https://www.w3schools.com/SQL/

SQL Primary Key

The PRIMARY KEY uniquely identifies each record in a table.

For e.g : Store ID in Stores Table, Order ID in Sales Table

SQL Foreign Key

A FOREIGN KEY is a field/column (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

For e.g Suppose we have two tables Stores Table and Customers Table. Stores Table contains details of all Reliance stores across India and Customers table having details of the all customers. Store ID in present in both the tables but Store ID will be unique in Stores Table and it can be present against multiple customers in customers table as many people are visiting one store.

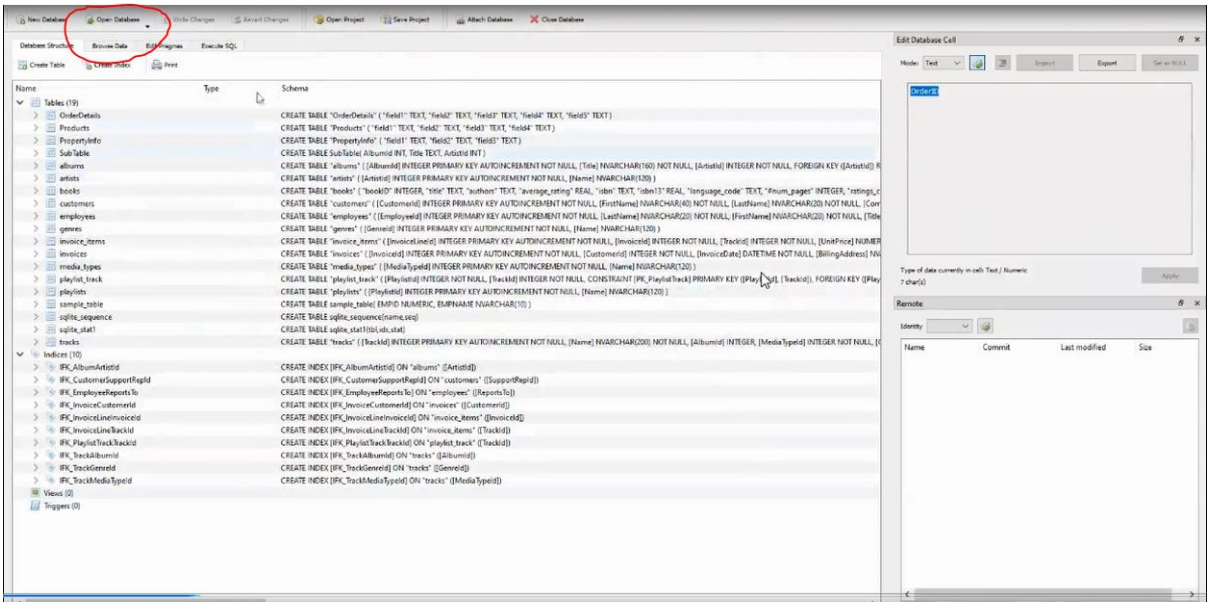
So, Store ID will be a Primary key in Stores Table and Foreign Key in Customers Table and two tables can be joined with Store ID.

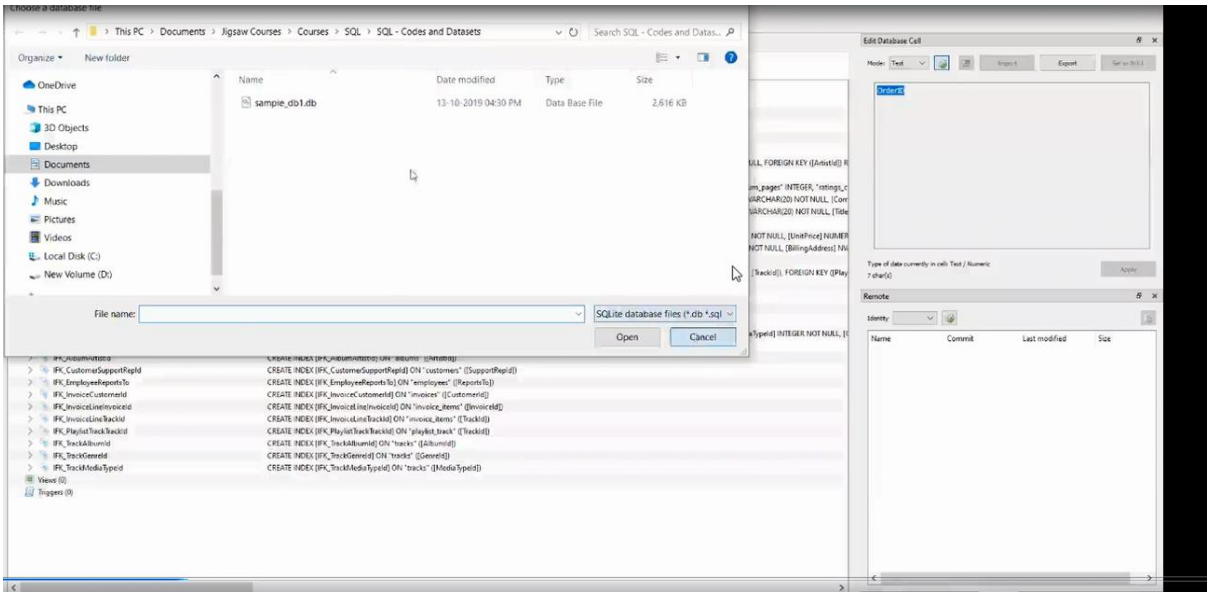
Primary key vs Unique Key

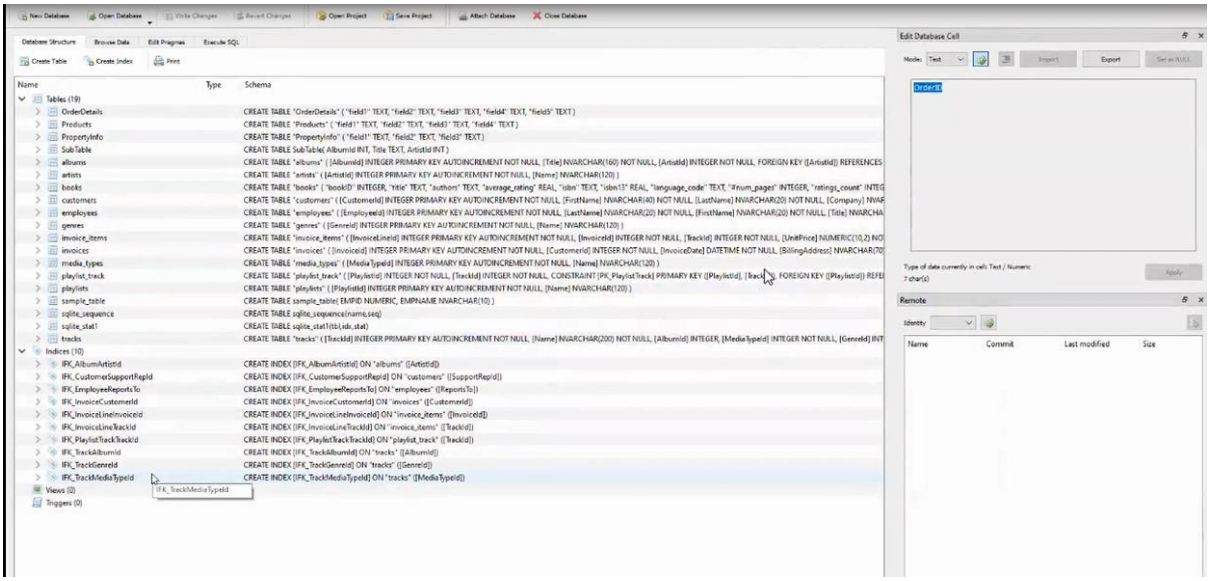
In the case of a primary key, we cannot save NULL values. In the case of a unique key, we can save a null value, however, only one NULL value is supported.

https://byjus.com/gate/difference-between-primary-key-and-unique-key/

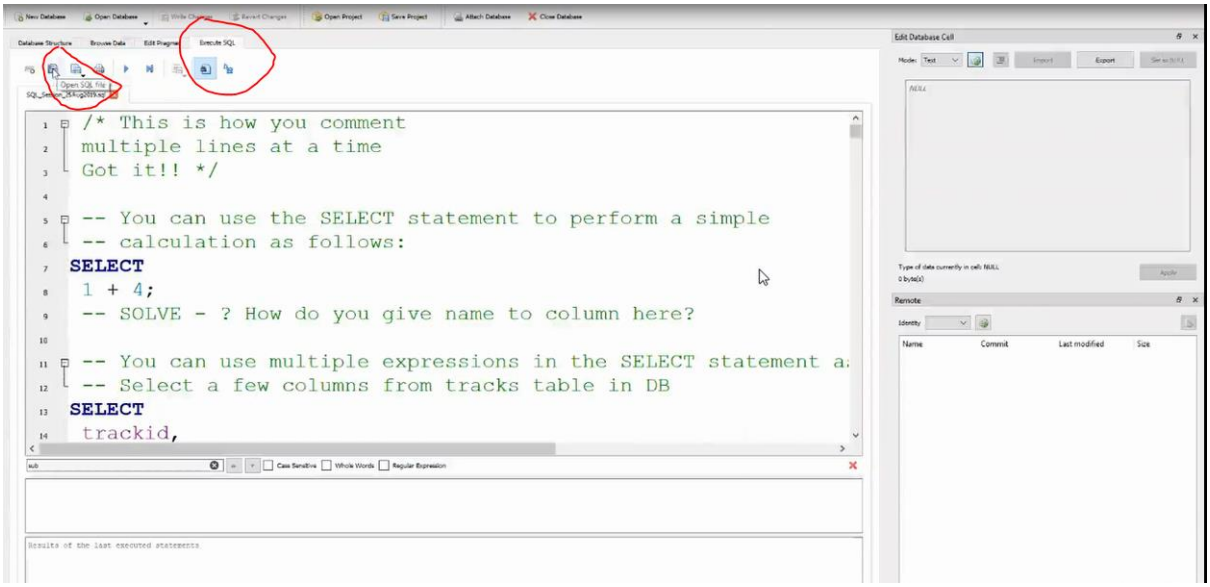
Open Database -> go to the db location -> open file (.db file)







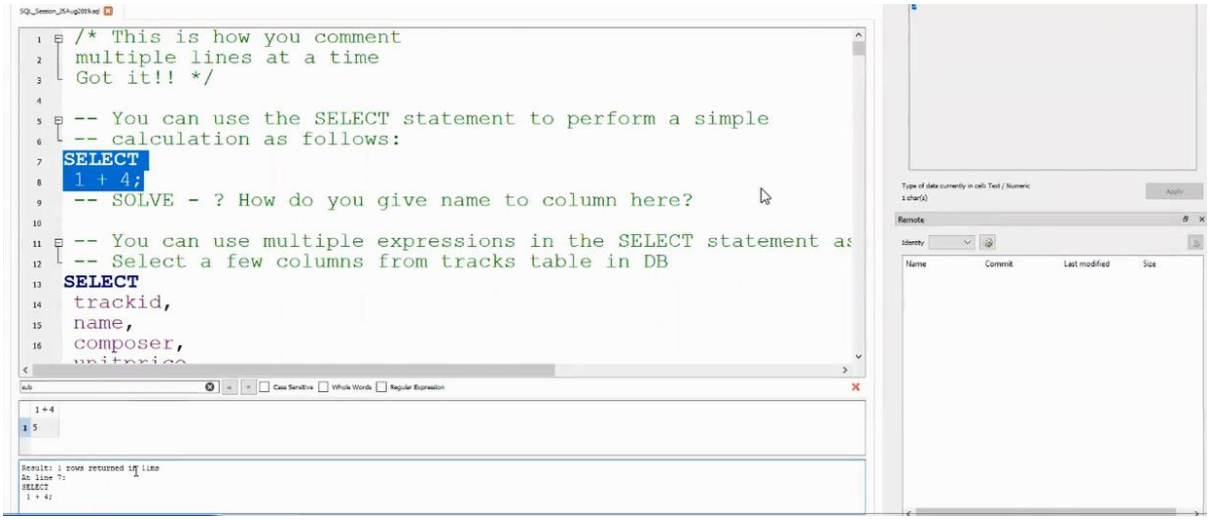
Execute SQL -> open SQL file -> go to location -> open file (.sql file)



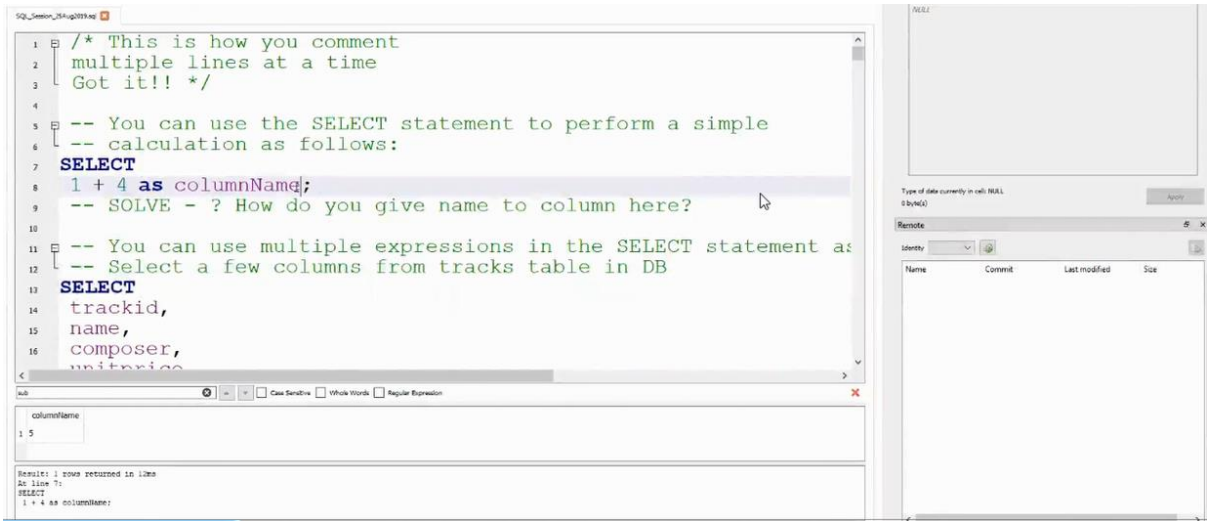
Database structure tab -> all tables are listed over here

Execute SQL -> code is written over here

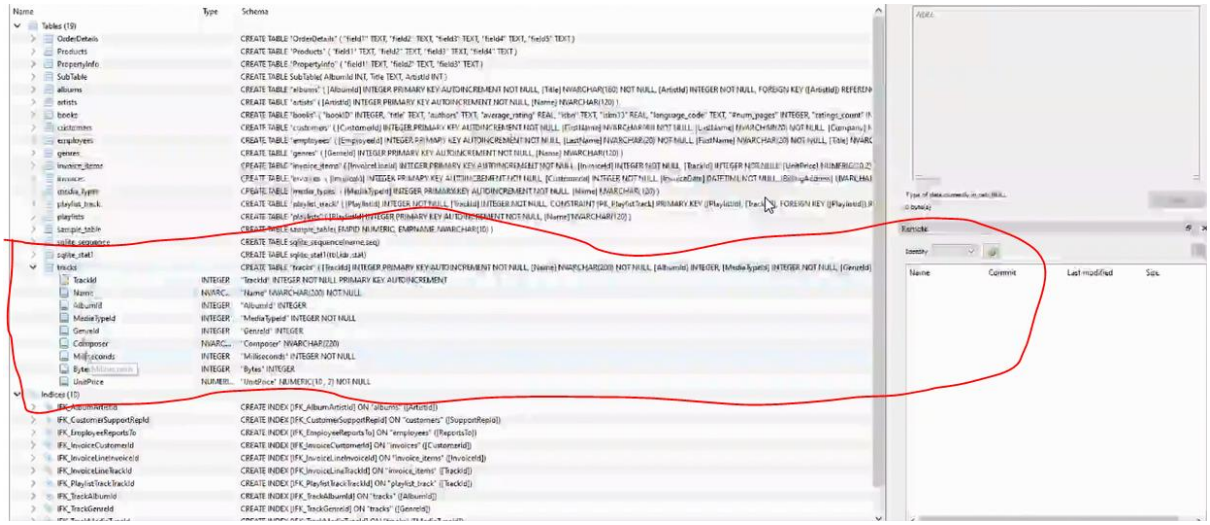
Simple math calculation



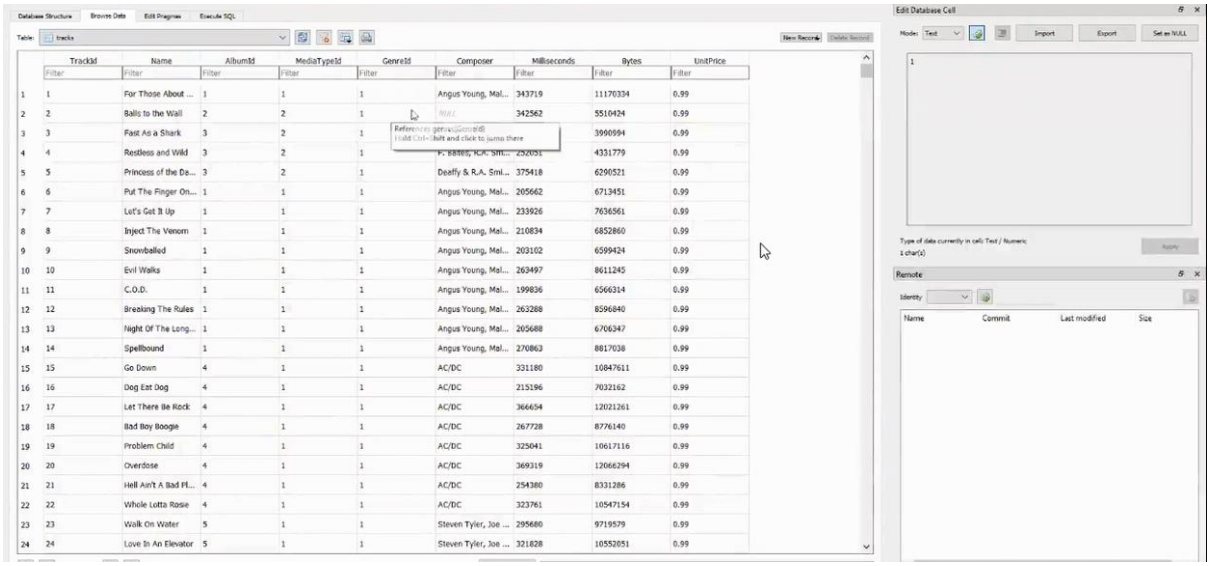
Columnname is available with this piece of code



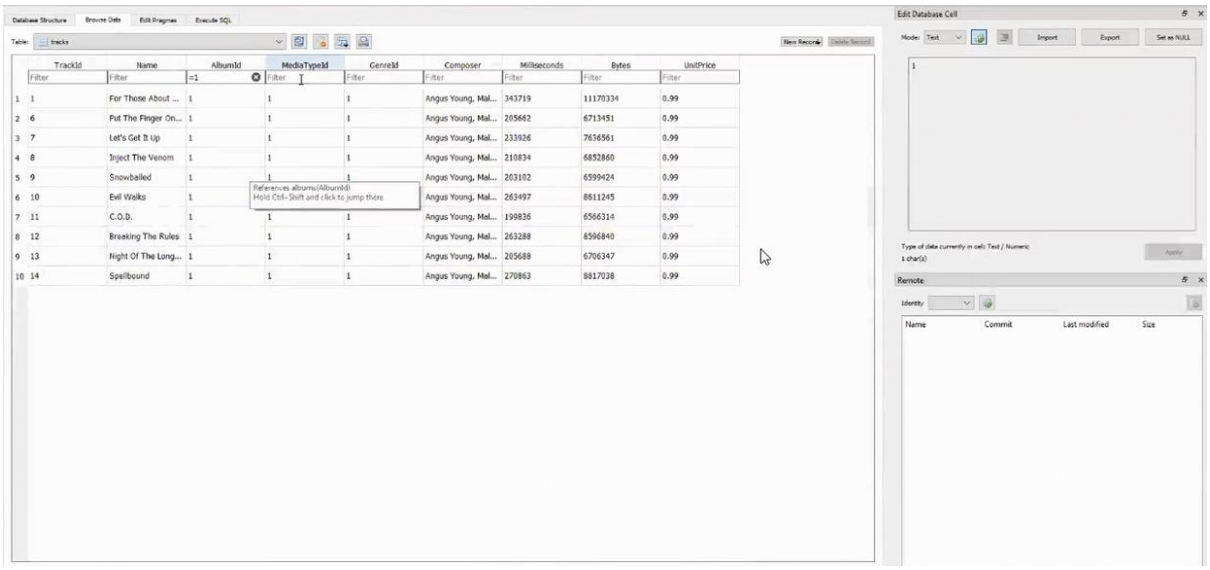
Trackid table

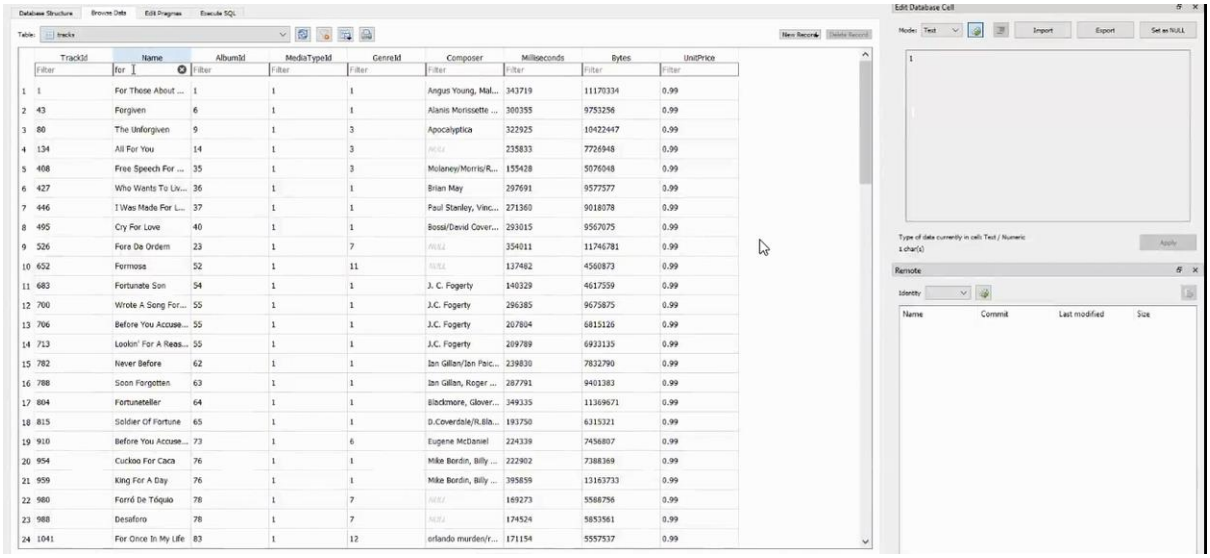


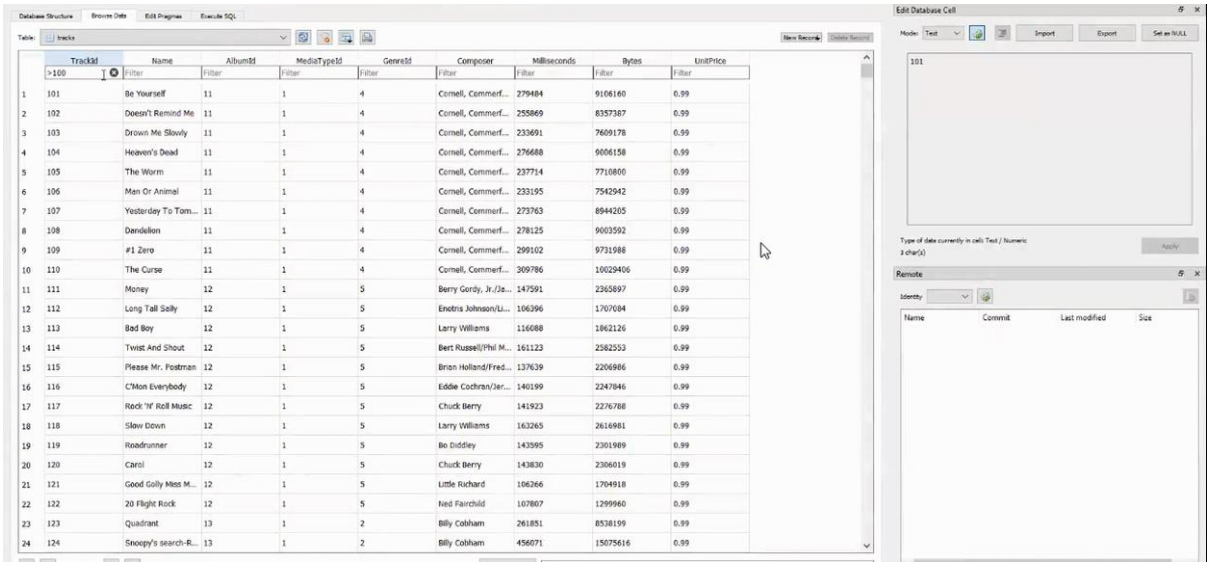
Browse data tab -> to have a look at the particular table



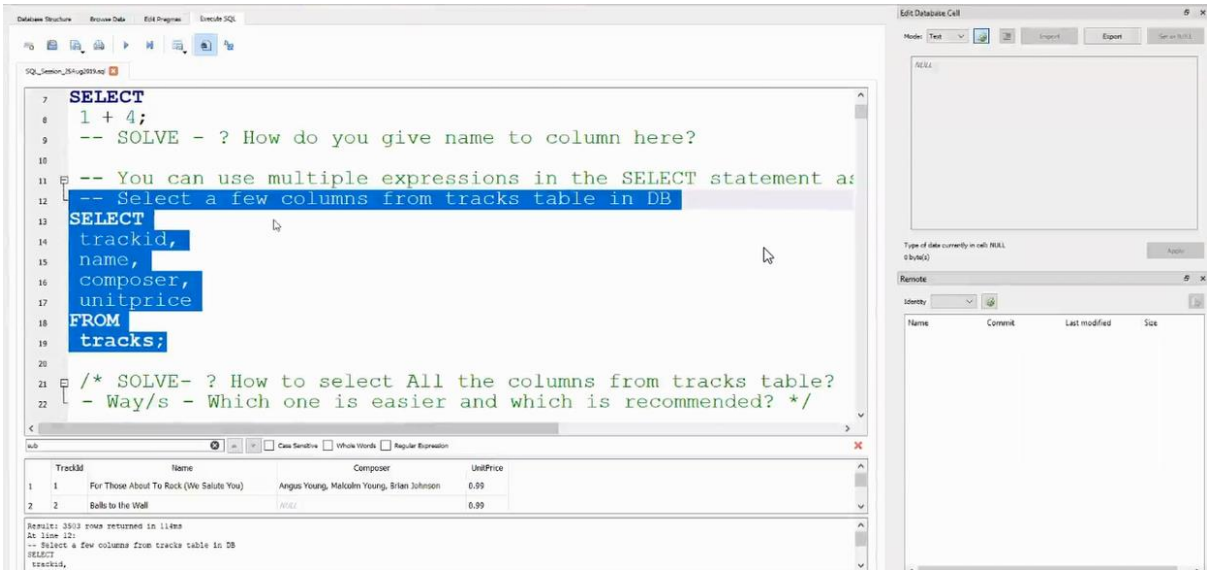
Data can be filtered over here



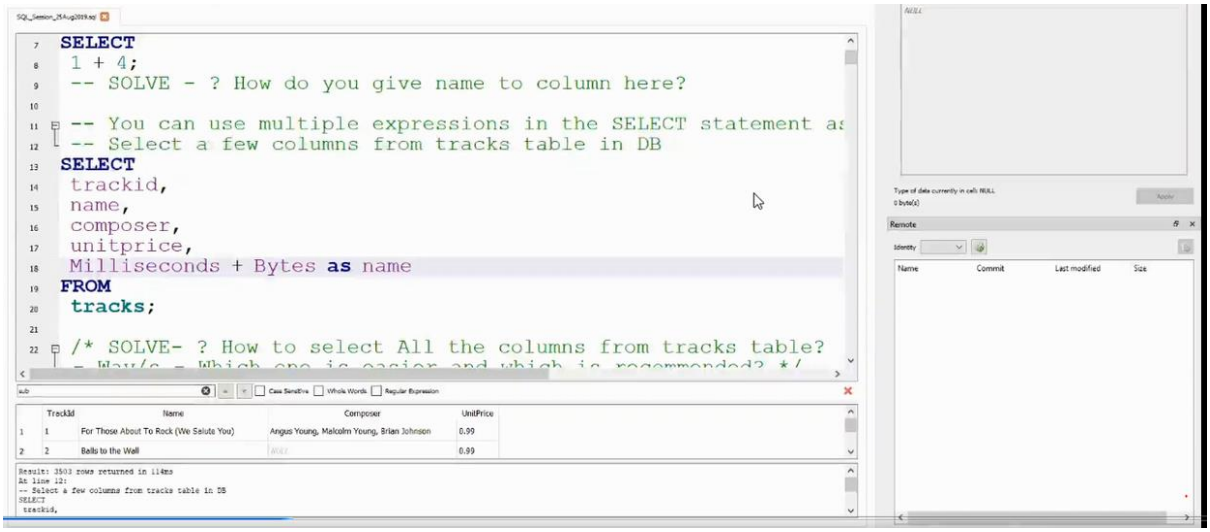




Selecting few columns from the table (select * to select all the columns)

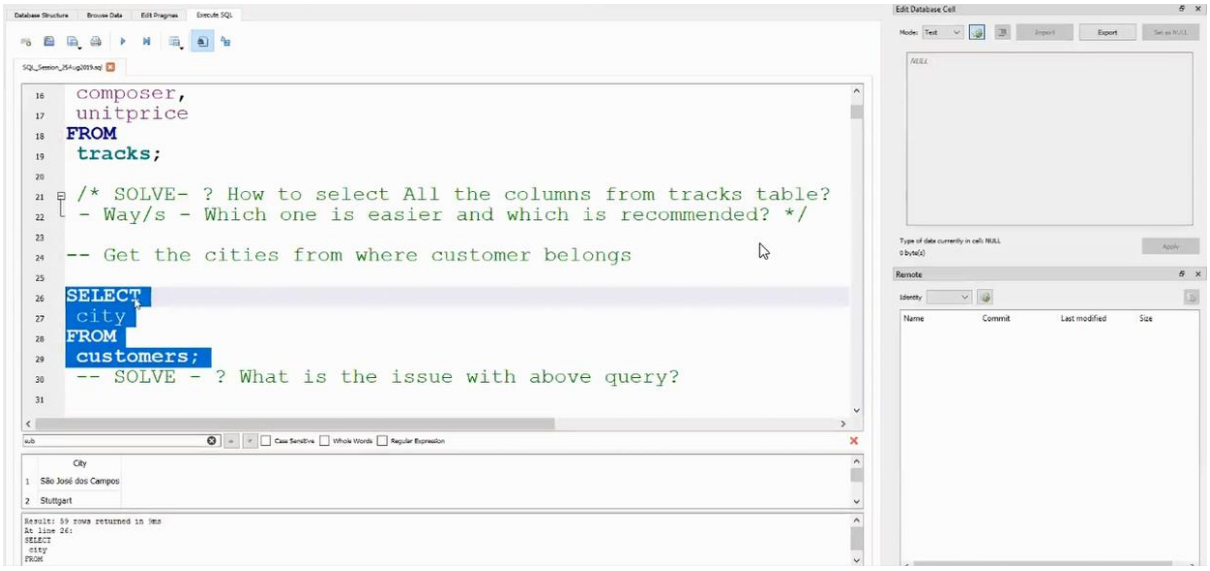


Addition of two columns

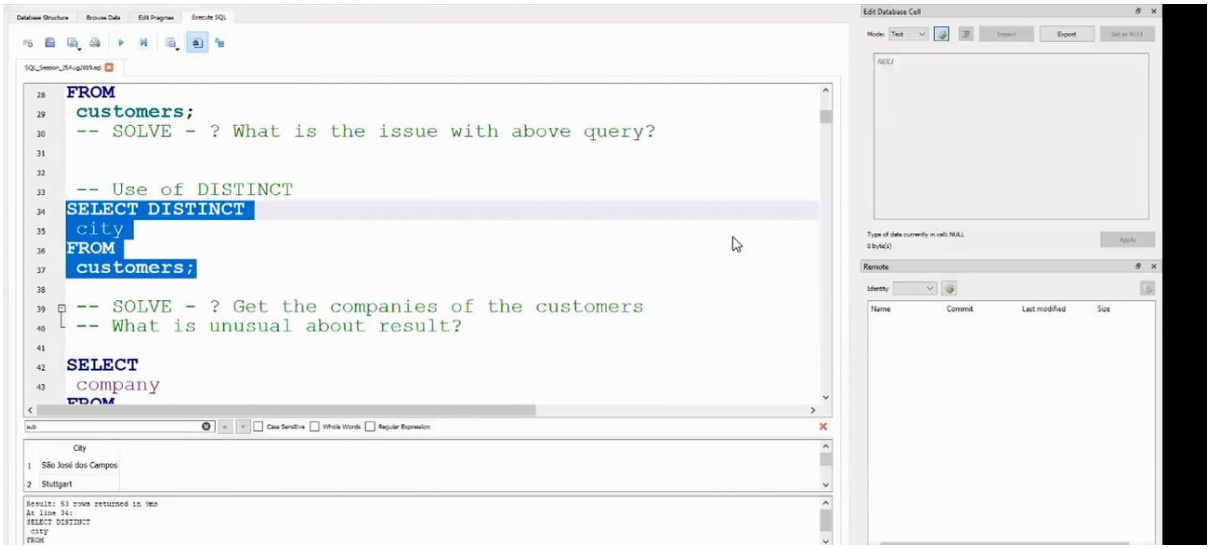


This code will select the all the rows and there will be duplicate city names present in the result

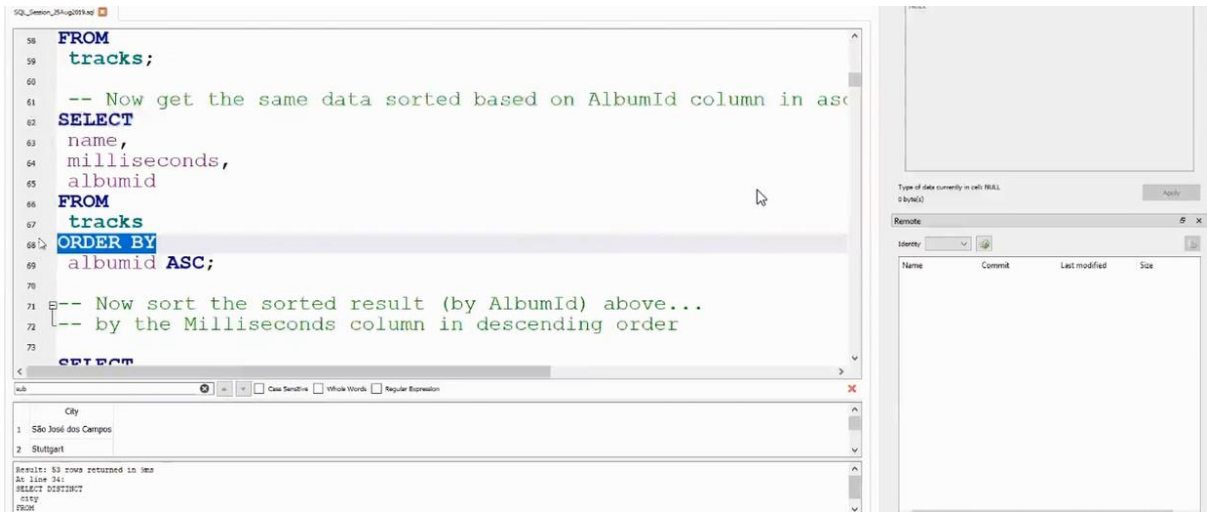
(59 rows)



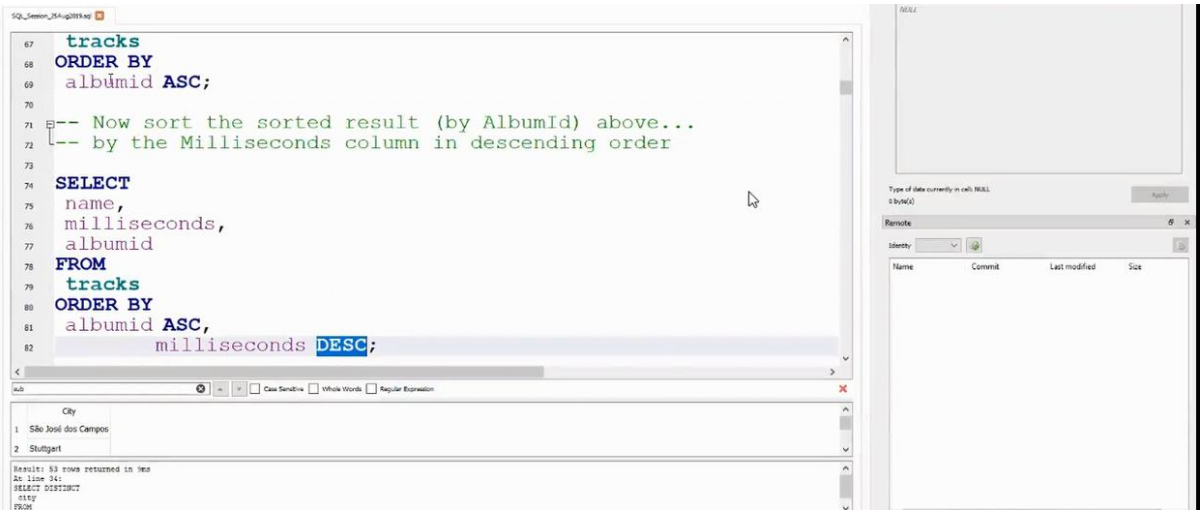
Distinct will select only unique citys (53 rows)



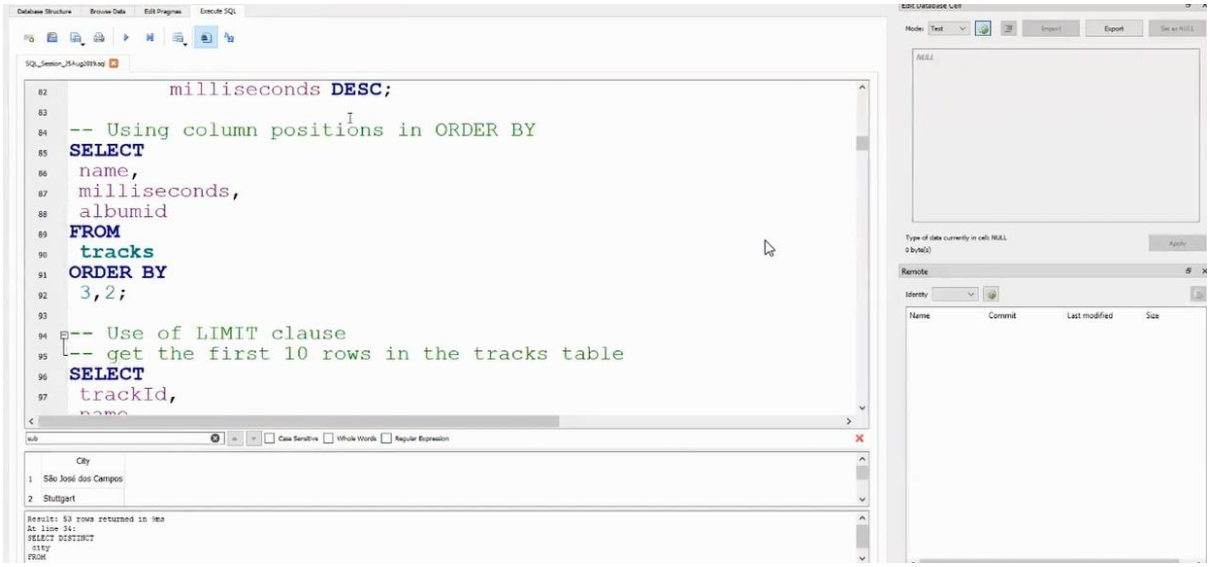
Orderby will sort the data



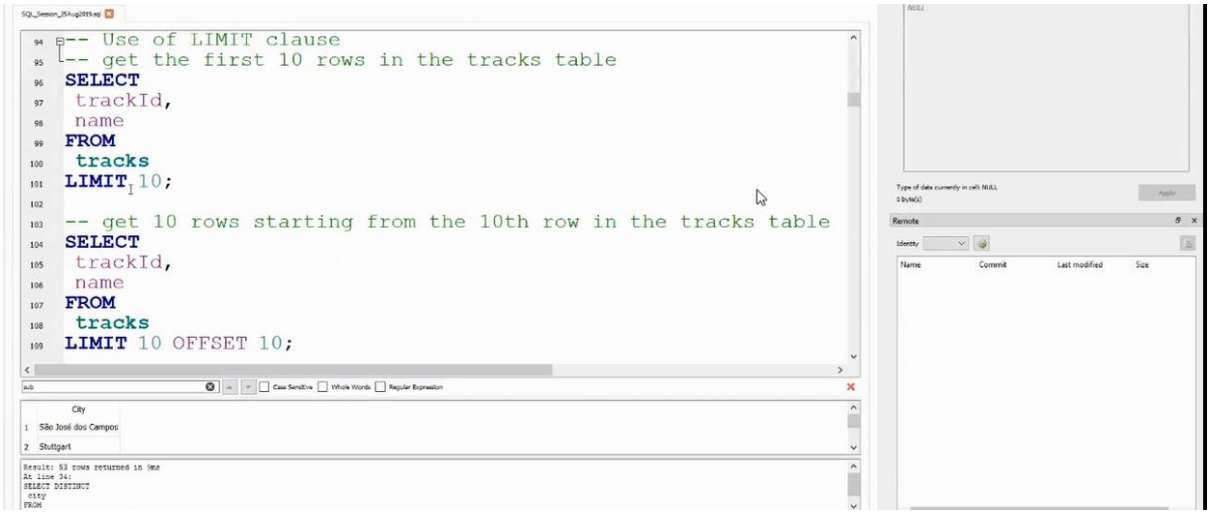
ASC : Ascending ; DESC : Descending



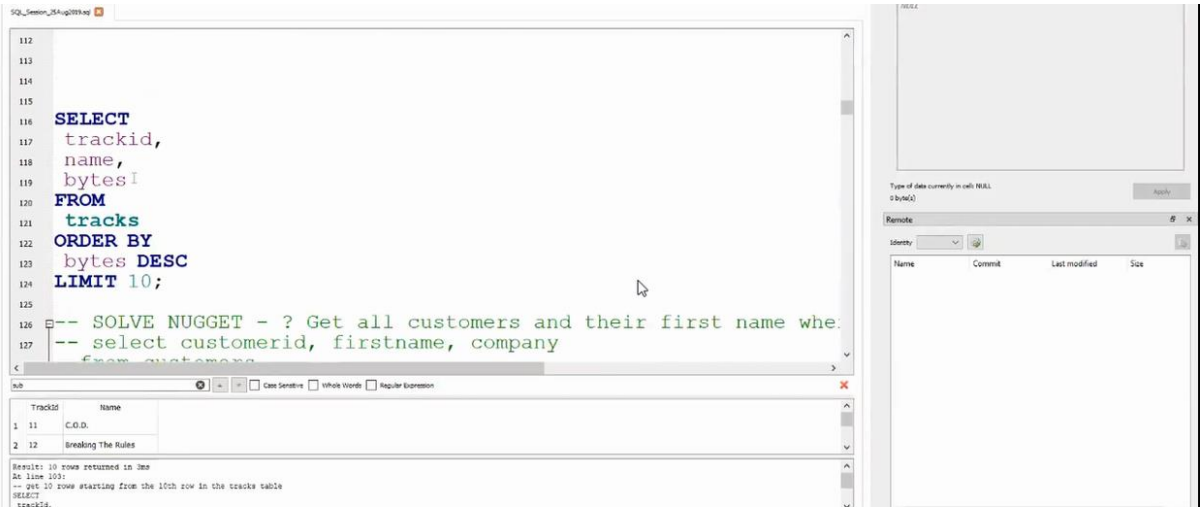
Using column positions to sort



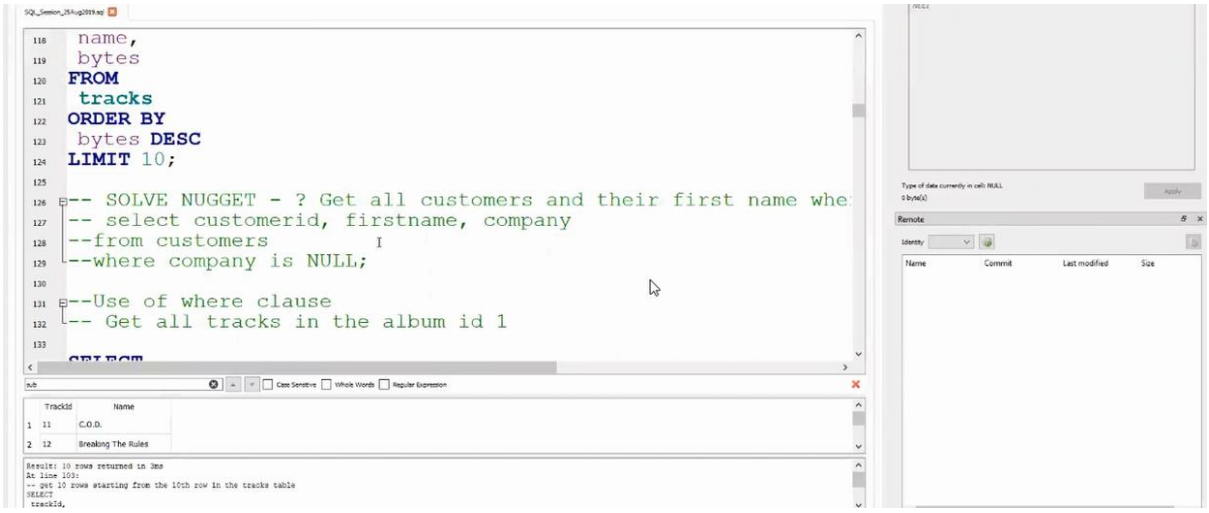
Limit will give only given no of rows of data, offset will skip given no of rows



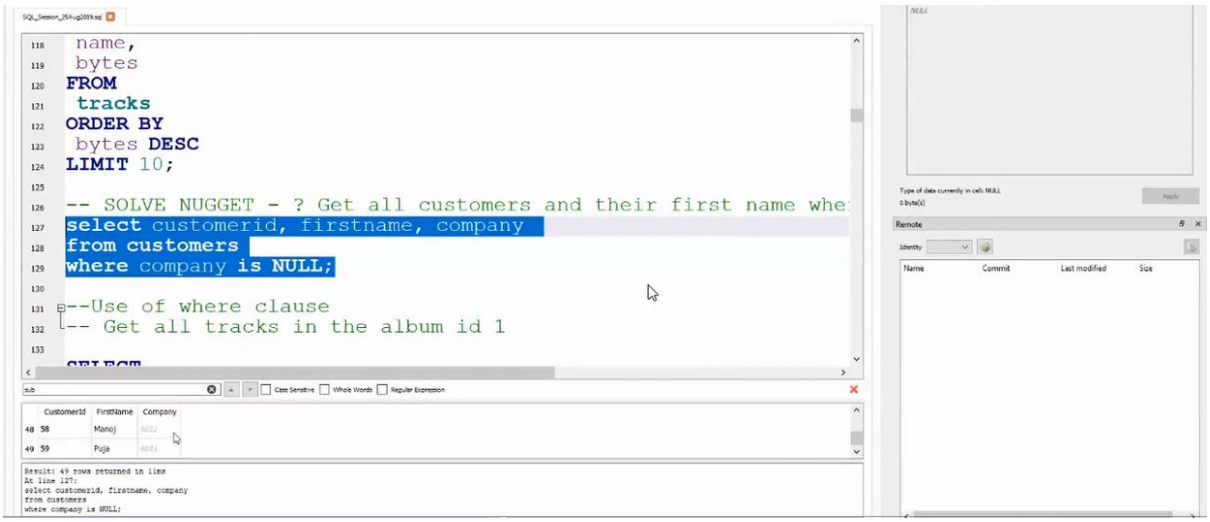
sort and limit on single piece of code

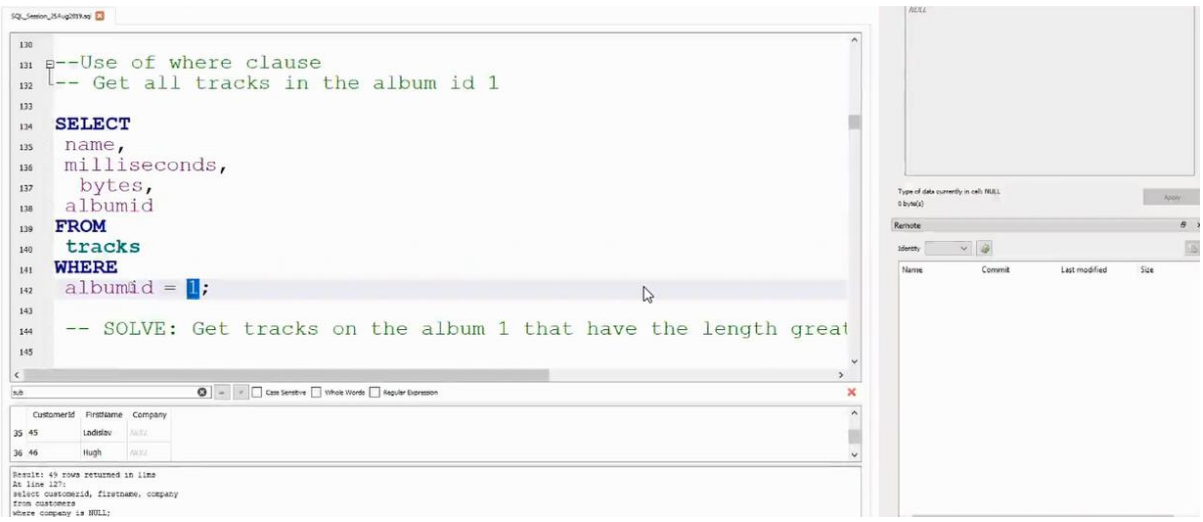


Select data where company is null (company name is absent)

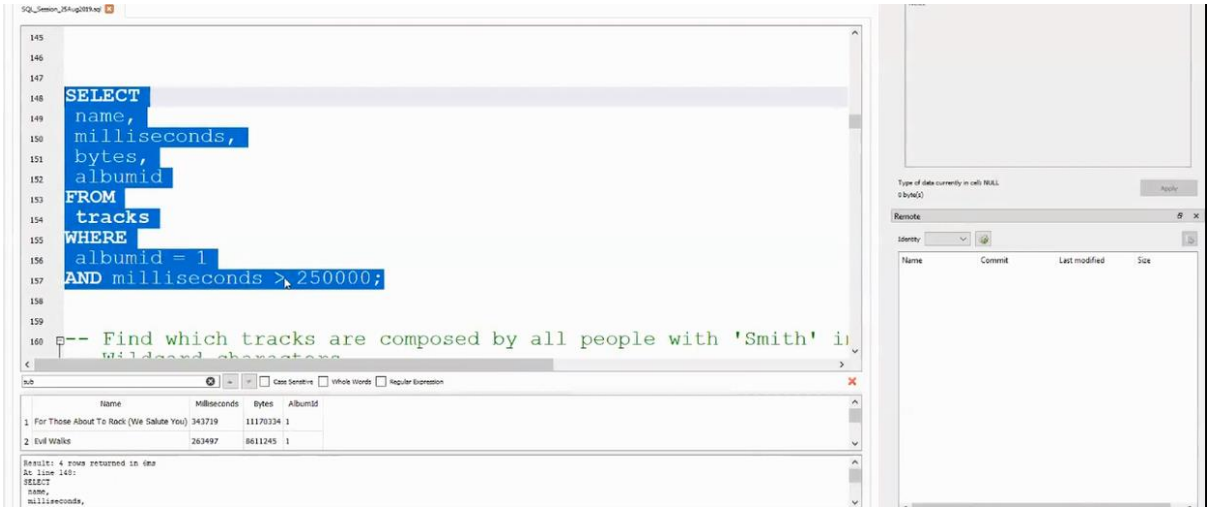


Where statement is used in this piece of code

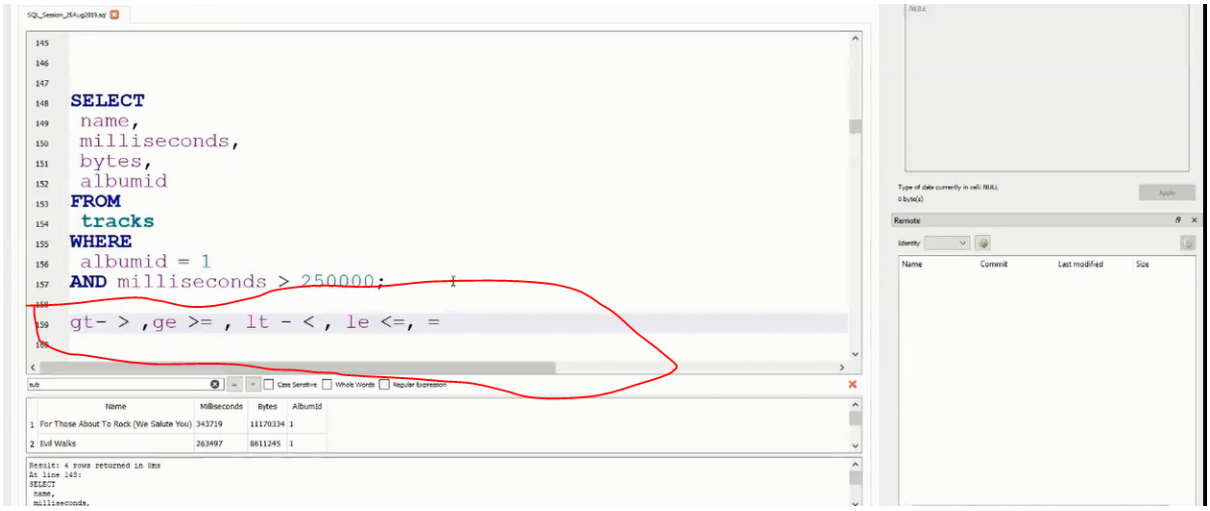




And will apply to conditions inside where statement



In some of the query languages words can be used instead of signs

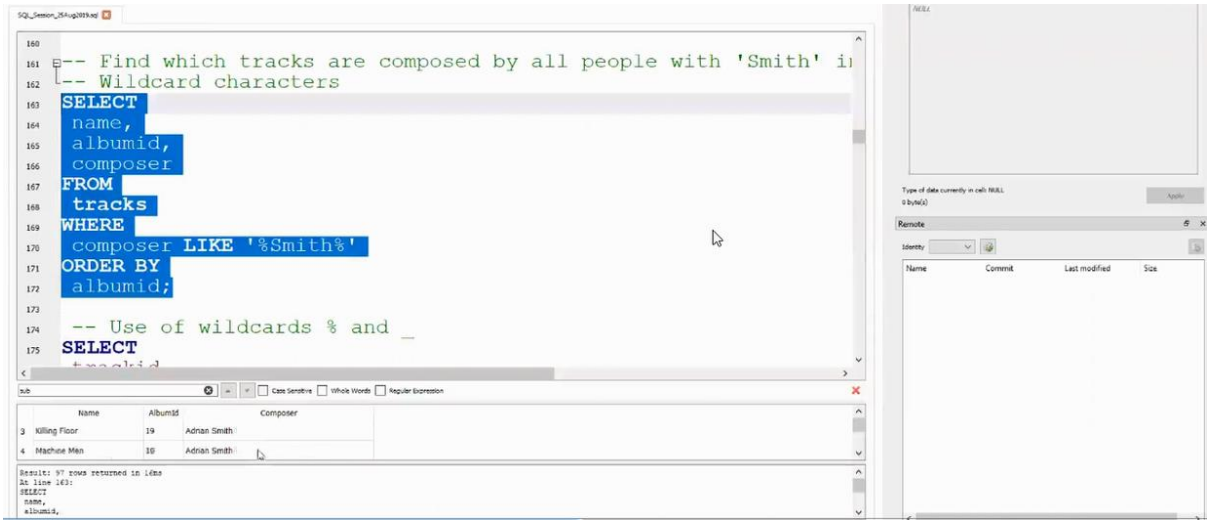


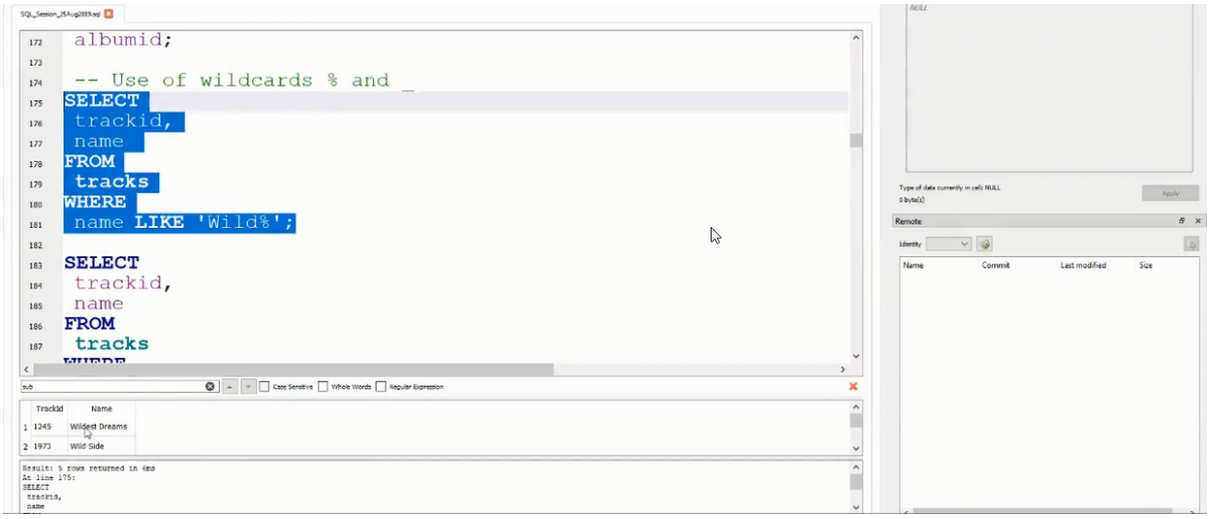
To select data rows which contains part of the string (% signs can be used)

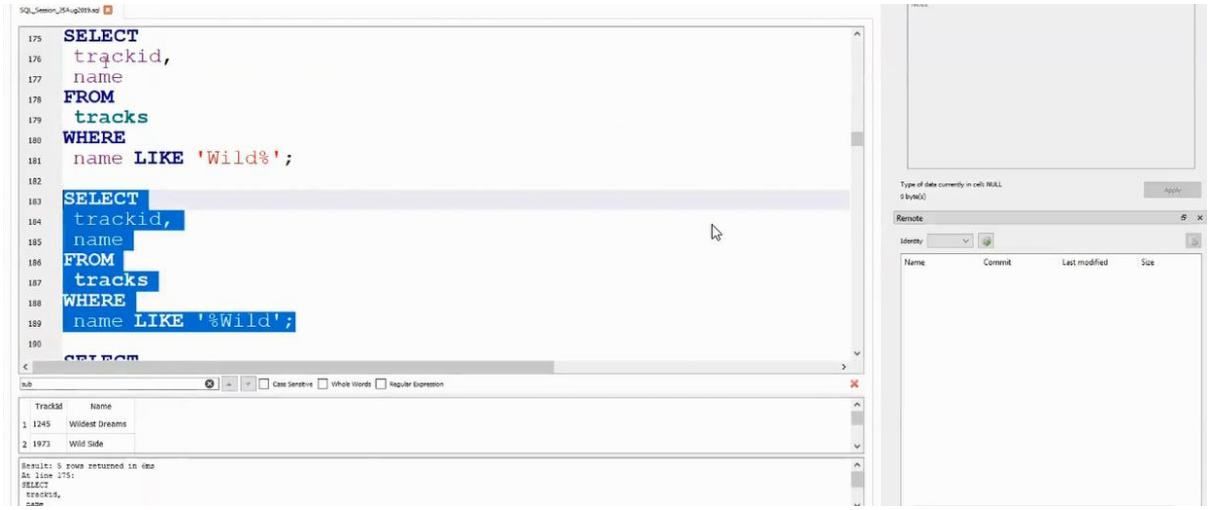
% before word – another words/letters are present before this word

% after word – another words/letters are present after this word

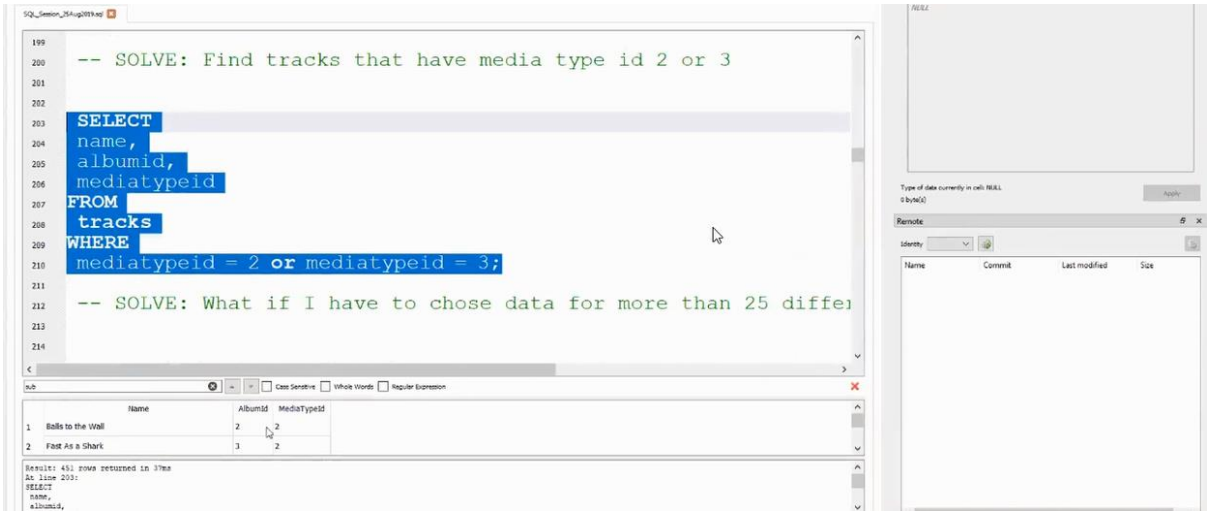
% on both side – another words/letters are present before & after this word



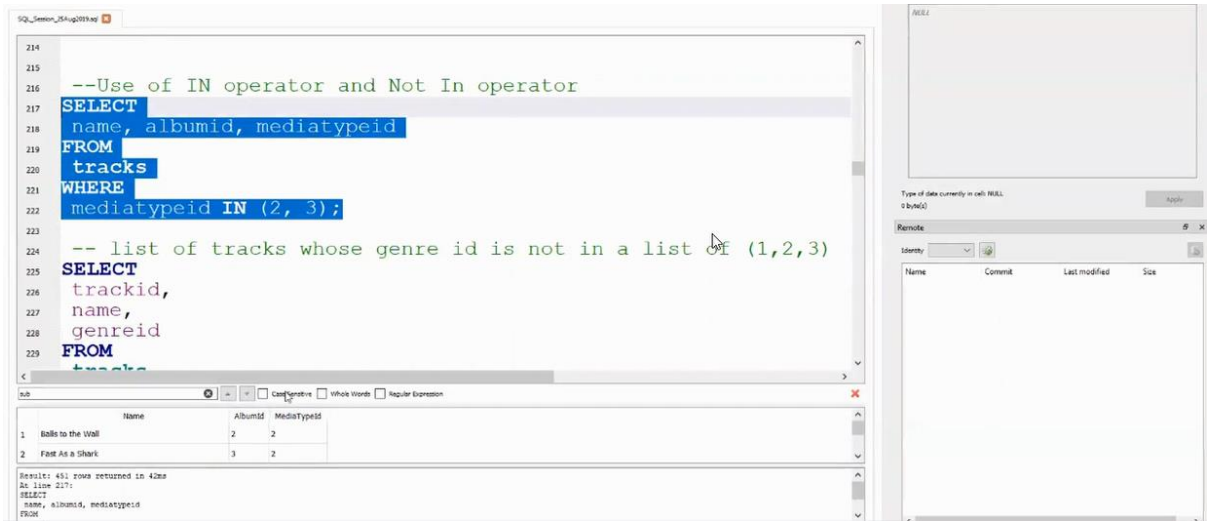




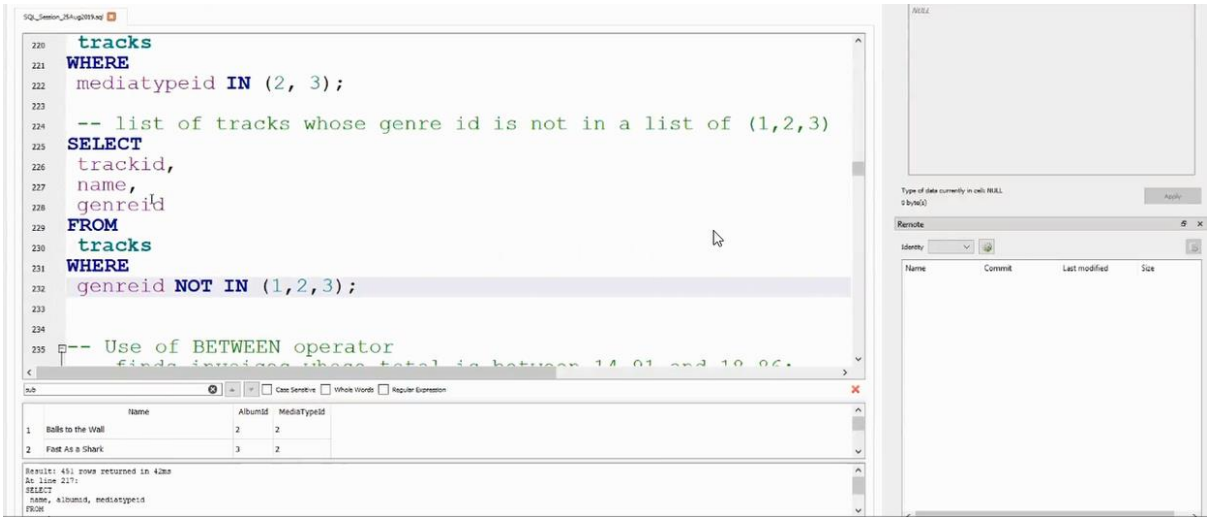
Or condition inside where



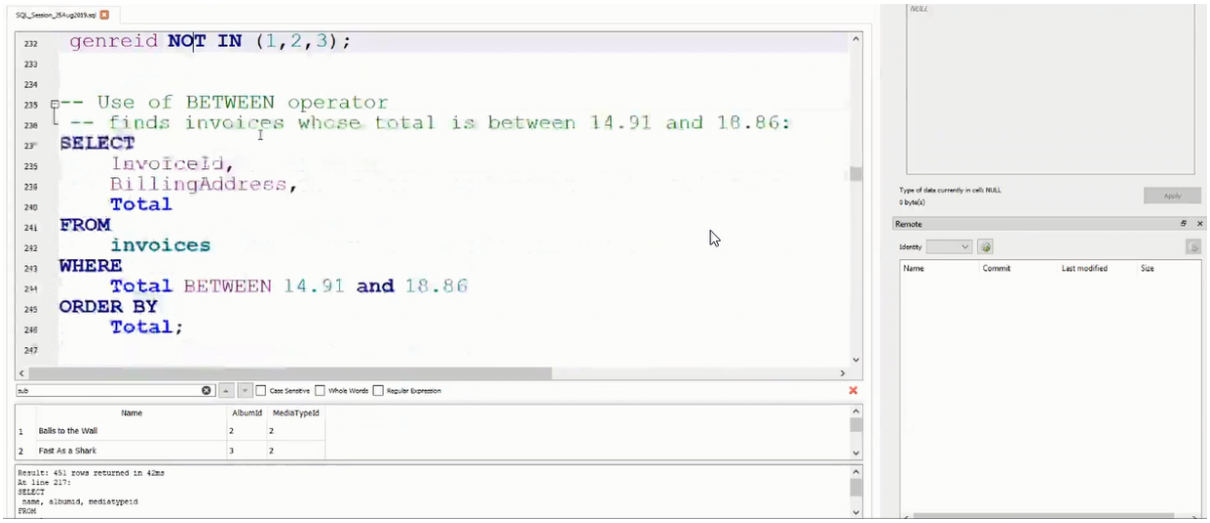
In condition – alternate for or condition (if applied on single column)



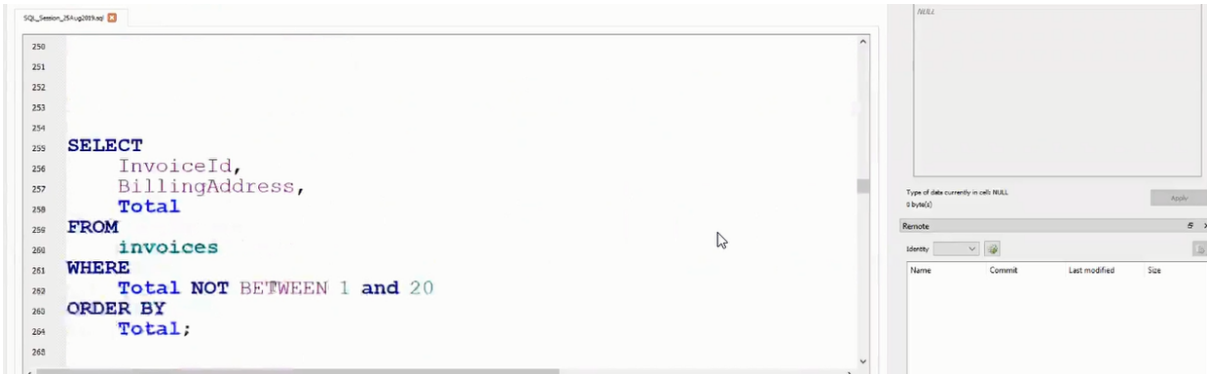
Notin condition – opposite of in



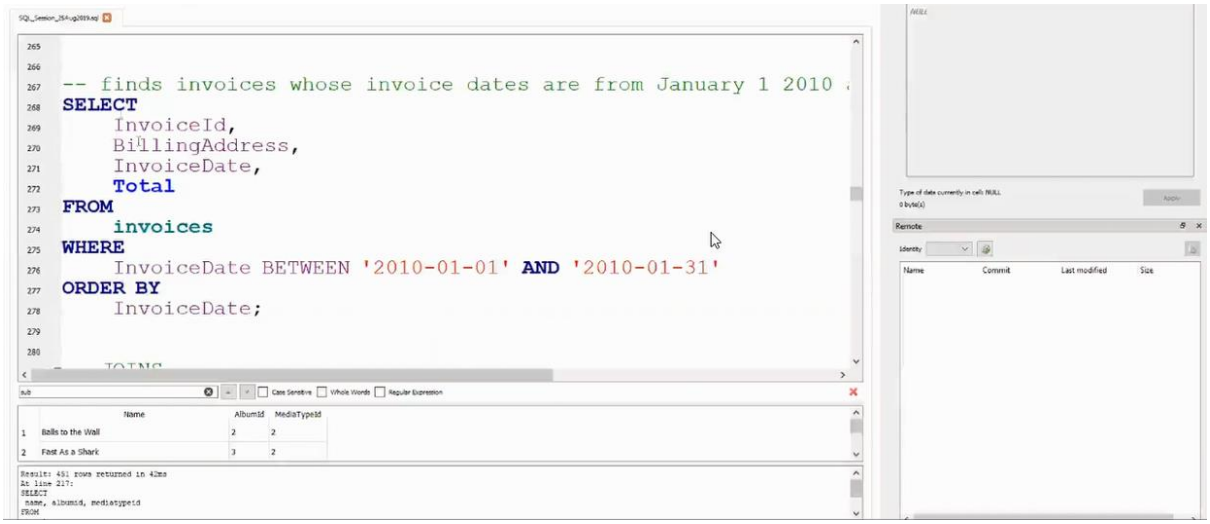
Between condition



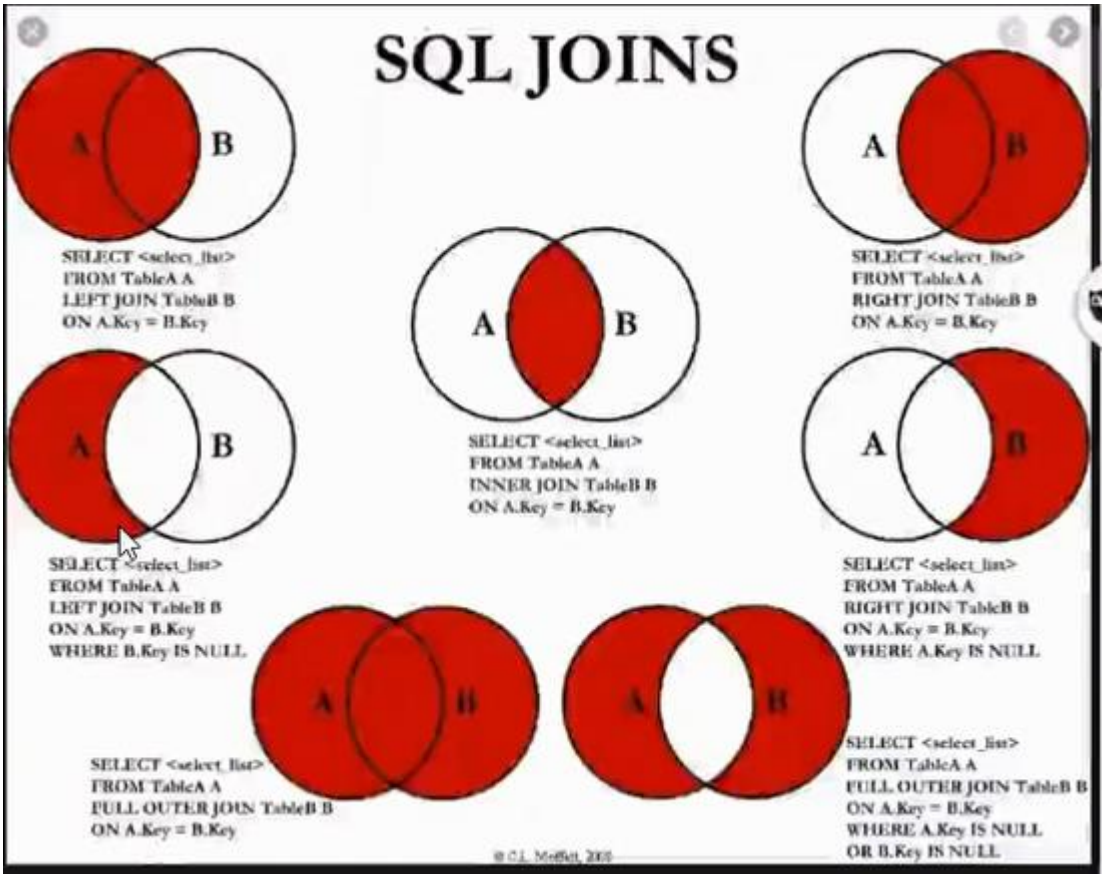
notbetween condition



Between condition on dates



SQL Joins



Cust_id	City
1	Mumbai
2	Mumbai
3	Bangalore
4	Mysore
5	Hyderabad
6	Pune

Cust_id	Sales
1	100
2	220.5
4	50
5	400
7	30
8	480.7

Inner Join

Cust_id	Sales	City
1	100	Mumbai
2	220.5	Mumbai
4	50	Mysore
5	400	Hyderabad

Left Join

Cust_id	City	Sales
1	Mumbai	100
2	Mumbai	220.5
3	Bangalore	NULL
4	Mysore	50
5	Hyderabad	400
6	Pune	NULL

Right Join

Cust_id	Sales	City
1	100	Mumbai
2	220.5	Mumbai
4	50	Mysore
5	400	Hyderabad
7	30	NULL
8	480.7	NULL

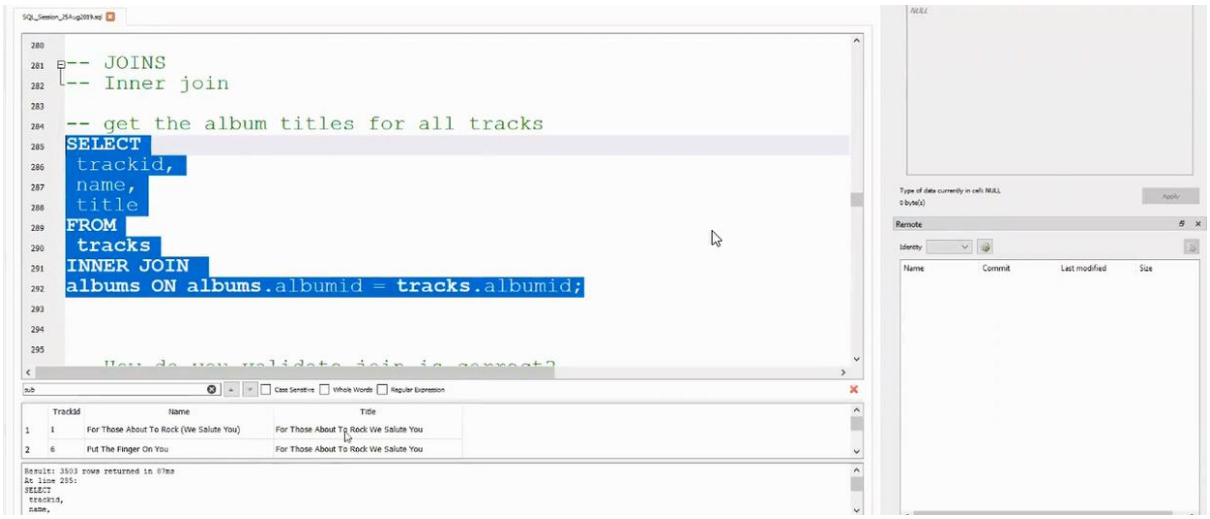
Full Outer Join

Cust_id	Sales	City
1	100	Mumbai
2	220.5	Mumbai
3	50	Bangalore
4	400	Mysore
5	400	Hyderabad
6	0	Pune
7	30	NULL
8	480.7	NULL

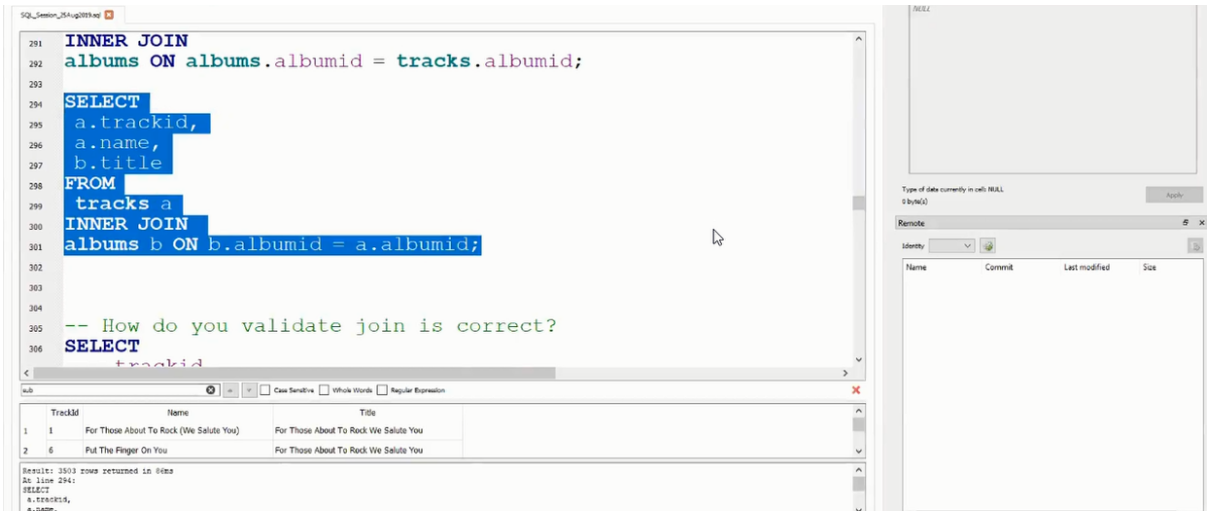
Inner join on tracks and album

Trackid and name from tracks table

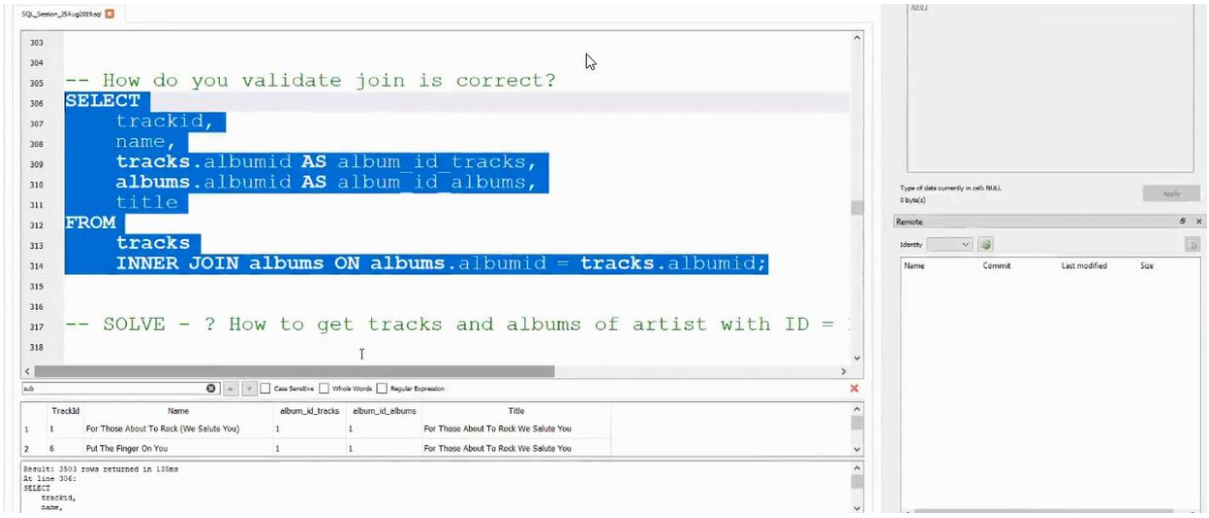
Title from album table



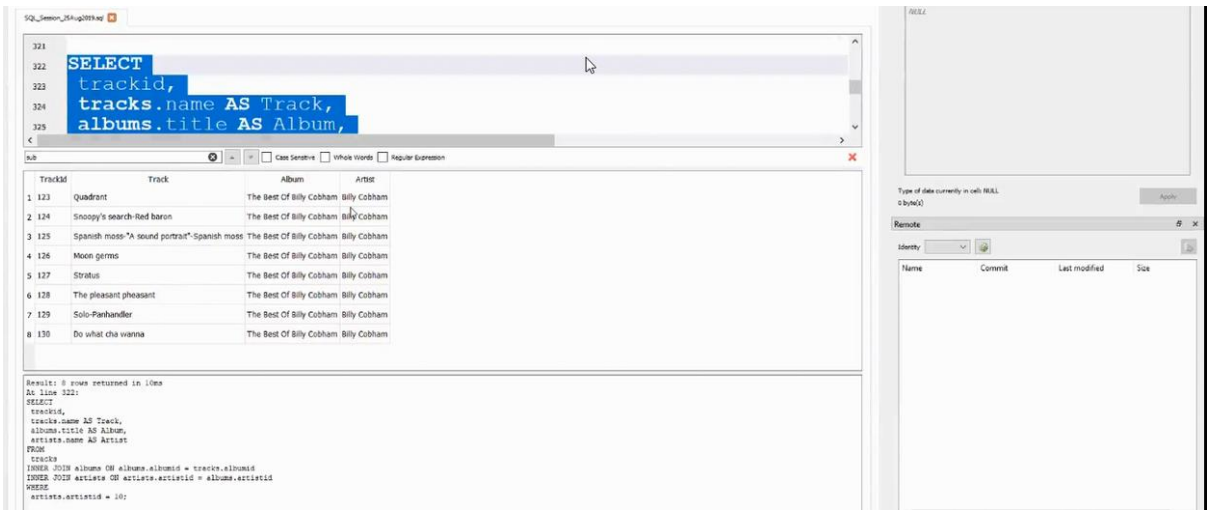
Usage of alias for table names



Getting albumids as well from both tables (to validate the join)



Joining multiple tables



```
-- SOLVE: Find the artists who do not have any albums
```

```
-- SOLVE: Find the artists who do not have any albums

SELECT
 artists.ArtistId,
 albumId
FROM
 artists
LEFT JOIN albums ON albums.artistid = artists.artistid
ORDER BY
 albumid;

--OR
 SELECT
```

Alternate way

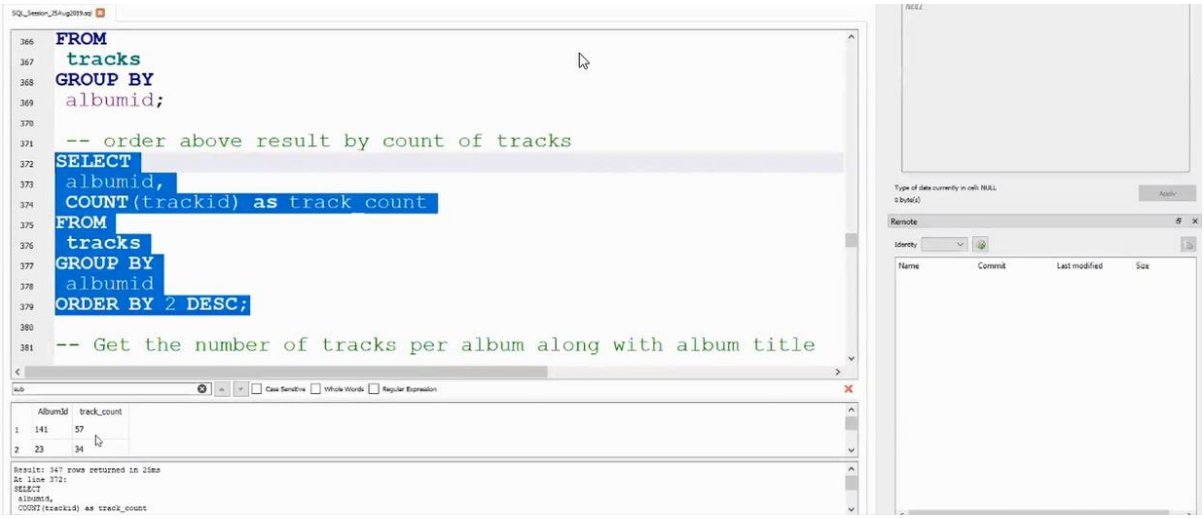
```
--OR
 SELECT
 artists.ArtistId,
 albumId
FROM
 artists
LEFT JOIN albums ON albums.artistid = artists.artistid
WHERE
 albumid IS NULL;
```

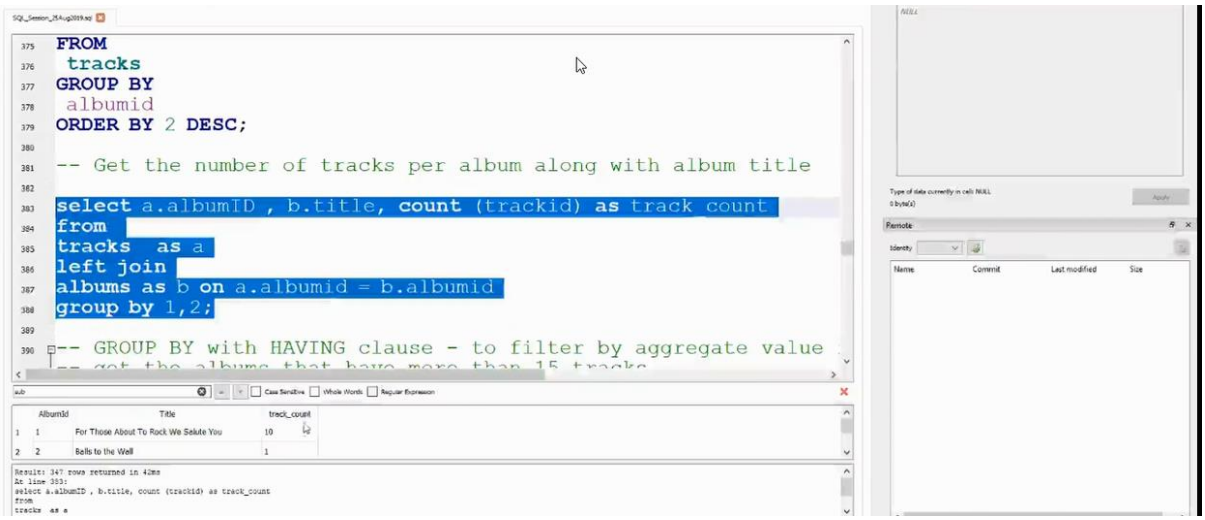
```
-- Note - SQLite does not support RIGHT JOIN and FULL OUTER jo
```

Groupby (like pivot table in excel)

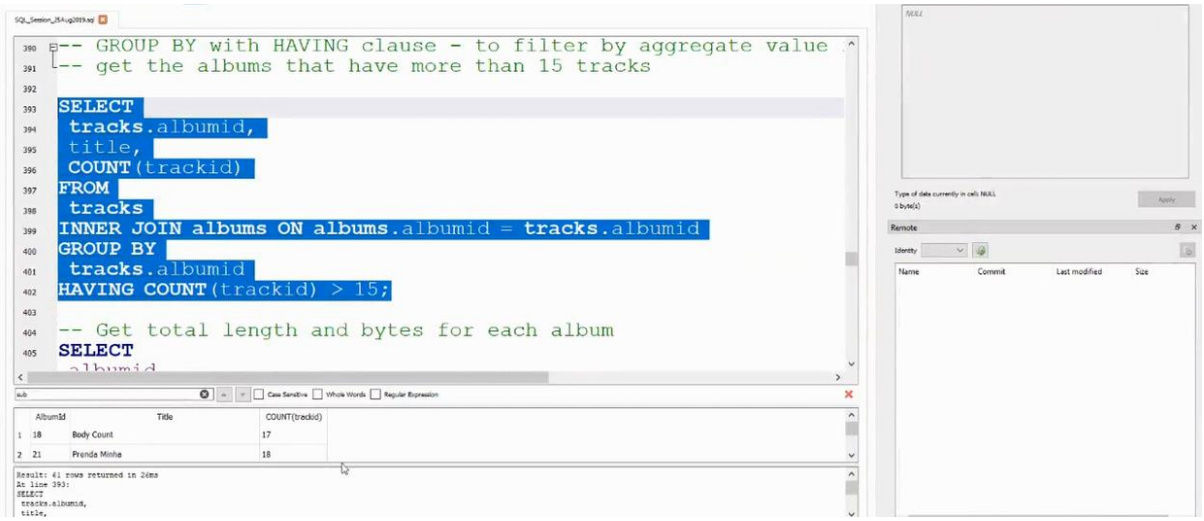
```
-- Note - SQLite does not support RIGHT JOIN and FULL OUTER jo

-- GROUP BY
-- Find the number of tracks per album

SELECT
 albumid,
 COUNT(trackid) as track_count
FROM
 tracks
GROUP BY
 albumid;

 -- order above result by count of tracks
SELECT
```

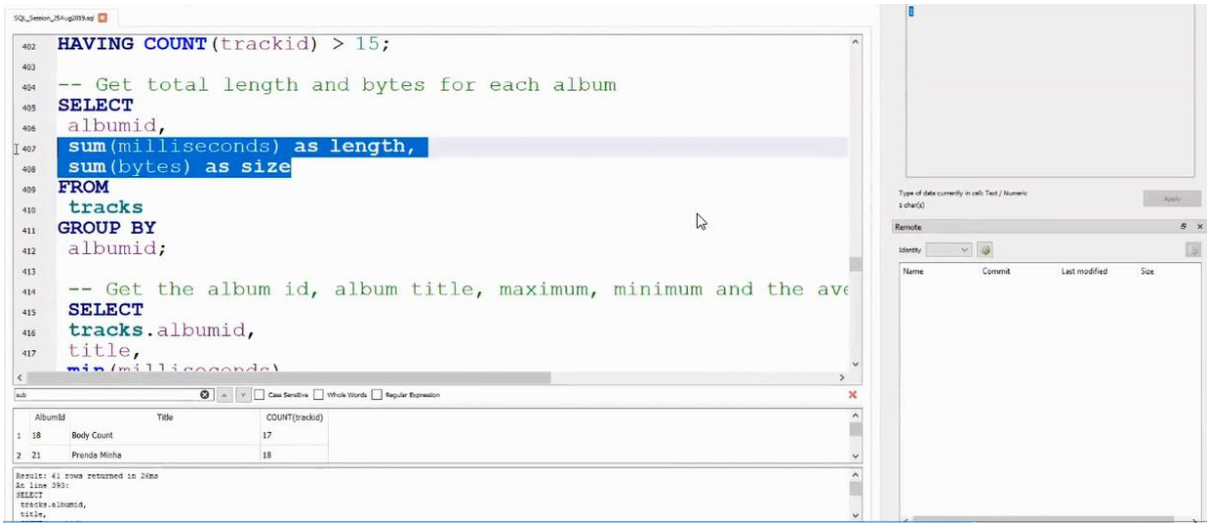
Sorting on top of groupby



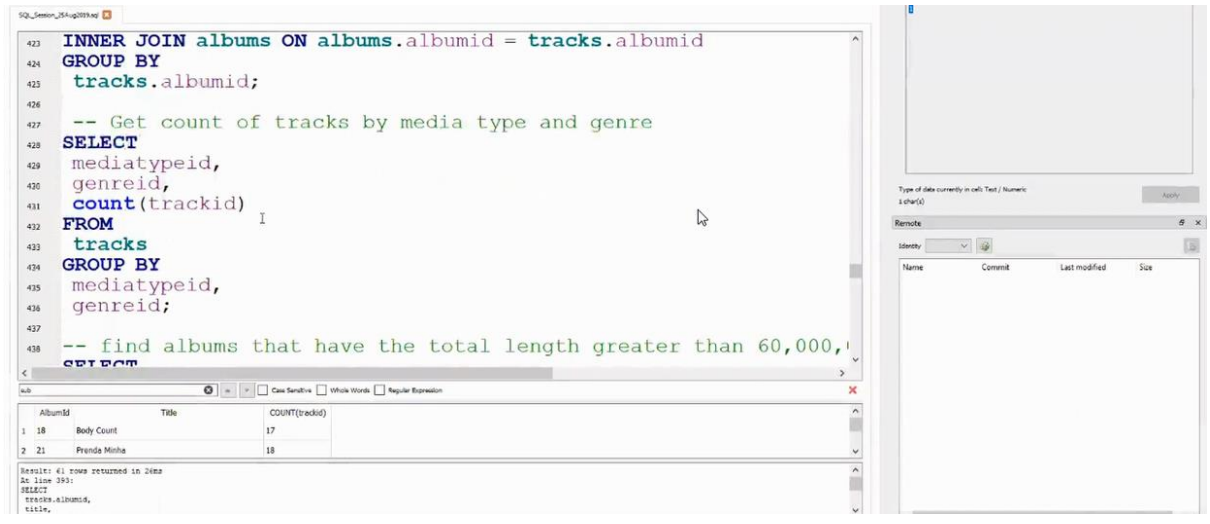


Having condition – to be applied on top of groupby to filter aggregated data

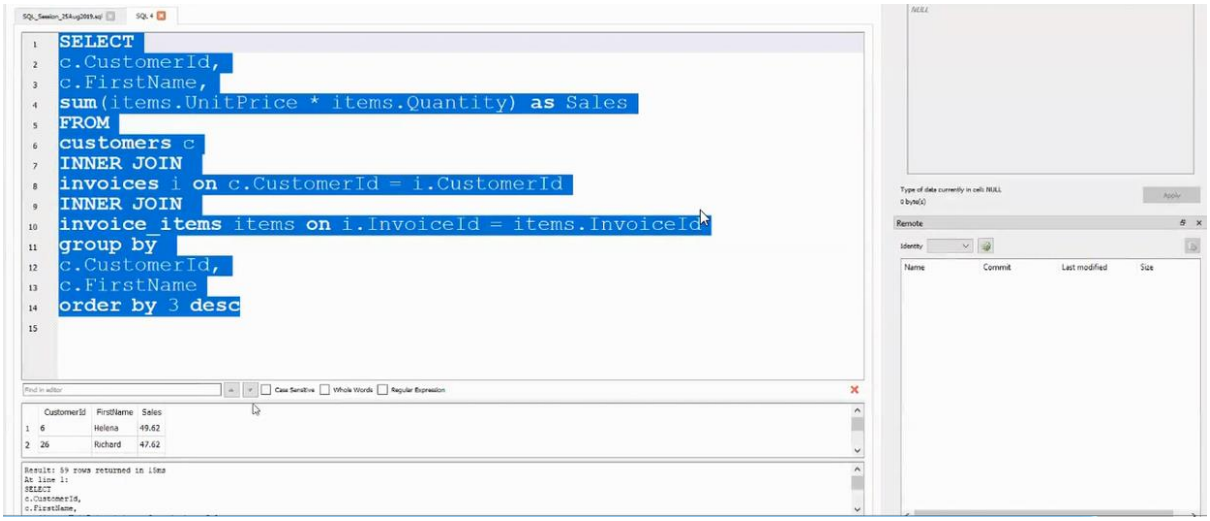



HAVING COUNT(trackid) > 15;
-- Get total length and bytes for each album
SELECT
albumid,
sum(milliseconds) as length,
sum(bytes) as size
FROM
tracks
GROUP BY
albumid;
-- Get the album id, album title, maximum, minimum and the ave
SELECT
tracks.albumid,
title,
AlbumId
Title
COUNT(trackid)
1 18 Body Count 17
2 21 Prenda Minha 18
Result: 41 rows returned in 26ms
At line 393:
SELECT
tracks.albumid,
title,

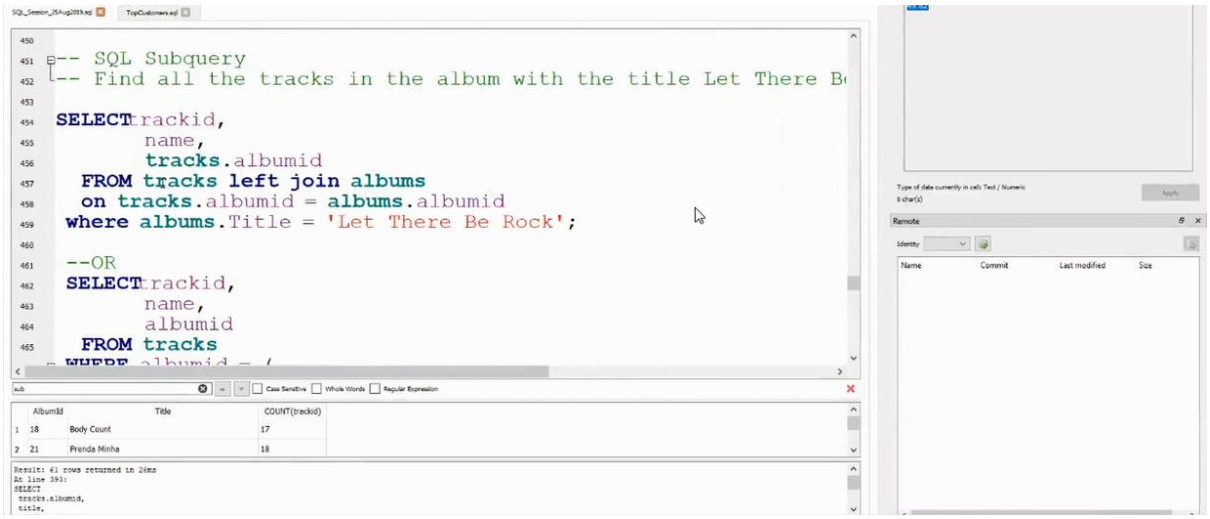

GROUP BY
albumid;
-- Get the album id, album title, maximum, minimum and the ave
SELECT
tracks.albumid,
title,
min(milliseconds),
max(milliseconds),
round(avg(milliseconds),2)
FROM
tracks
INNER JOIN albums ON albums.albumid = tracks.albumid
GROUP BY
tracks.albumid;
AlbumId
Title
COUNT(trackid)
1 18 Body Count 17
2 21 Prenda Minha 18
Result: 41 rows returned in 26ms
At line 393:
SELECT
tracks.albumid,
title,

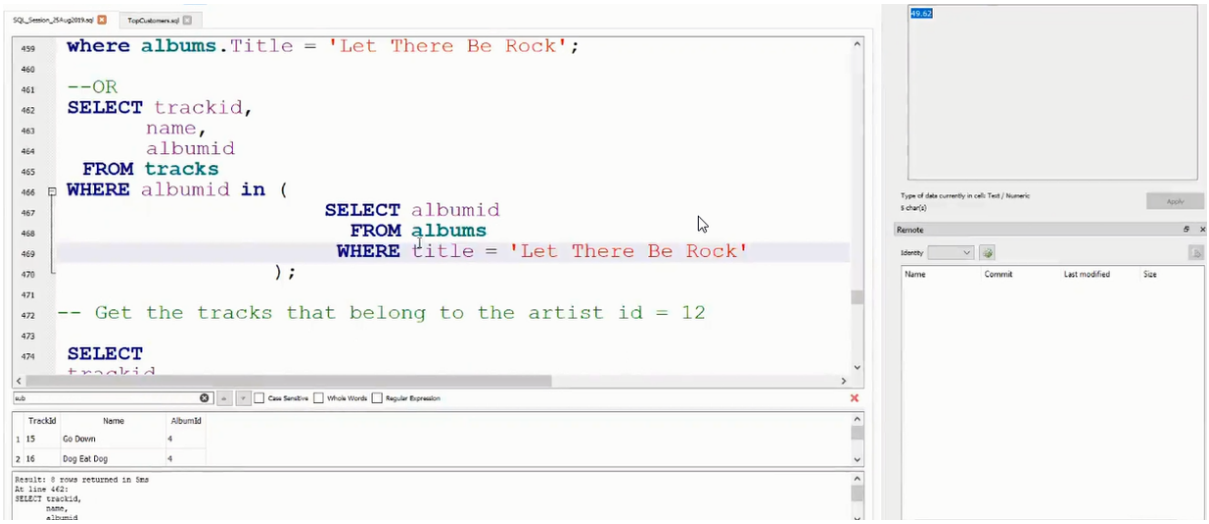

INNER JOIN albums ON albums.albumid = tracks.albumid
GROUP BY
tracks.albumid;
-- Get count of tracks by media type and genre
SELECT
mediatypeid,
genreid,
count(trackid)
FROM
tracks
GROUP BY
mediatypeid,
genreid;
-- find albums that have the total length greater than 60,000,
AlbumId
Title
COUNT(trackid)
1 18 Body Count 17
2 21 Prenda Minha 18
Result: 41 rows returned in 26ms
At line 393:
SELECT
tracks.albumid,
title,

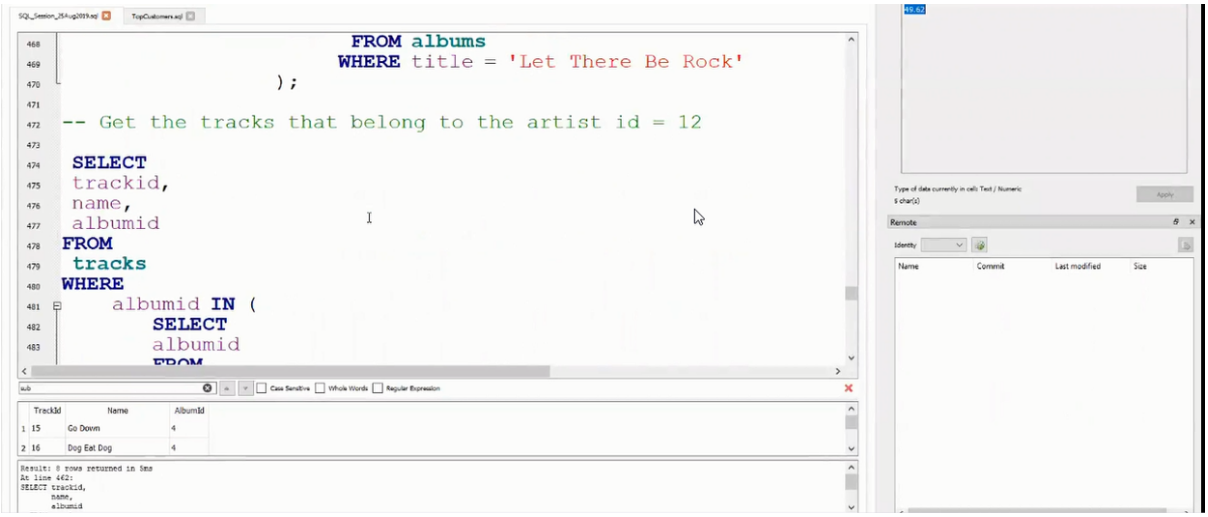
Top 5 customers by sales?

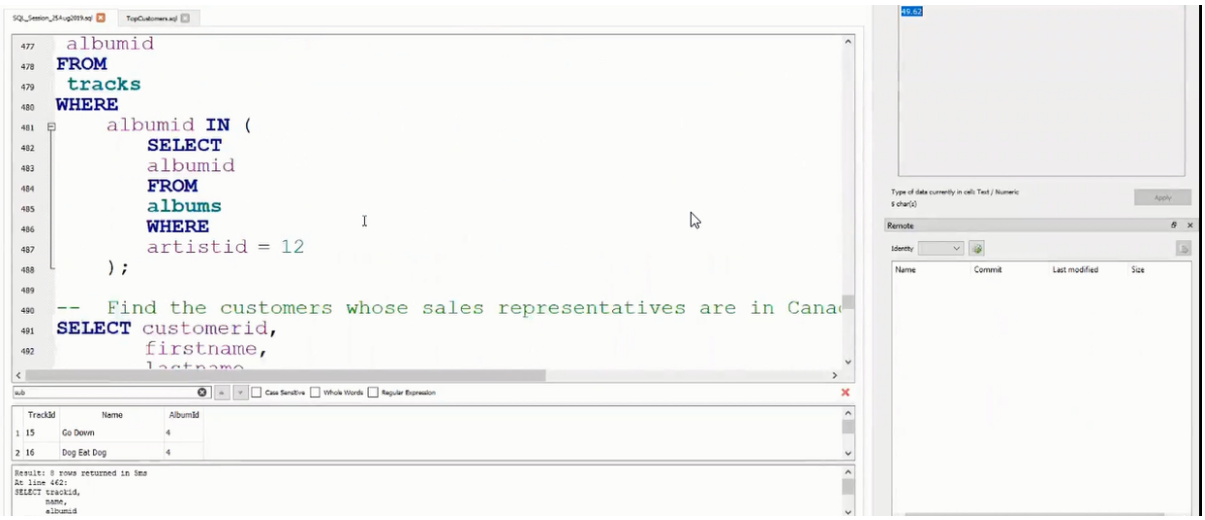


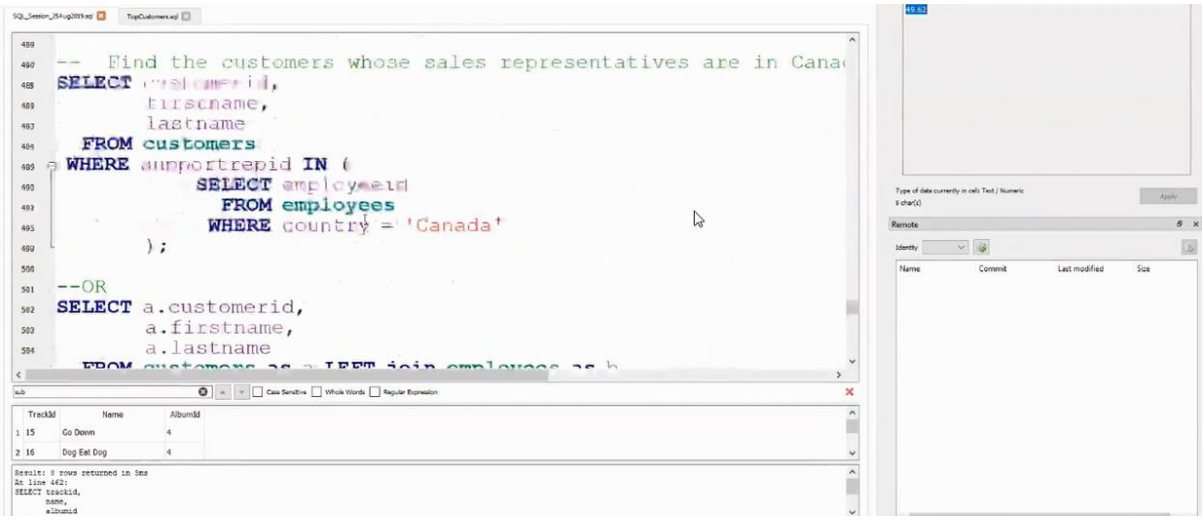
SQL Subqueries

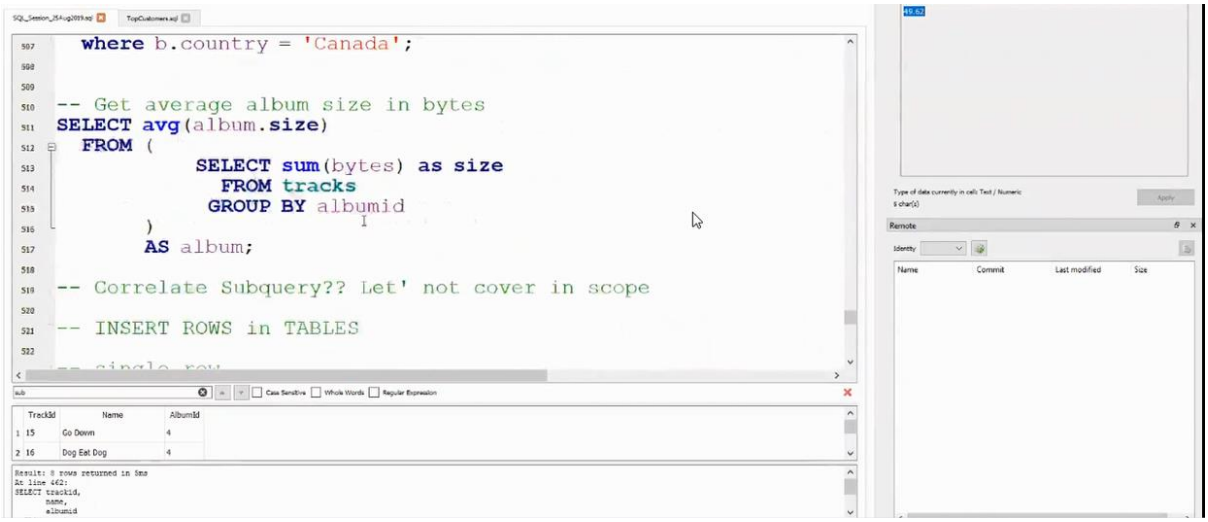




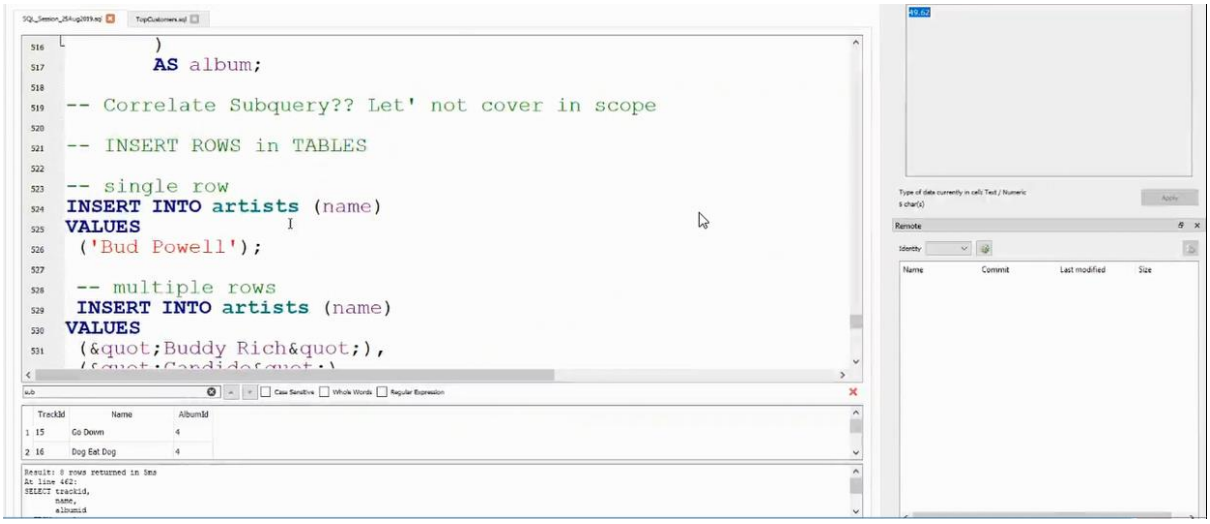

FROM albums
WHERE title = 'Let There Be Rock'
);
-- Get the tracks that belong to the artist id = 12
SELECT
trackid,
name,
albumid
FROM
tracks
WHERE
albumid IN (
SELECT
albumid
TrackId Name AlbumId
1 15 Go Down 4
2 16 Dog Eat Dog 4
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
name,
albumid

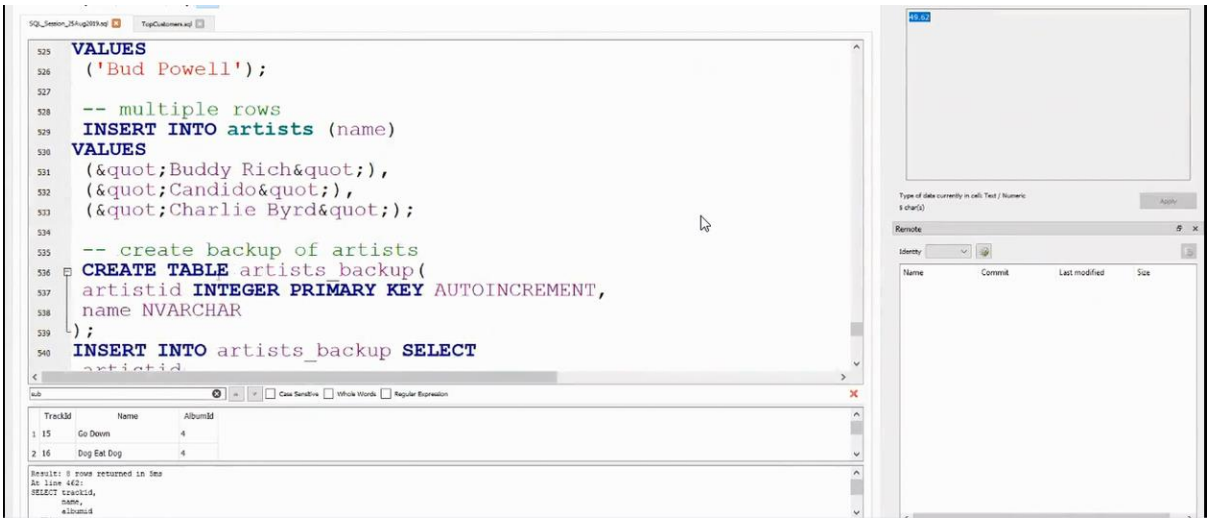

albumid
FROM
tracks
WHERE
albumid IN (
SELECT
albumid
FROM
albums
WHERE
artistid = 12
);
-- Find the customers whose sales representatives are in Cana
SELECT customerid,
firstname,
TrackId Name AlbumId
1 15 Go Down 4
2 16 Dog Eat Dog 4
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
name,
albumid


-- Find the customers whose sales representatives are in Cana
SELECT customerid,
firstname,
lastname
FROM customers
WHERE supportrepid IN (
SELECT employeeid
FROM employees
WHERE country = 'Canada'
);
--OR
SELECT a.customerid,
a.firstname,
a.lastname
TrackId Name AlbumId
1 15 Go Down 4
2 16 Dog Eat Dog 4
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
name,
albumid

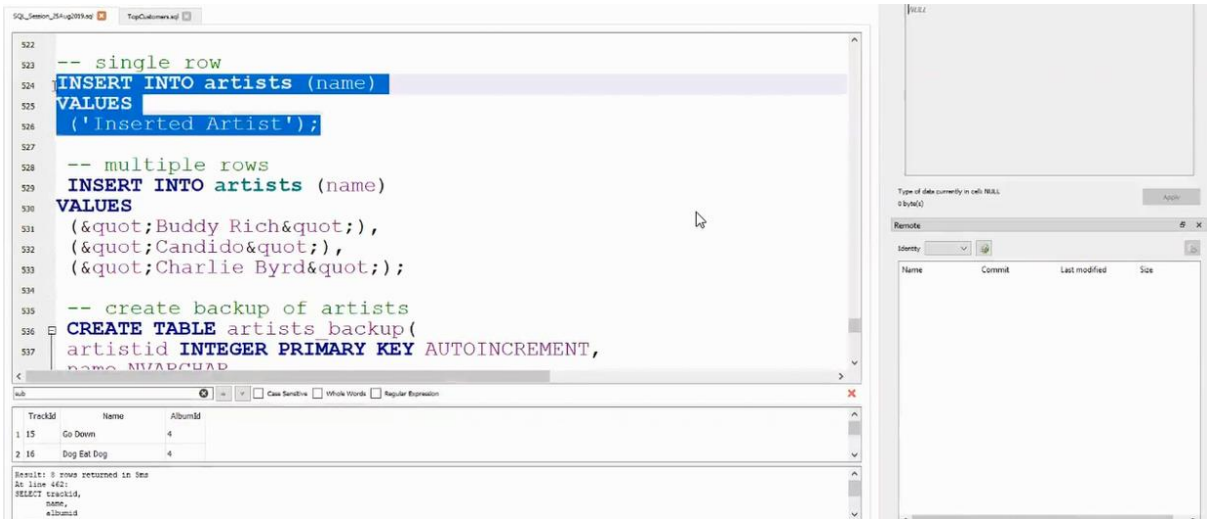


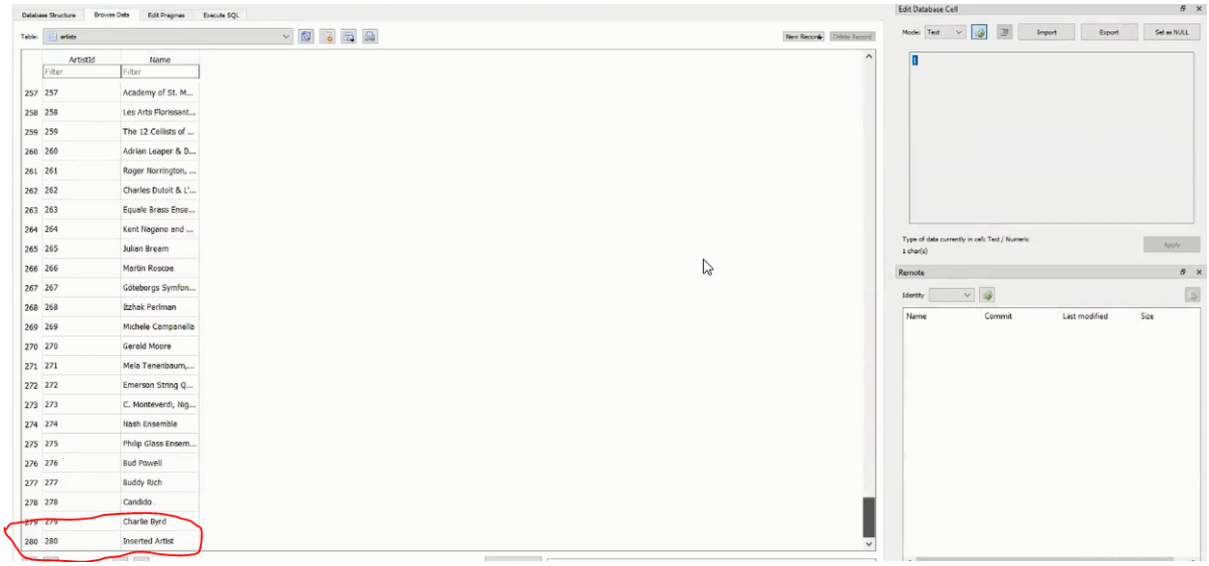
Correlate subquery





To insert values in a table



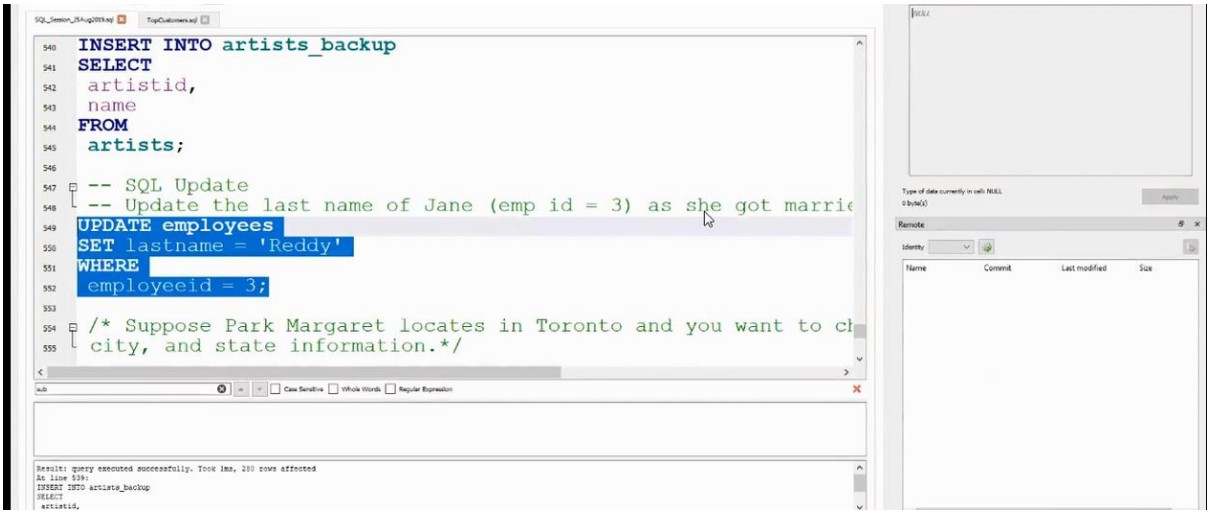


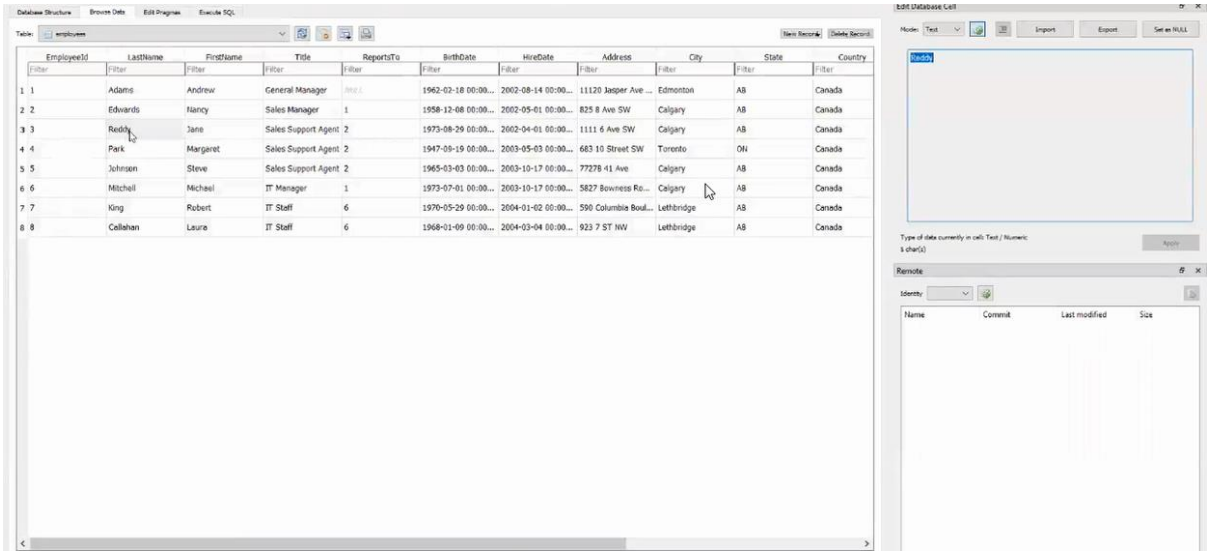
Insert multiple values in a table

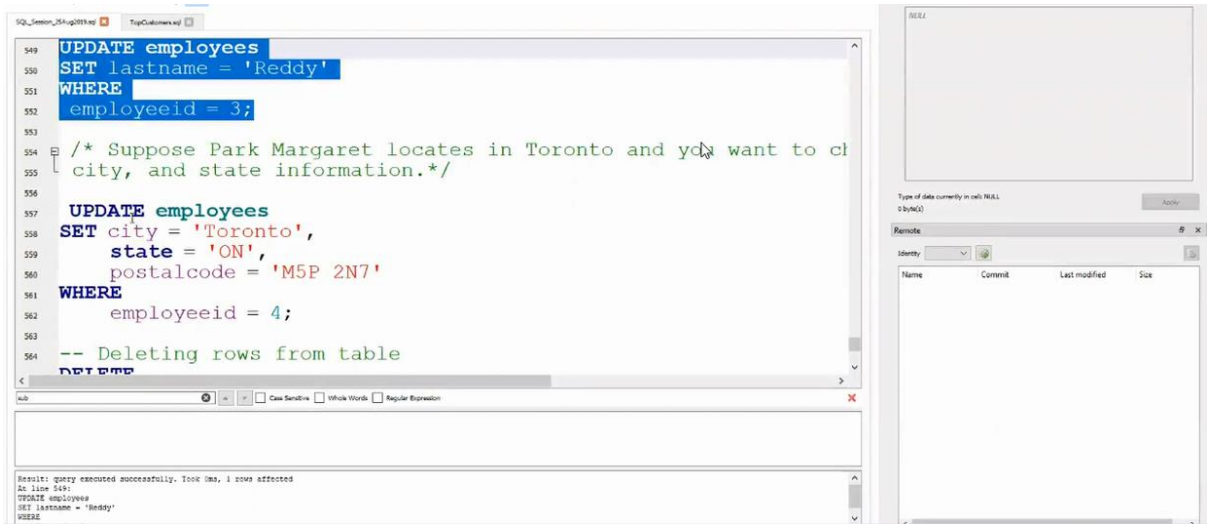
Create new table – artists backup is table created from artist table

New rows can be inserted into new table from original tables

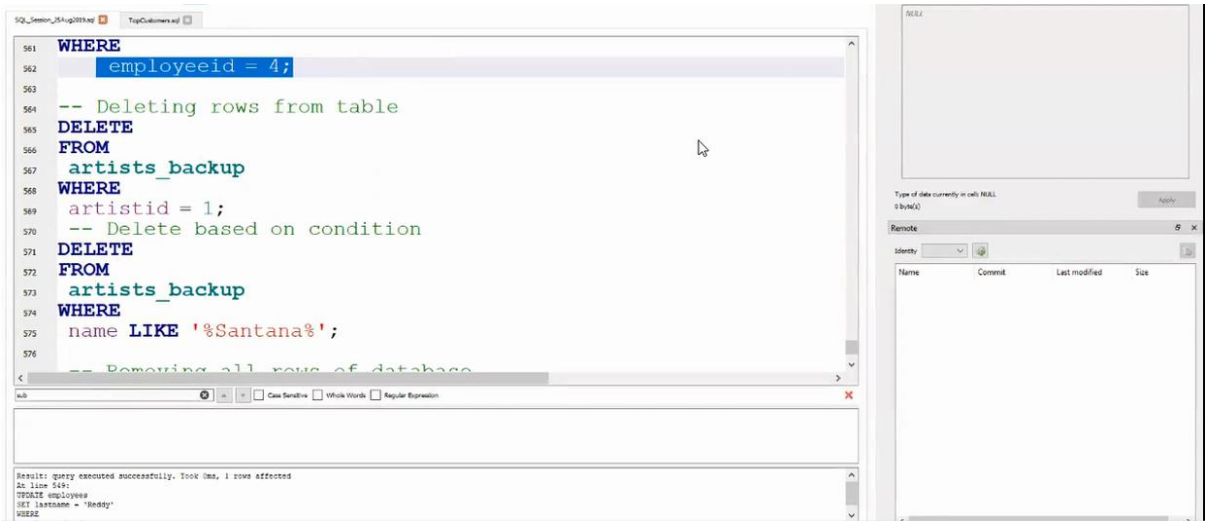
Update values in a table



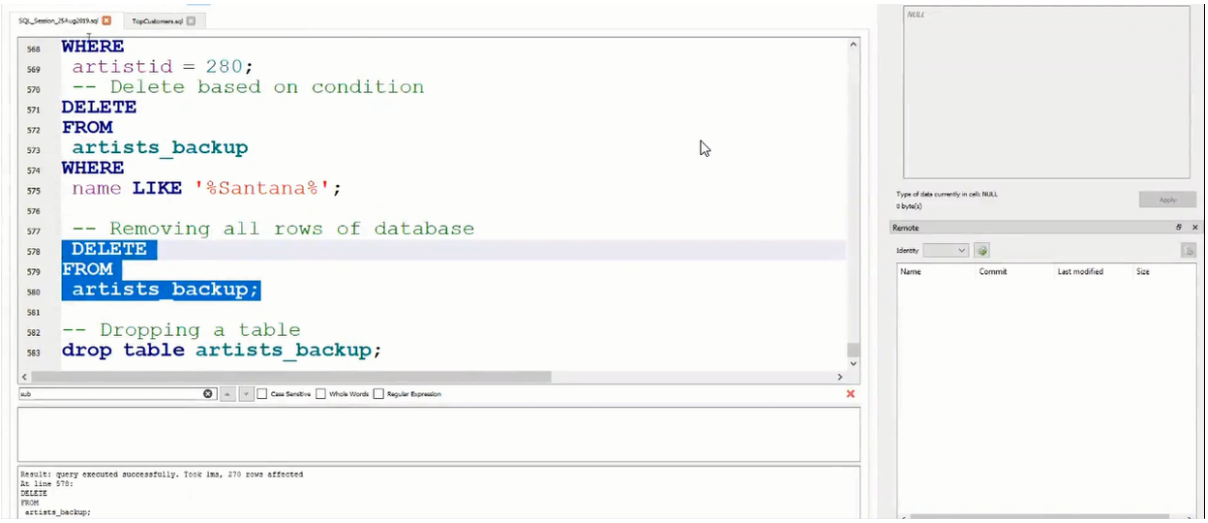




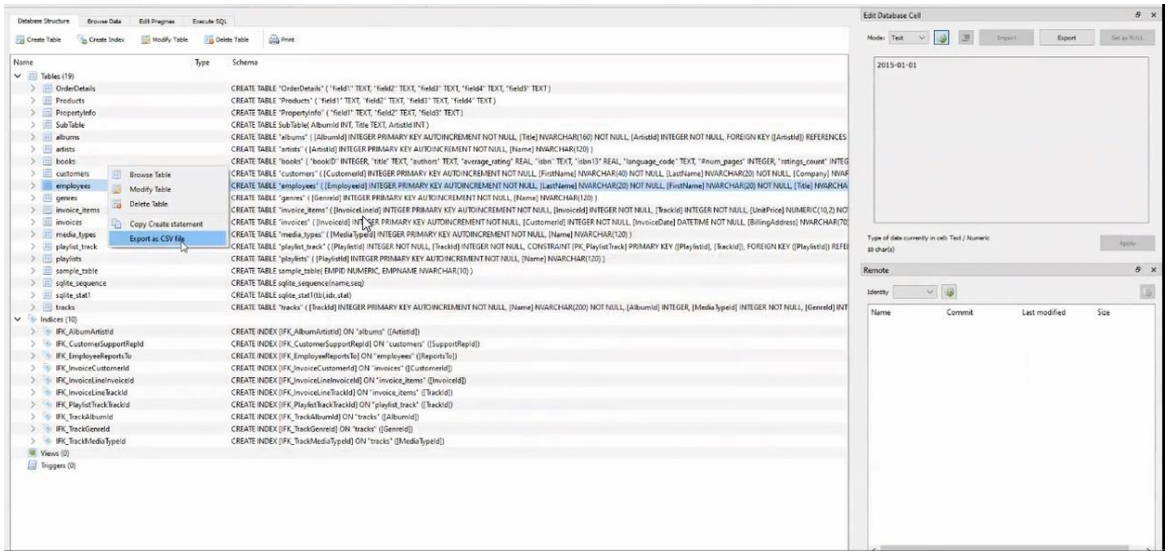
Deleting rows



Delete all the rows and dropping a table



Export table as csv (right click on table)



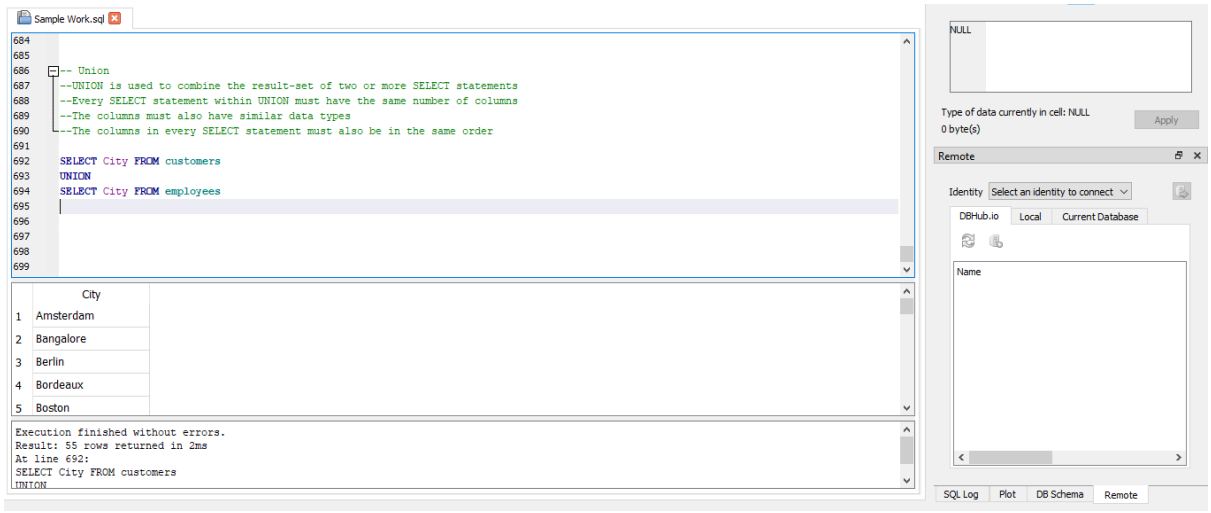
Union

UNION is used to combine the result-set of two or more SELECT statements

Every SELECT statement within UNION must have the same number of columns

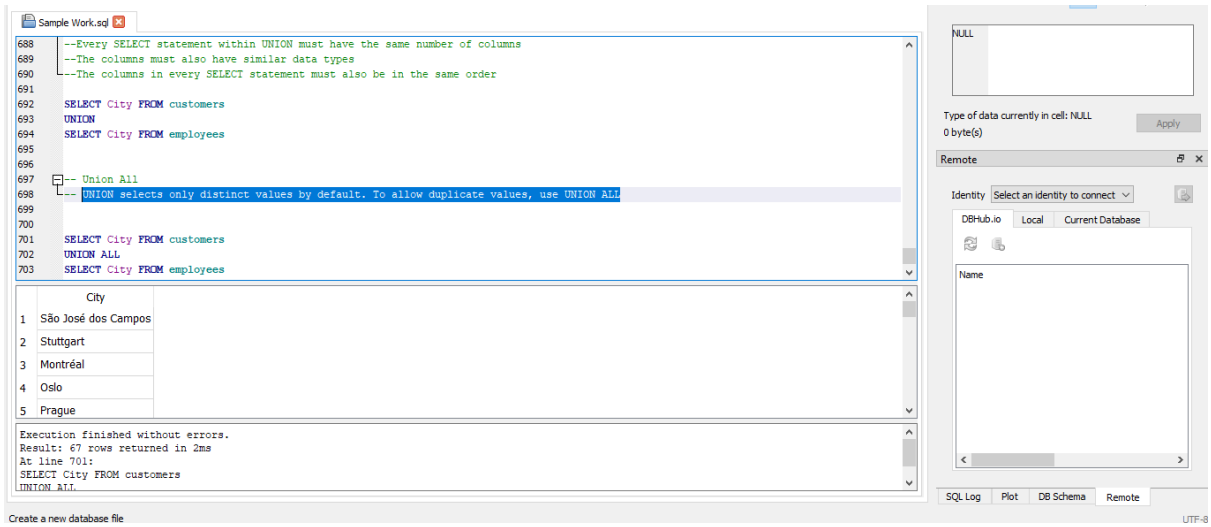
The columns must also have similar data types

The columns in every SELECT statement must also be in the same order



Union All

UNION selects only distinct values by default. To allow duplicate values, use UNION ALL



SQL Set Operators (Union, Union All, Minus, Intersect)

https://learnsql.com/blog/introducing-sql-set-operators-union-union-minus-intersect/

Stored Procedure

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So, if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Stored Procedure Example

The following SQL statement creates a stored procedure named "SelectAllCustomers" that selects all records from the "Customers" table:

Example

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Execute the stored procedure above as follows:

Example

```
EXEC SelectAllCustomers;
```

Stored Procedure With One Parameter

The following SQL statement creates a stored procedure that selects Customers from a particular City from the "Customers" table:

Example

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
SELECT * FROM Customers WHERE City = @City
GO;
```

Execute the stored procedure above as follows:

Example

```
EXEC SelectAllCustomers @City = 'London';
```

Stored Procedure With Multiple Parameters

Setting up multiple parameters is very easy. Just list each parameter and the data type separated by a comma as shown below.

The following SQL statement creates a stored procedure that selects Customers from a particular City with a particular PostalCode from the "Customers" table:

Example

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND PostalCode = @PostalCode
GO;
```

Execute the stored procedure above as follows:

Example

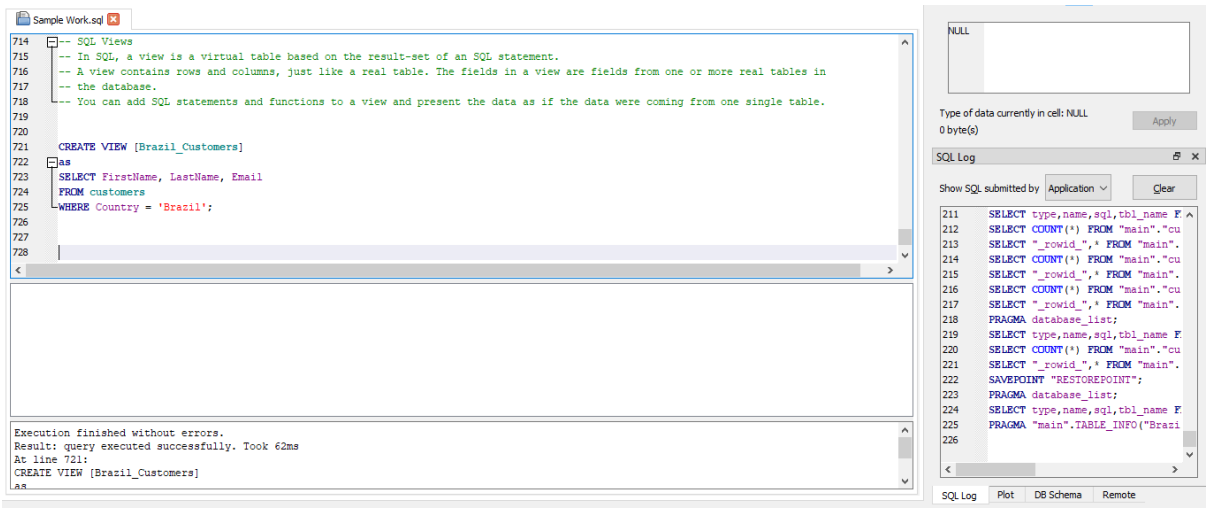
```
EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';
```

SQL Views

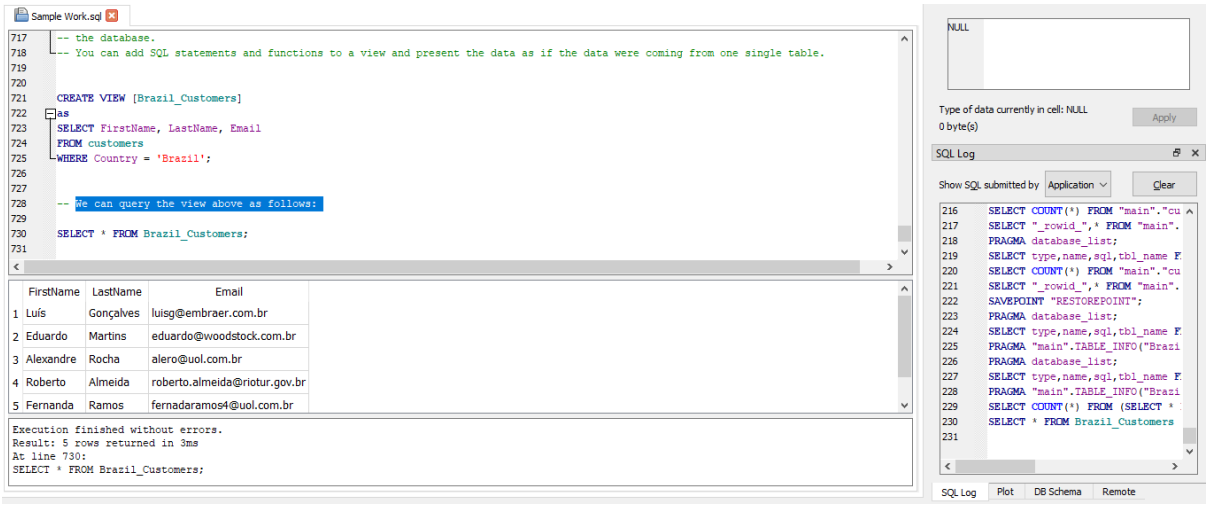
In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

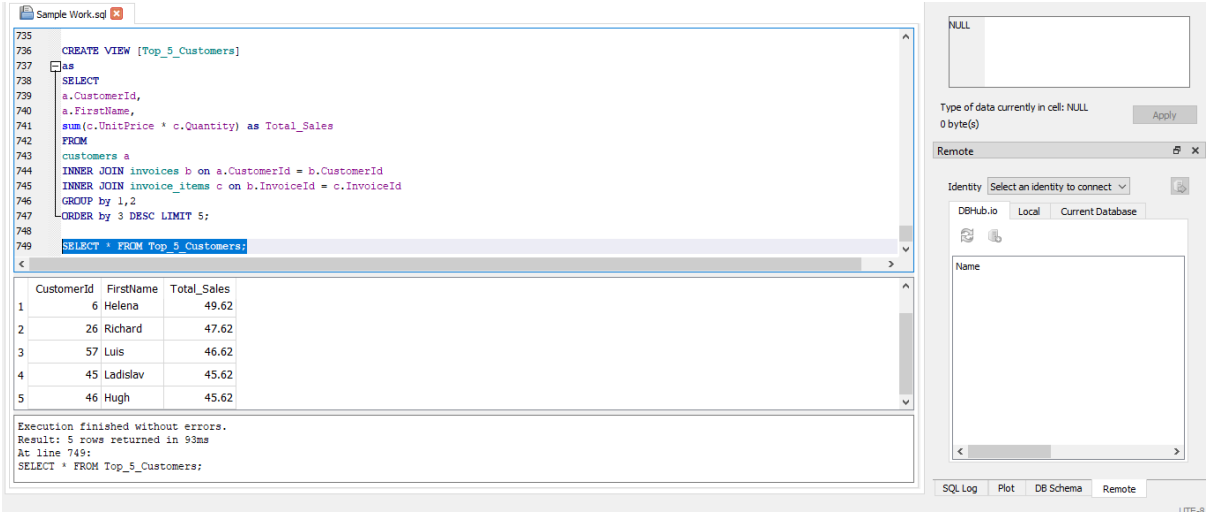
You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.



We can query the view above as follows:



View: Top 5 Customers by Sales



SQL Set Operators (Union, UnionAll, Minus, Intersect)

https://learnsql.com/blog/introducing-sql-set-operators-union-union-minus-intersect/

SQL Practice Exercises

https://www.w3resource.com/sql-exercises/sql-joins-exercises.php

https://www.w3resource.com/sql-exercises/joins-hr/index.php

Types of Indexes (Clustered vs Non-Clustered Indexes)

https://www.spotlightcloud.io/blog/when-to-use-clustered-or-non-clustered-indexes-in-sql-server

Types of Views (Materialised vs Non-Materialised)

https://www.besanttechnologies.com/difference-between-view-and-materialized-view

Replace Values in a Column

```
Update [Table Name]
Set [Column Name] = New Value
where [Column Name] = Old Value;
```

Delete Rows with Null values

```
Delete from [Table Name]
where [Column Name] is null;
```

Replace Column Name in a Table

```
Alter Table [Table Name]
Rename Column Old Column Name To New Column Name
```

Transaction Query Language (TQL) / Transaction Control Language (TCL)

https://sebhastian.com/mysql-transaction/

https://www.mysqltutorial.org/mysql-transaction.aspx

4. **TCL(transaction Control Language) :** TCL commands deals with the transaction within the database.

   Examples of TCL commands:

   - **COMMIT**– commits a Transaction.
   - **ROLLBACK**– rollbacks a transaction in case of any error occurs.
   - **SAVEPOINT**–sets a savepoint within a transaction.
   - **SET TRANSACTION**–specify characteristics for the transaction.

Triggers

https://www.geeksforgeeks.org/sql-trigger-student-database/

https://learn.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver16

https://docs.oracle.com/cd/B19306_01/server.102/b14220/triggers.htm

Common Table Expressions (CTE)

https://www.geeksforgeeks.org/cte-in-sql/

https://www.sqlshack.com/sql-server-common-table-expressions-cte/