Search Results   Export   Cost: 26ms   1 2 3 4 ... 165   Total 165
Name      Sales
varchar   bigint
Iron Maiden   140
```

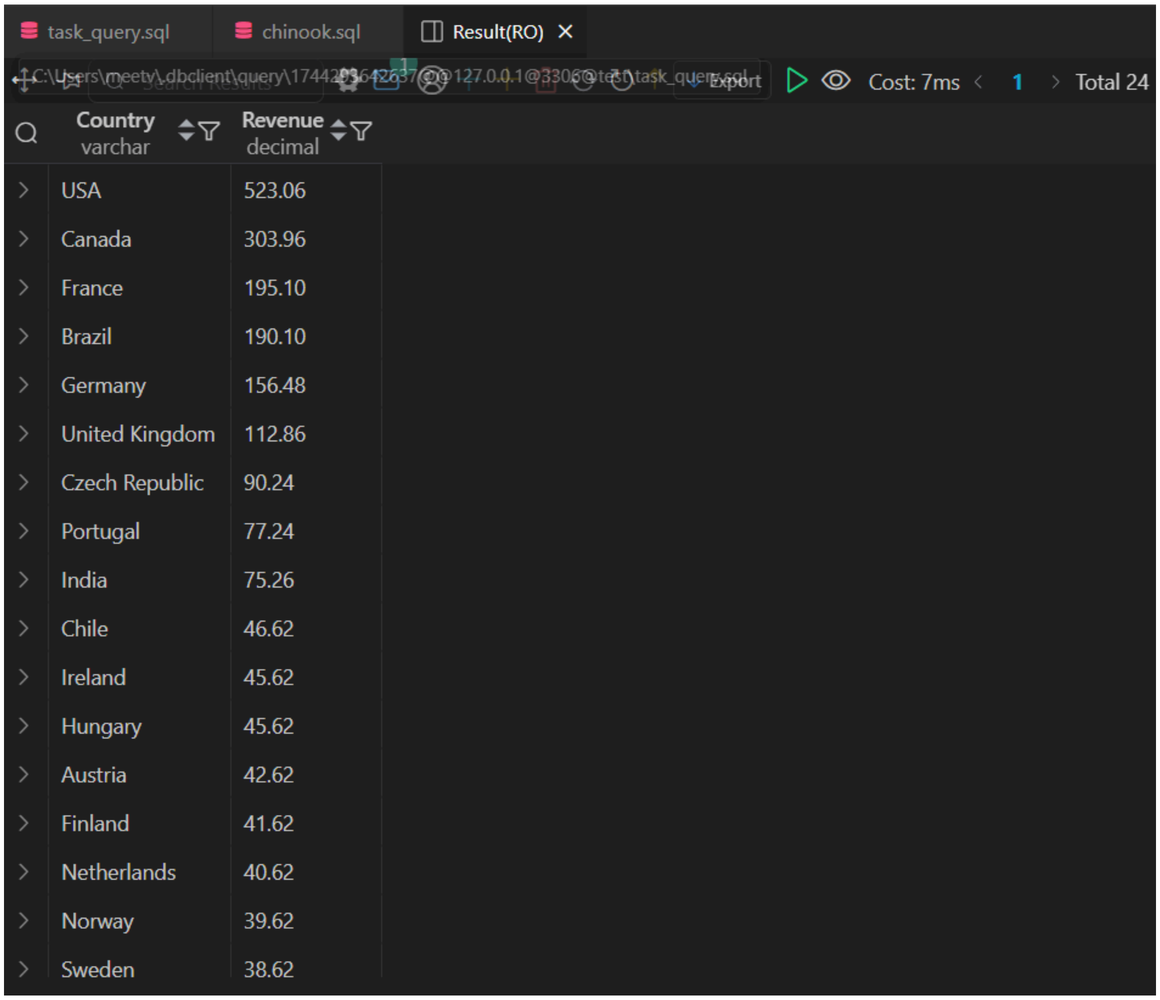
18. 16. Revenue by country

sql:



```
SELECT Customer.Country, SUM(Invoice.Total) AS Revenue
FROM Customer
JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY Customer.Country
ORDER BY Revenue DESC;  7ms
```

output:



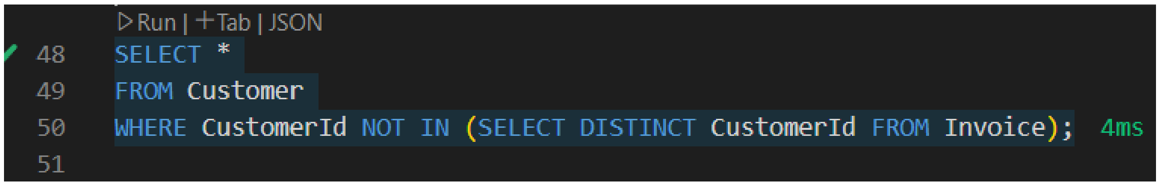
task_query.sql | chinook.sql | Result(RO)

Cost: 7ms 1 Total 24

Country varchar	Revenue decimal
USA	523.06
Canada	303.96
France	195.10
Brazil	190.10
Germany	156.48
United Kingdom	112.86
Czech Republic	90.24
Portugal	77.24
India	75.26
Chile	46.62
Ireland	45.62
Hungary	45.62
Austria	42.62
Finland	41.62
Netherlands	40.62
Norway	39.62
Sweden	38.62

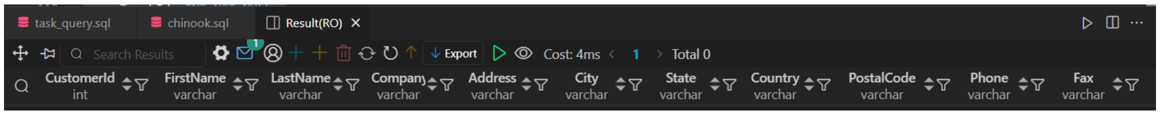
1. 19. List customers who haven't made any invoices

sql:



```
SELECT *
FROM Customer
WHERE CustomerId NOT IN (SELECT DISTINCT CustomerId FROM Invoice);  4ms
```

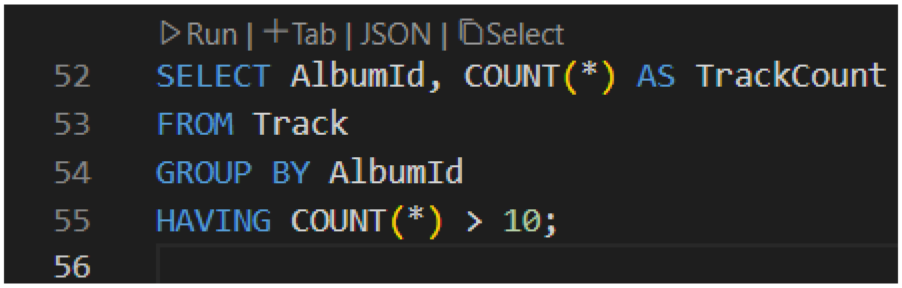
output:



CustomerId int	FirstName varchar	LastName varchar	Company varchar	Address varchar	City varchar	State varchar	Country varchar	PostalCode varchar	Phone varchar	Fax varchar

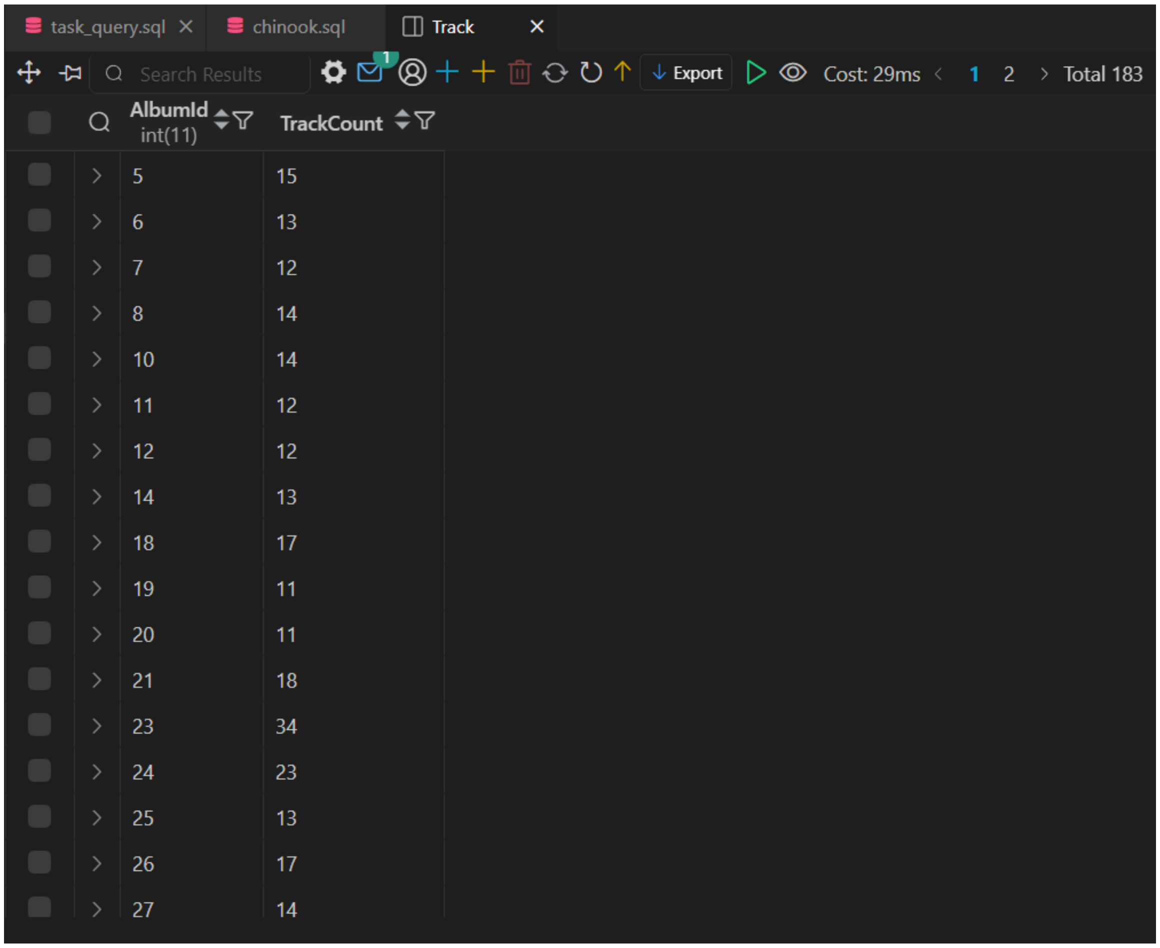
2. 20. Find albums that contain more than 10 tracks

sql:



```
SELECT AlbumId, COUNT(*) AS TrackCount
FROM Track
GROUP BY AlbumId
HAVING COUNT(*) > 10;
```

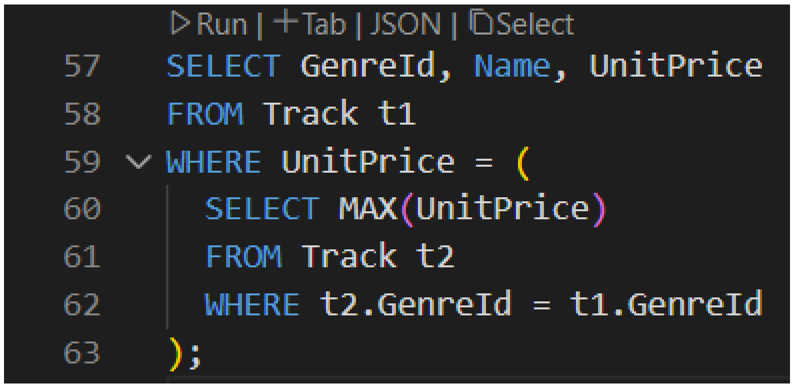
output:



AlbumId int(11)	TrackCount
5	15
6	13
7	12
8	14
10	14
11	12
12	12
14	13
18	17
19	11
20	11
21	18
23	34
24	23
25	13
26	17
27	14

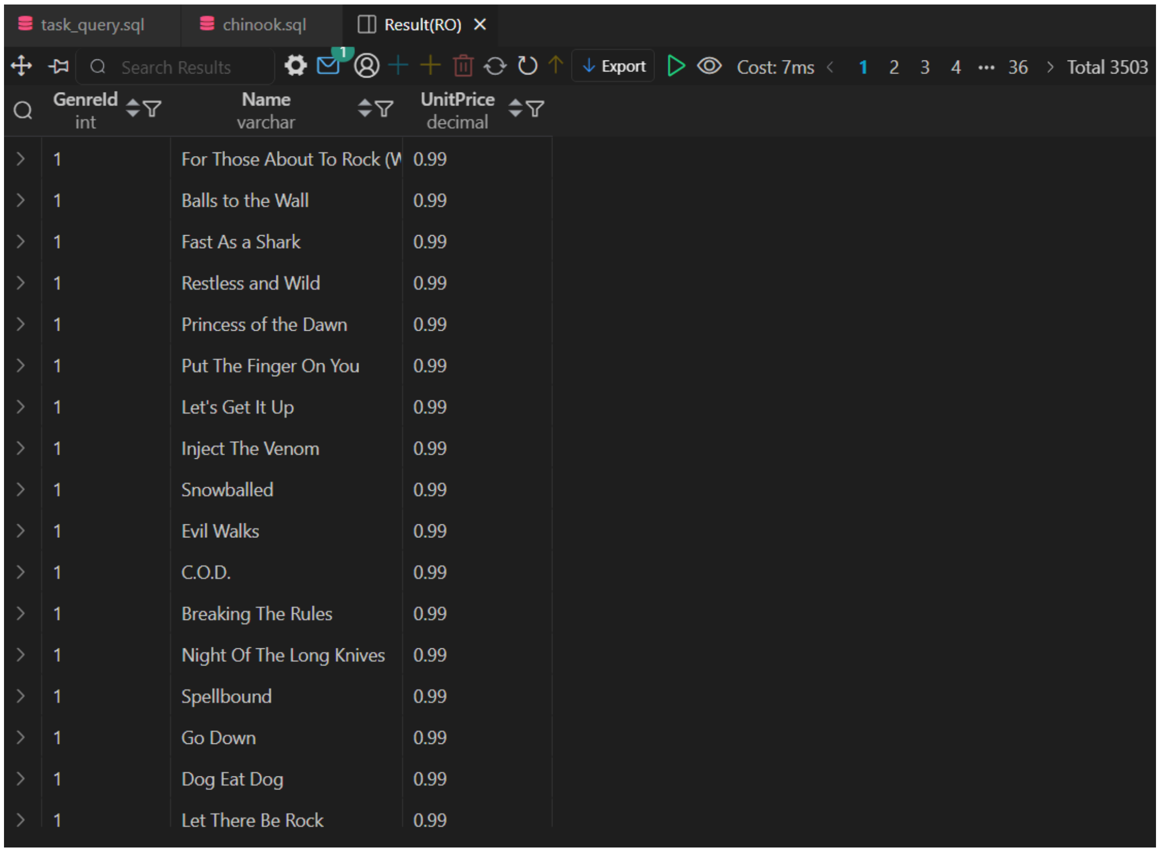
3. 21. Get the most expensive track in each genre

sql:



```
▷Run | +Tab | JSON | ⧉Select
SELECT GenreId, Name, UnitPrice
FROM Track t1
WHERE UnitPrice = (
    SELECT MAX(UnitPrice)
    FROM Track t2
    WHERE t2.GenreId = t1.GenreId
);
```

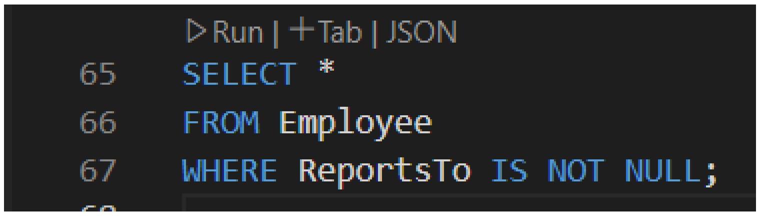
output:



```
GenreId  Name                        UnitPrice
int      varchar                     decimal
1        For Those About To Rock (W  0.99
1        Balls to the Wall           0.99
1        Fast As a Shark             0.99
1        Restless and Wild           0.99
1        Princess of the Dawn        0.99
1        Put The Finger On You       0.99
1        Let's Get It Up             0.99
1        Inject The Venom            0.99
1        Snowballed                  0.99
1        Evil Walks                  0.99
1        C.O.D.                      0.99
1        Breaking The Rules          0.99
1        Night Of The Long Knives    0.99
1        Spellbound                  0.99
1        Go Down                     0.99
1        Dog Eat Dog                 0.99
1        Let There Be Rock           0.99
```

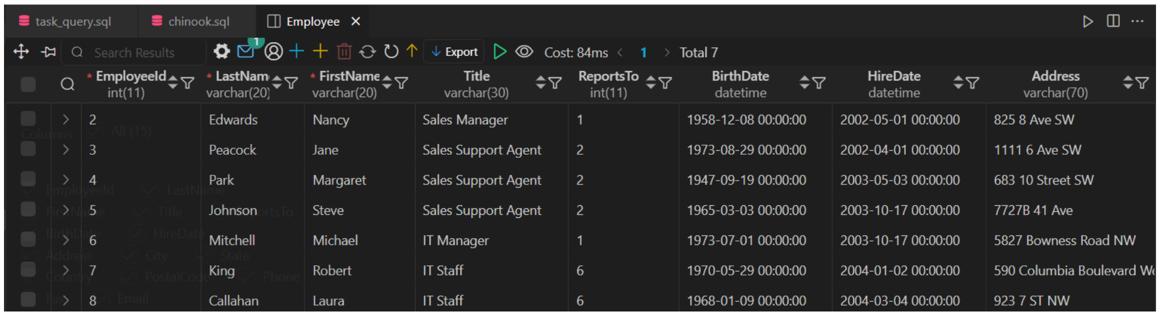
4. 22. List employees who report to someone (non-null ReportsTo)

sql:



```
▷ Run | ＋Tab | JSON
SELECT *
FROM Employee
WHERE ReportsTo IS NOT NULL;
```

output:

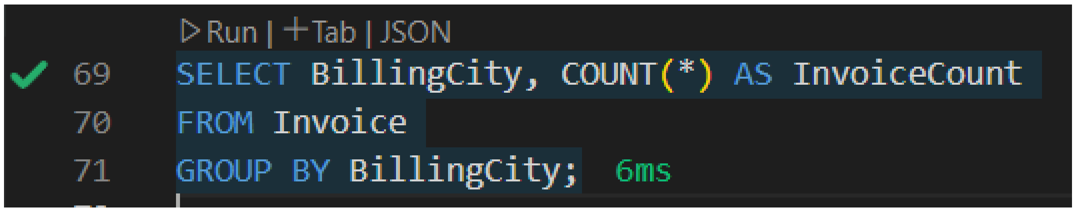


```
task_query.sql   chinook.sql   Employee ×
Search Results   Export   Cost: 84ms < 1 > Total 7
```

EmployeeId int(11)	LastName varchar(20)	FirstName varchar(20)	Title varchar(30)	ReportsTo int(11)	BirthDate datetime	HireDate datetime	Address varchar(70)
2	Edwards	Nancy	Sales Manager	1	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW
3	Peacock	Jane	Sales Support Agent	2	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW
4	Park	Margaret	Sales Support Agent	2	1947-09-19 00:00:00	2003-05-03 00:00:00	683 10 Street SW
5	Johnson	Steve	Sales Support Agent	2	1965-03-03 00:00:00	2003-10-17 00:00:00	7727B 41 Ave
6	Mitchell	Michael	IT Manager	1	1973-07-01 00:00:00	2003-10-17 00:00:00	5827 Bowness Road NW
7	King	Robert	IT Staff	6	1970-05-29 00:00:00	2004-01-02 00:00:00	590 Columbia Boulevard We
8	Callahan	Laura	IT Staff	6	1968-01-09 00:00:00	2004-03-04 00:00:00	923 7 ST NW

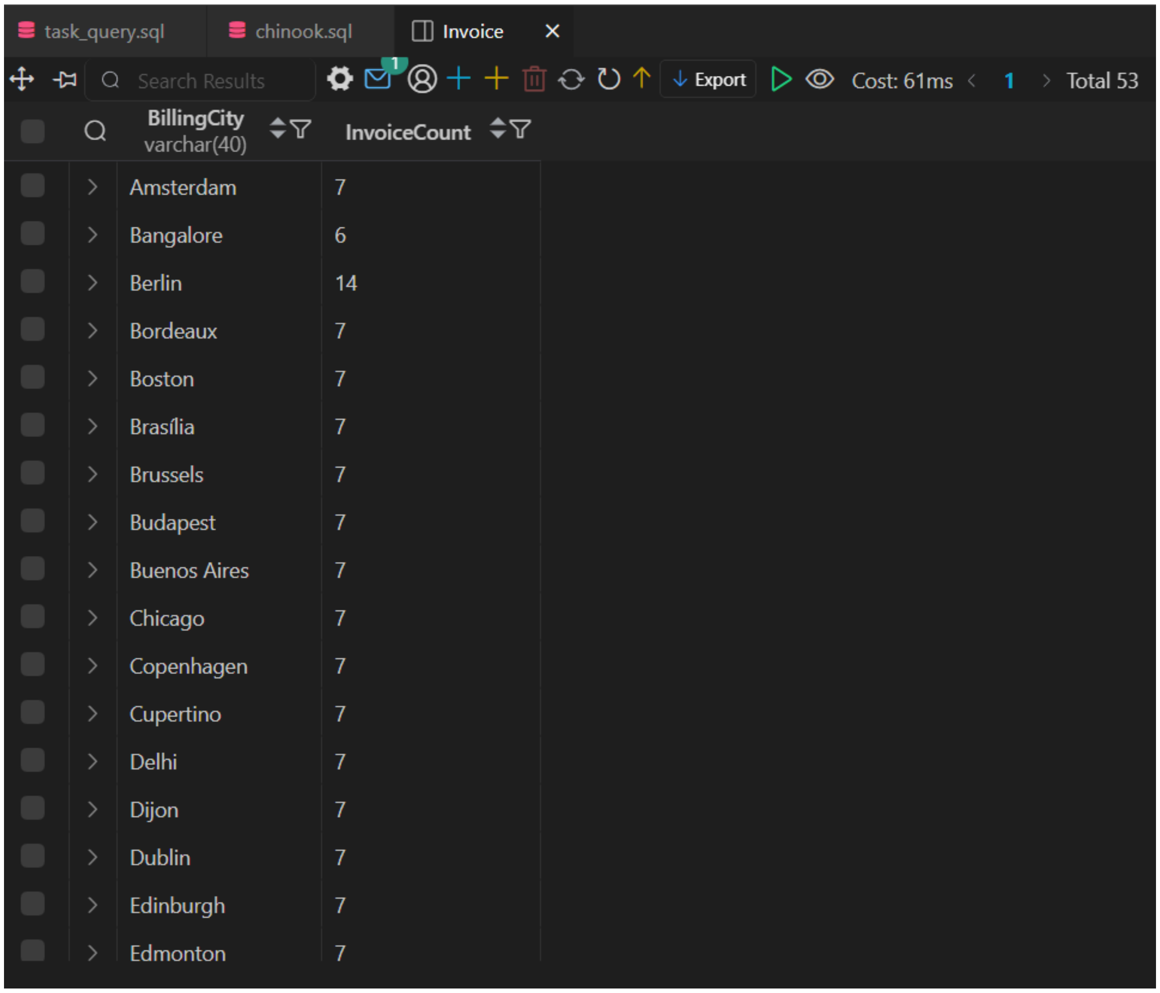
5. 23. Count how many invoices were created per billing city

sql:



```
▷ Run | ＋Tab | JSON
SELECT BillingCity, COUNT(*) AS InvoiceCount
FROM Invoice
GROUP BY BillingCity;   6ms
```

output:



task_query.sql | chinook.sql | Invoice

Cost: 61ms | 1 | Total 53

BillingCity varchar(40)	InvoiceCount
Amsterdam	7
Bangalore	6
Berlin	14
Bordeaux	7
Boston	7
Brasília	7
Brussels	7
Budapest	7
Buenos Aires	7
Chicago	7
Copenhagen	7
Cupertino	7
Delhi	7
Dijon	7
Dublin	7
Edinburgh	7
Edmonton	7