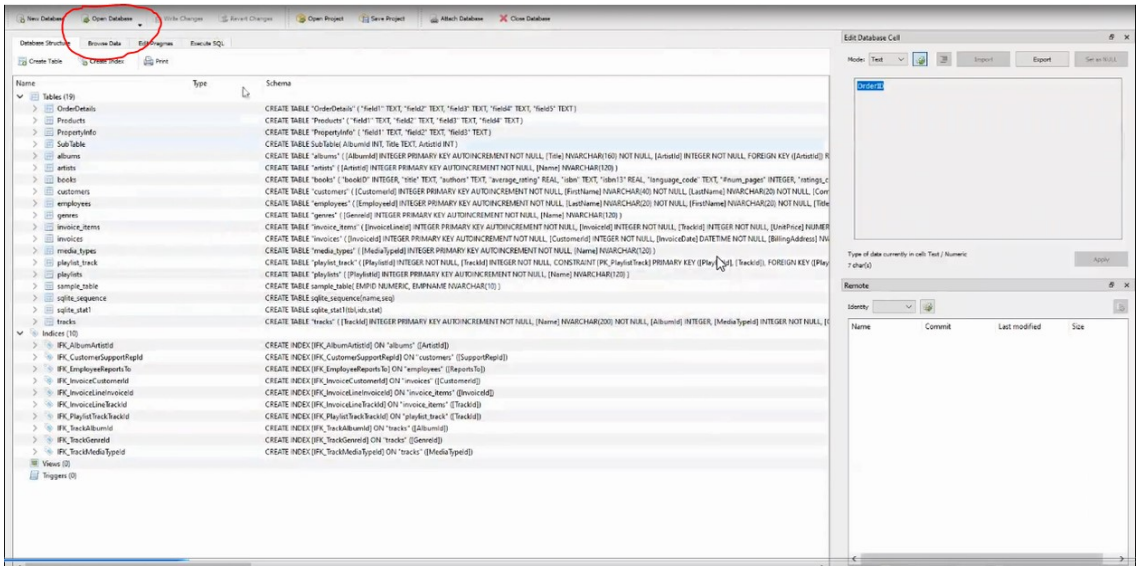
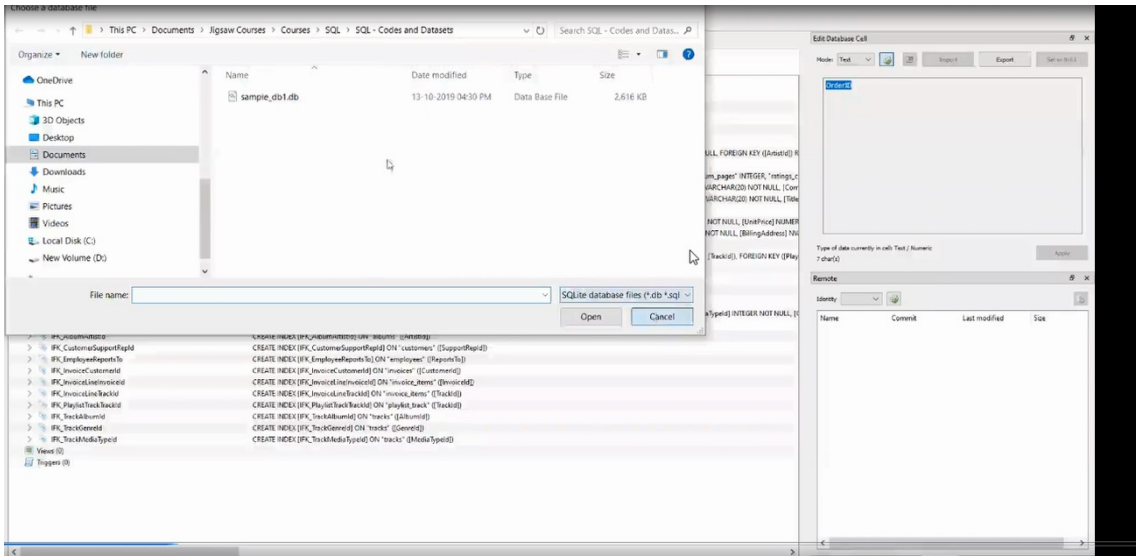
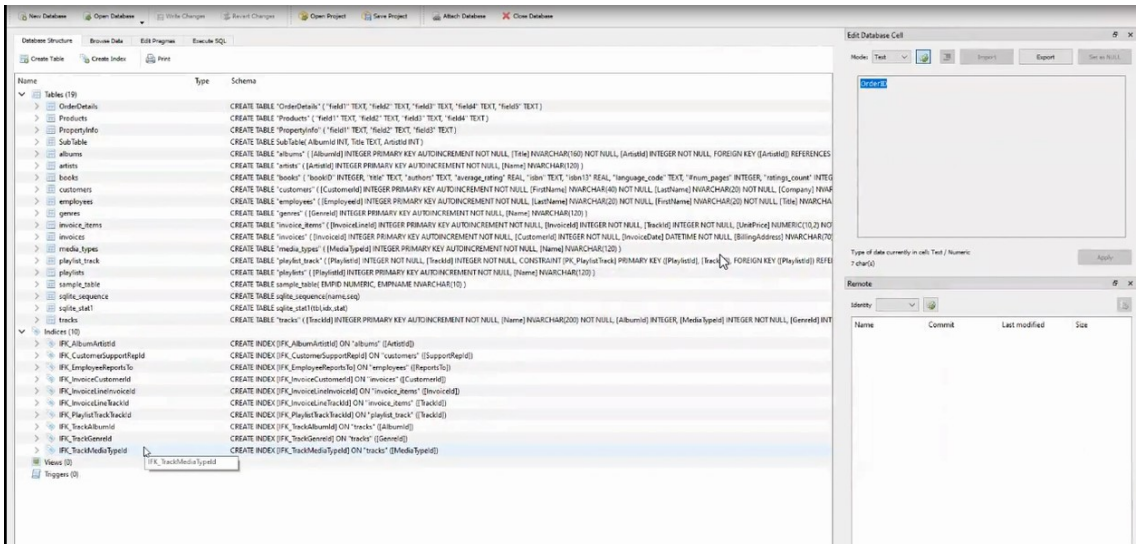
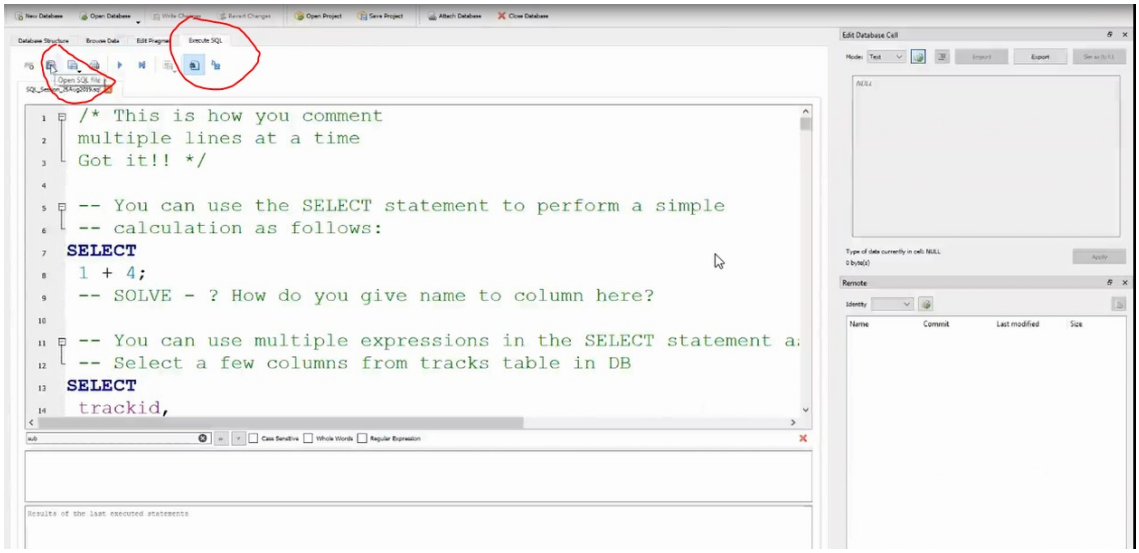
Open Database -> go to the db location -> open file (.db file)







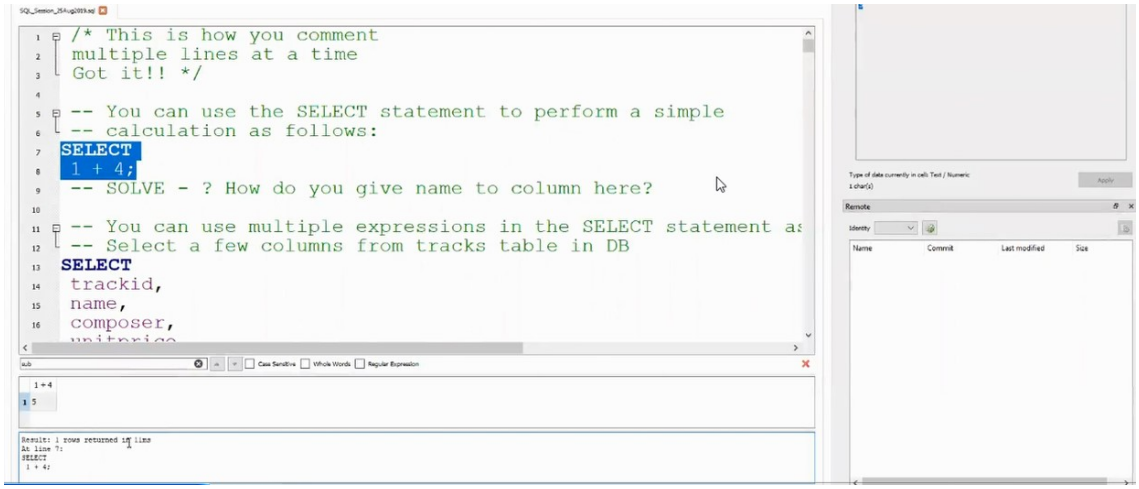
Execute SQL -> open SQL file -> go to location -> open file (.sql file)



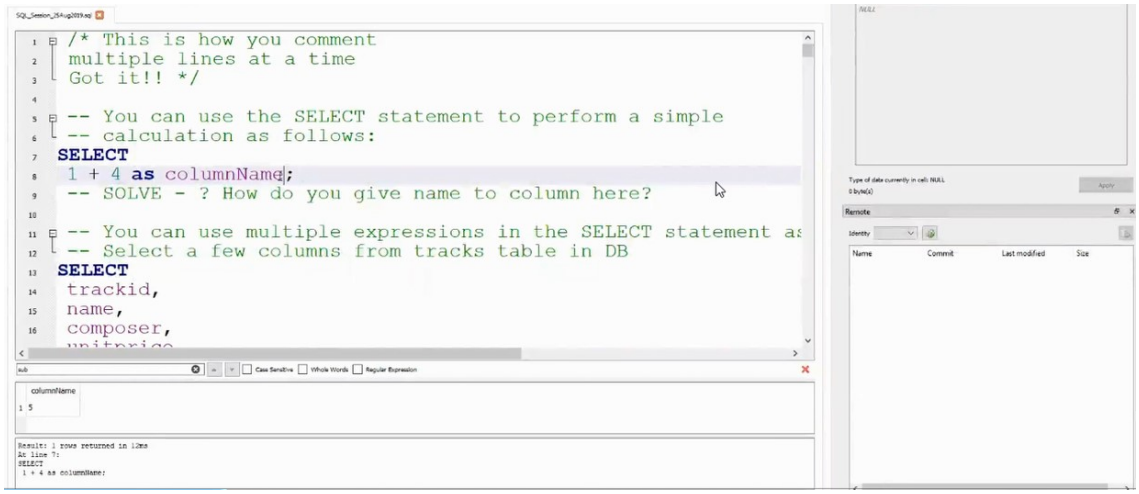
Database structure tab -> all tables are listed over here

Execute SQL -> code is written over here

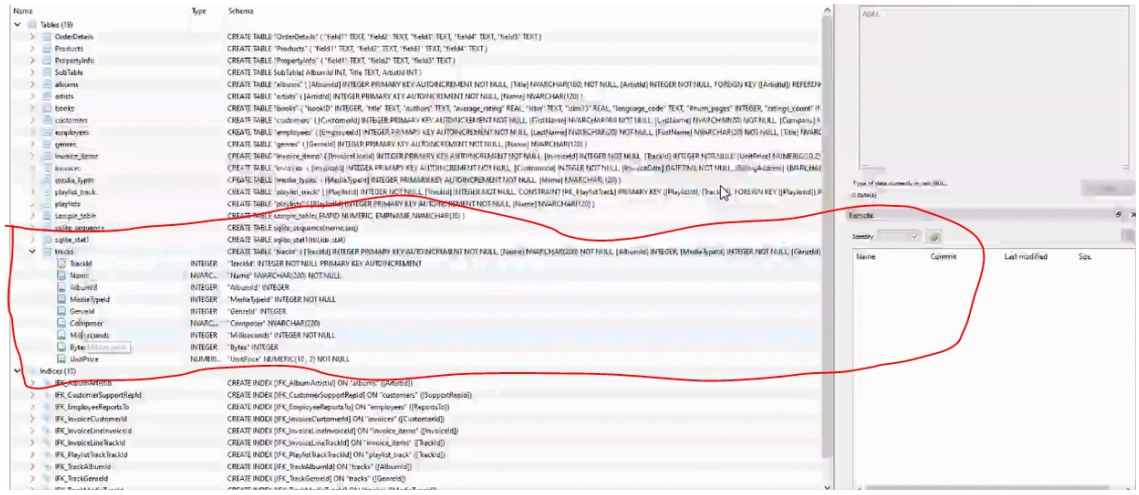
Simple math calculation



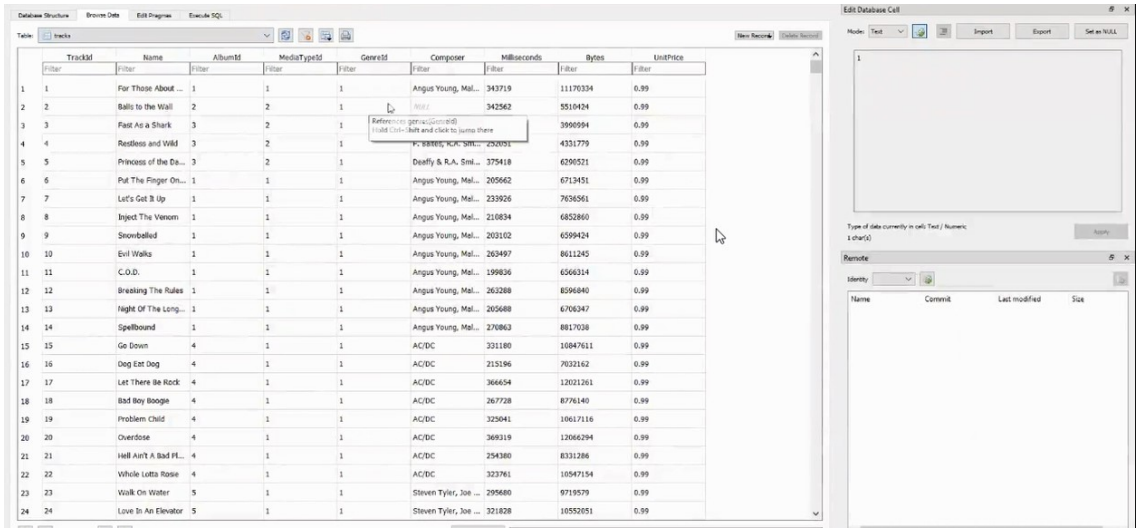
Columnname is available with this piece of code



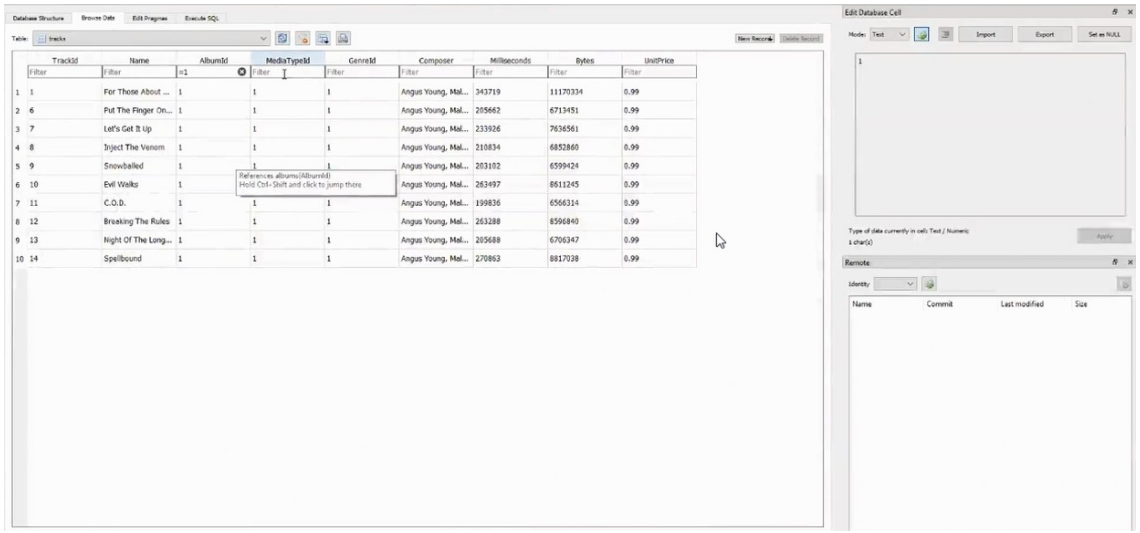
Trackid table

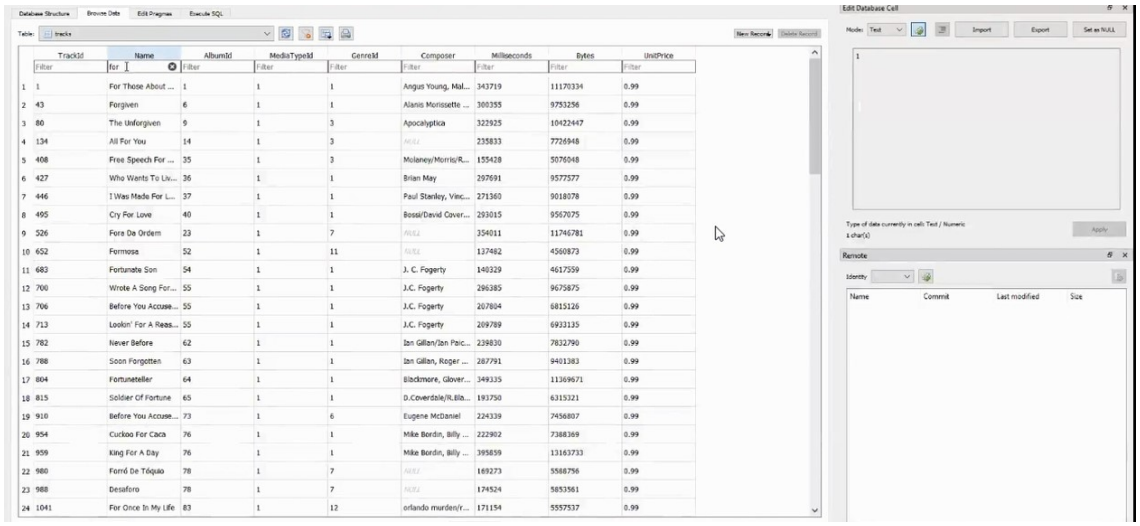


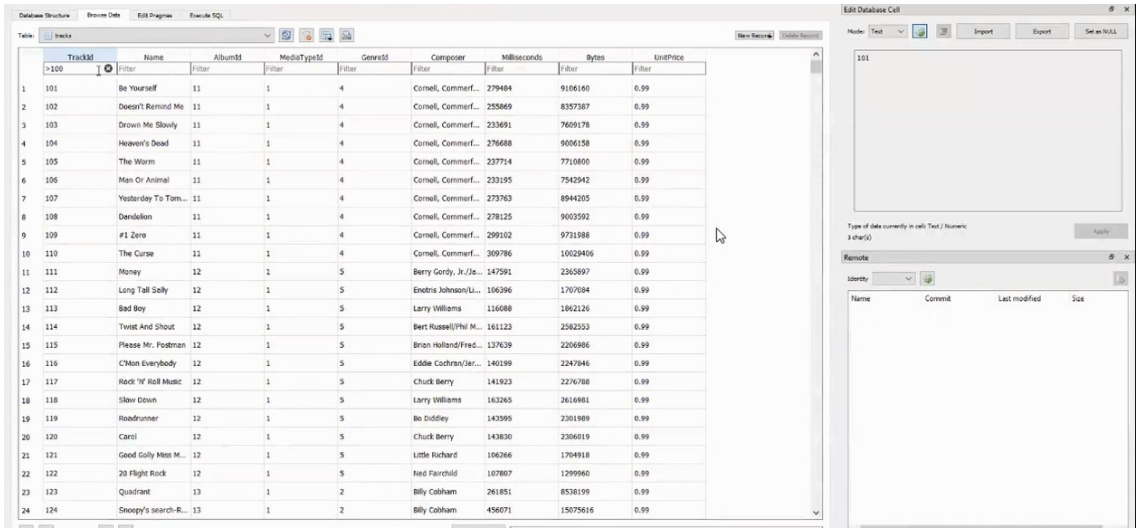
Browse data tab -> to have a look at the particular table



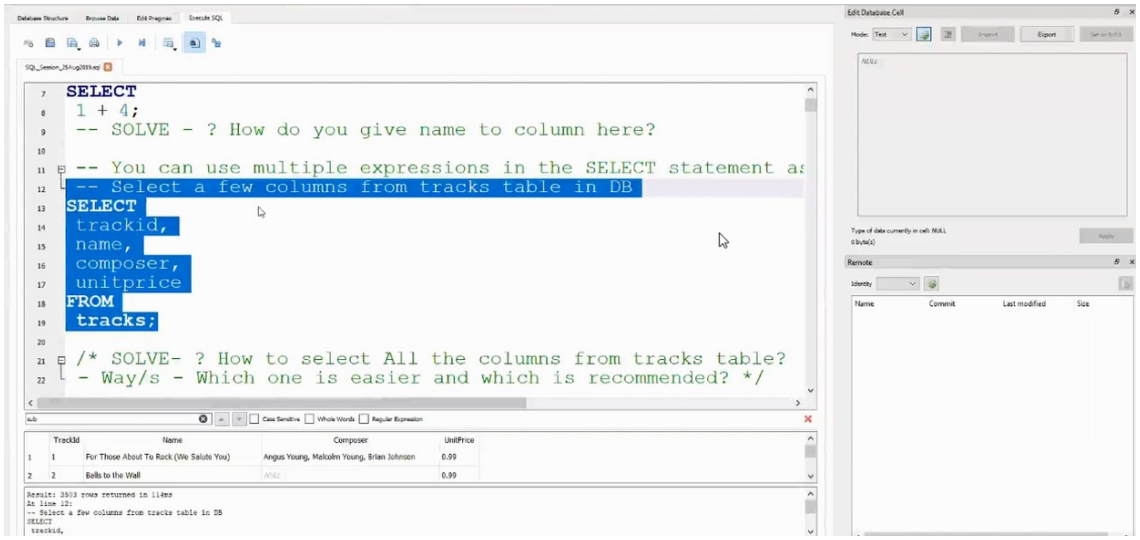
Data can be filtered over here



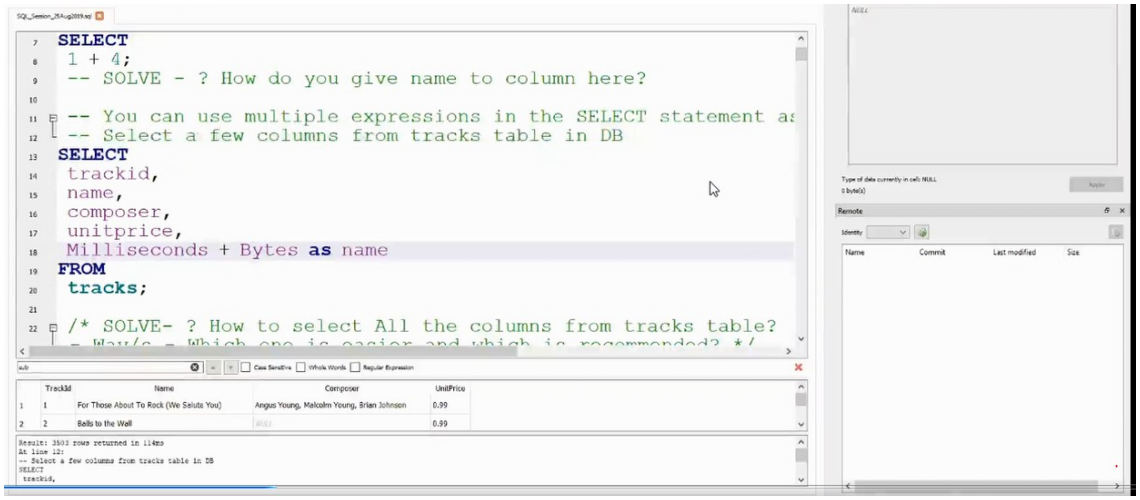




Selecting few columns from the table (select * to select all the columns)

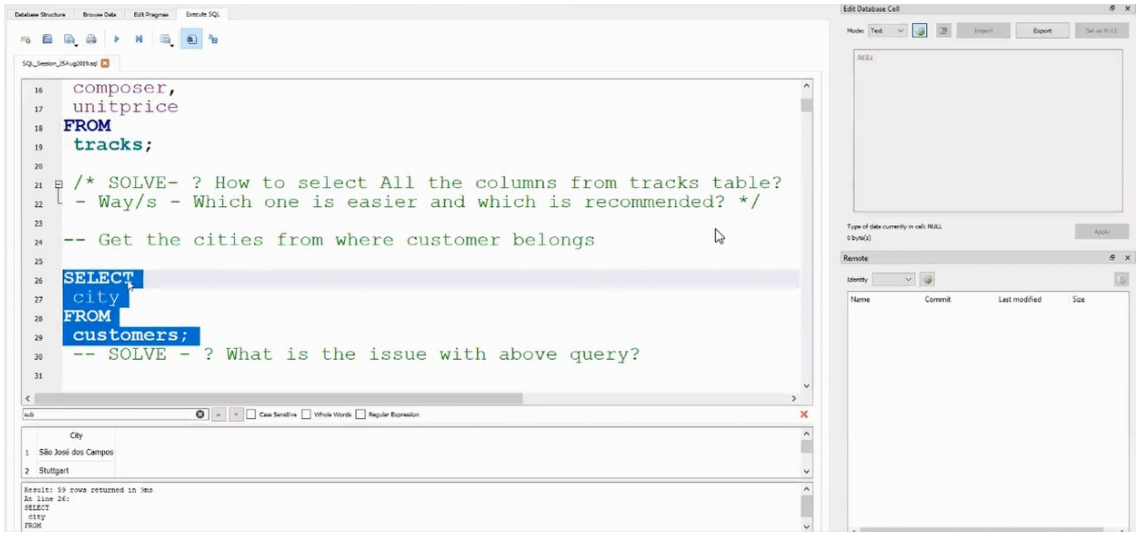


Addition of two columns

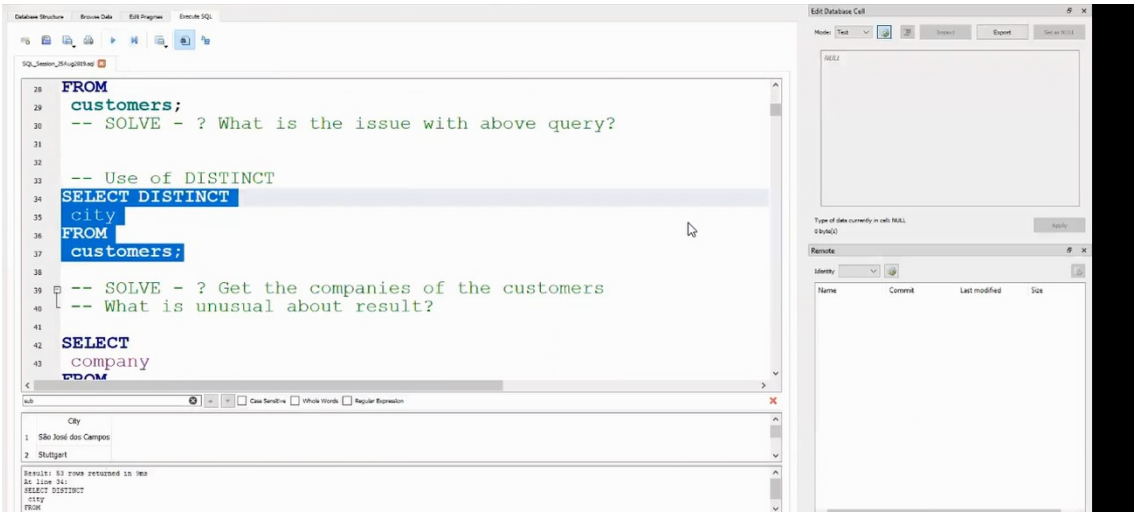


This code will select the all the rows and there will be duplicate city names present in the result

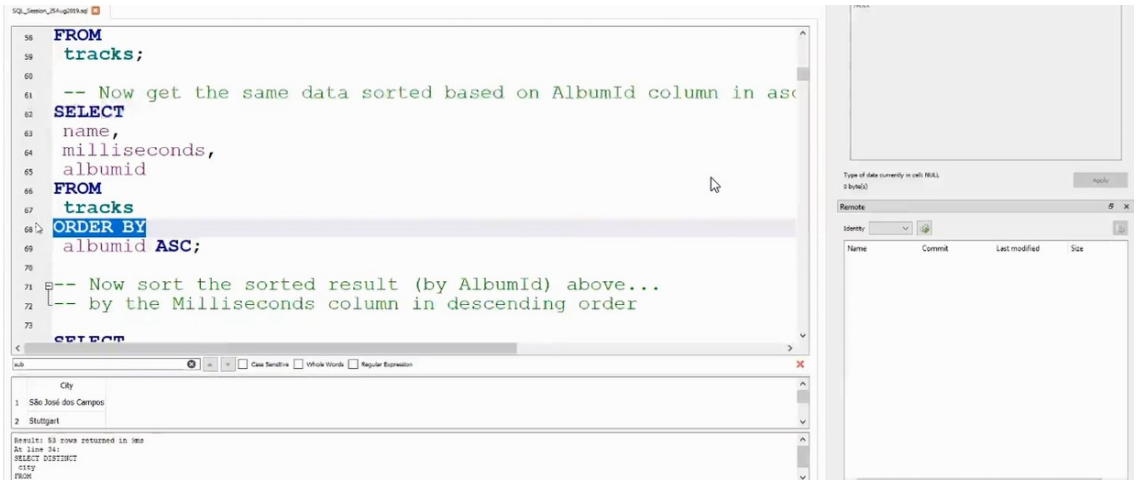
(59 rows)



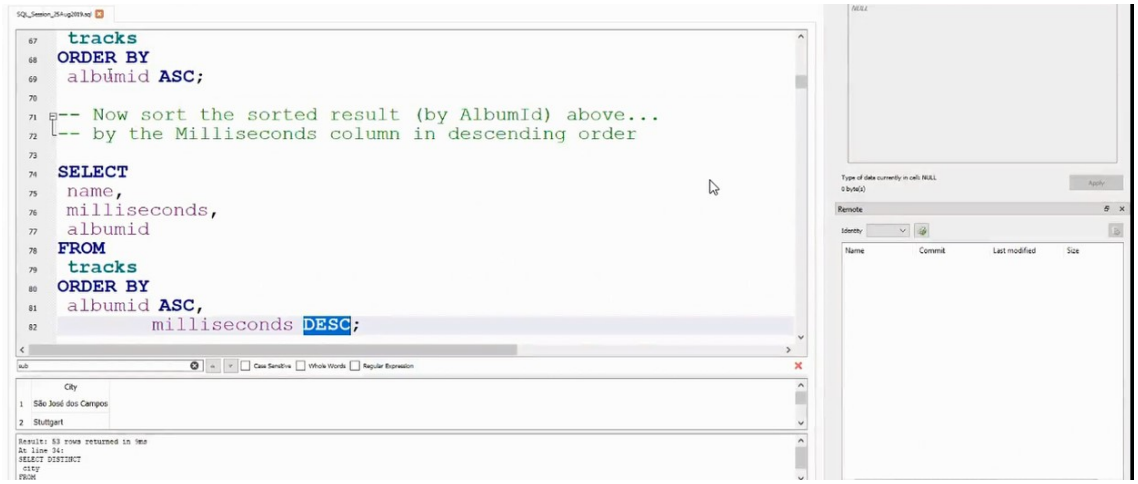
Distinct will select only unique citys (53 rows)



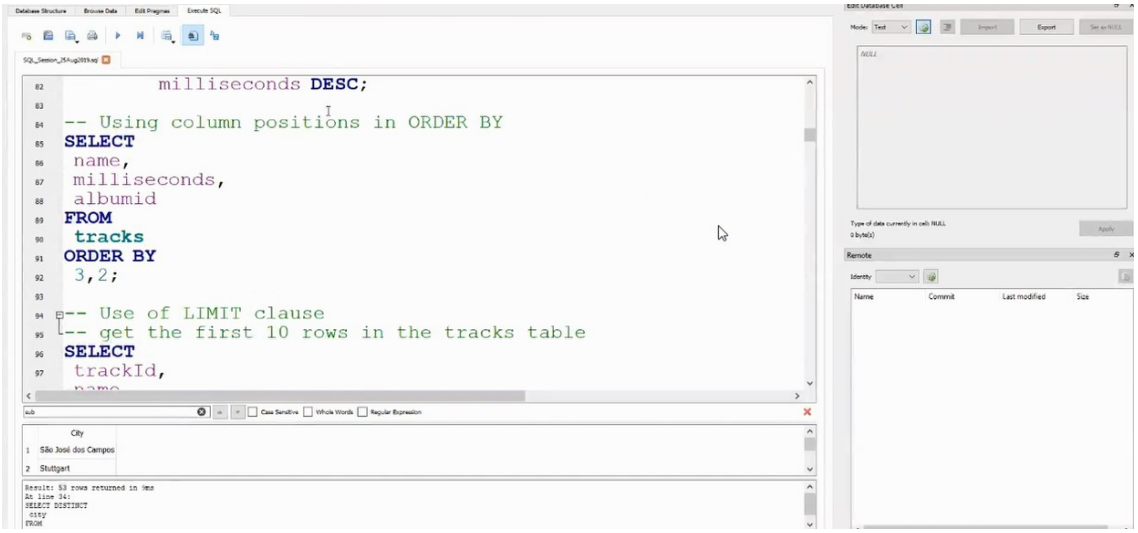
Orderby will sort the data



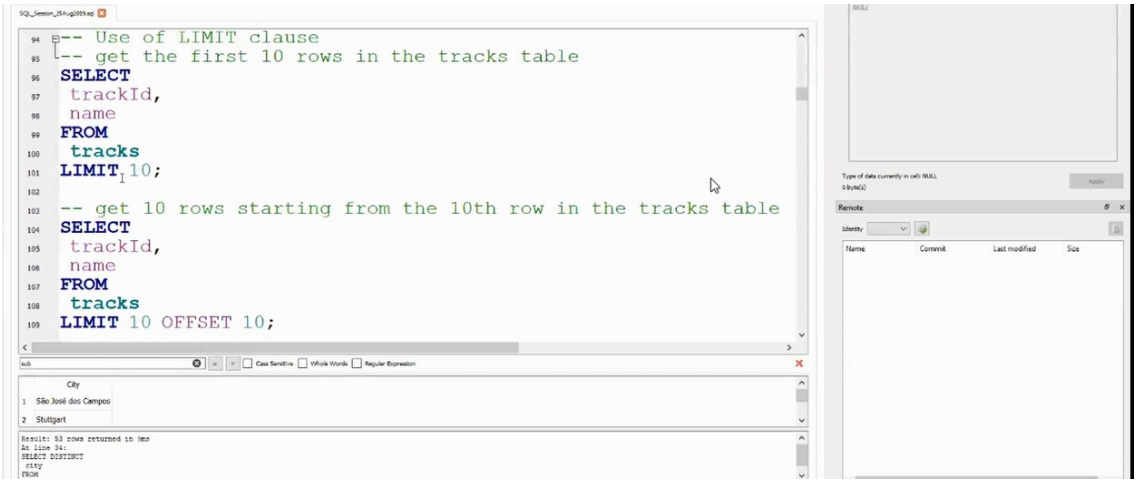
ASC : Ascending ; DESC : Descending



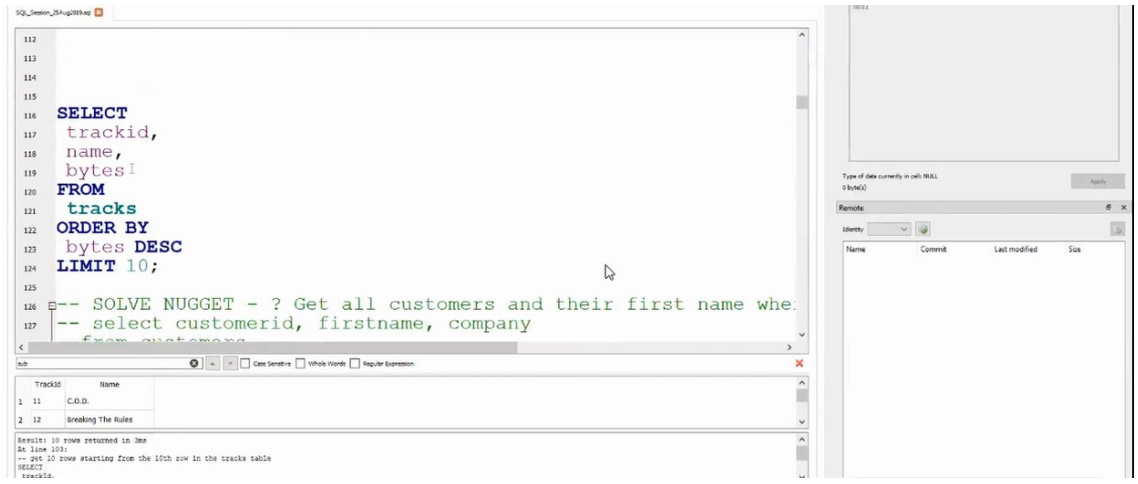
Using column positions to sort



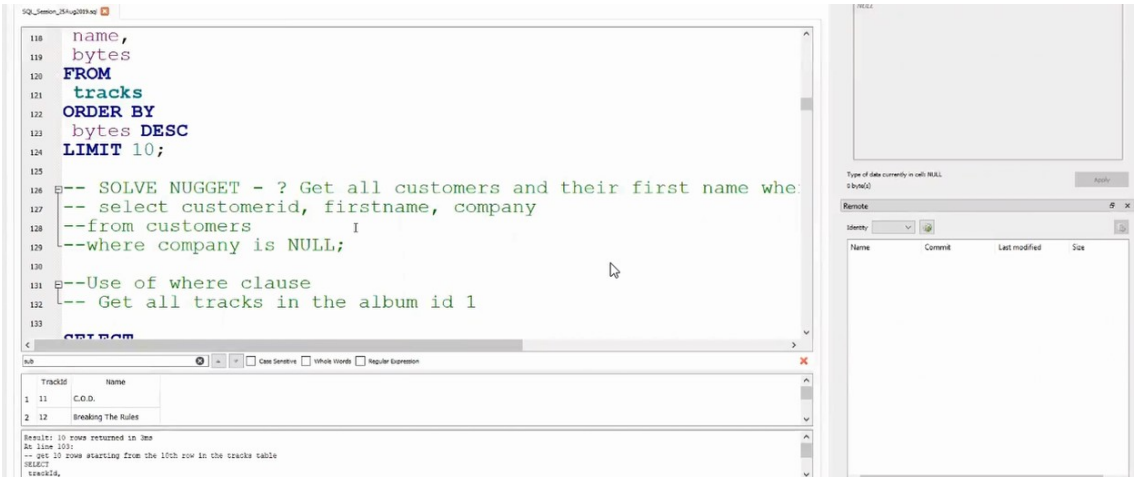
Limit will give only given no of rows of data, offset will skip given no of rows



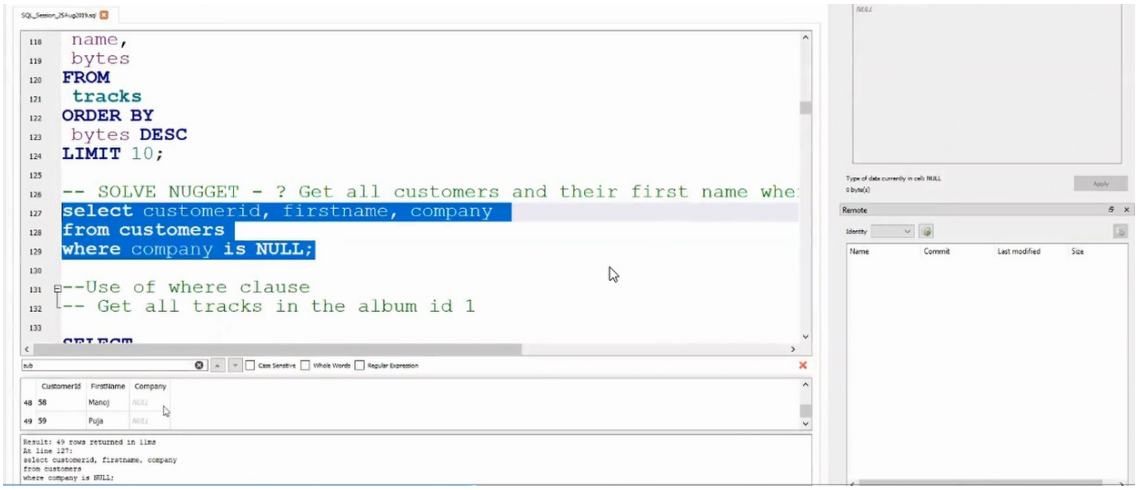
sort and limit on single piece of code

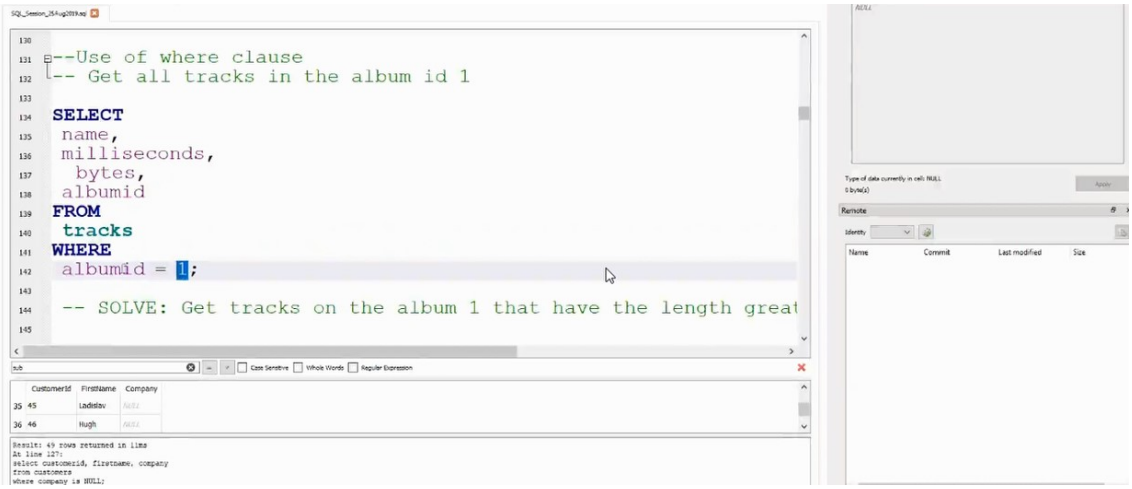


Select data where company is null (company name is absent)

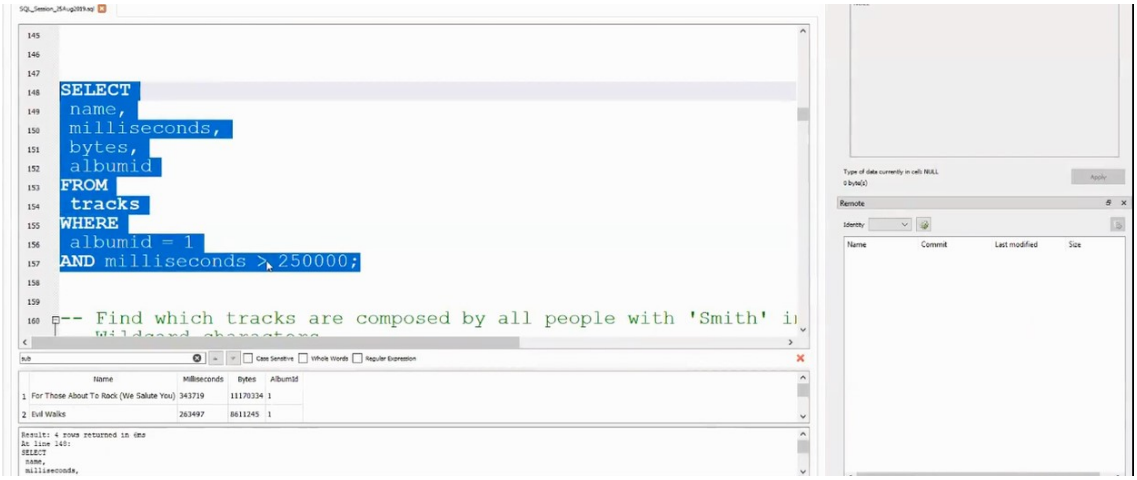


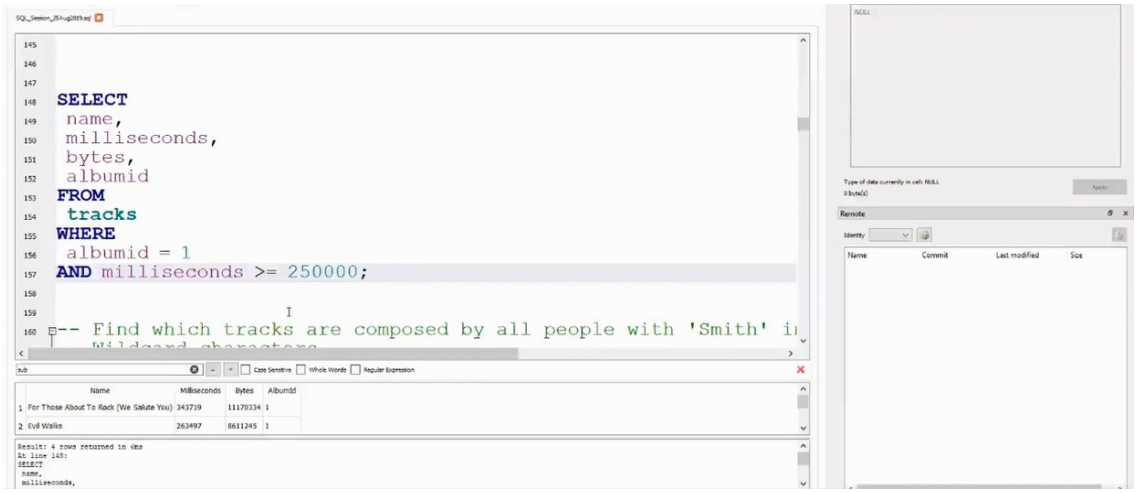
Where statement is used in this piece of code



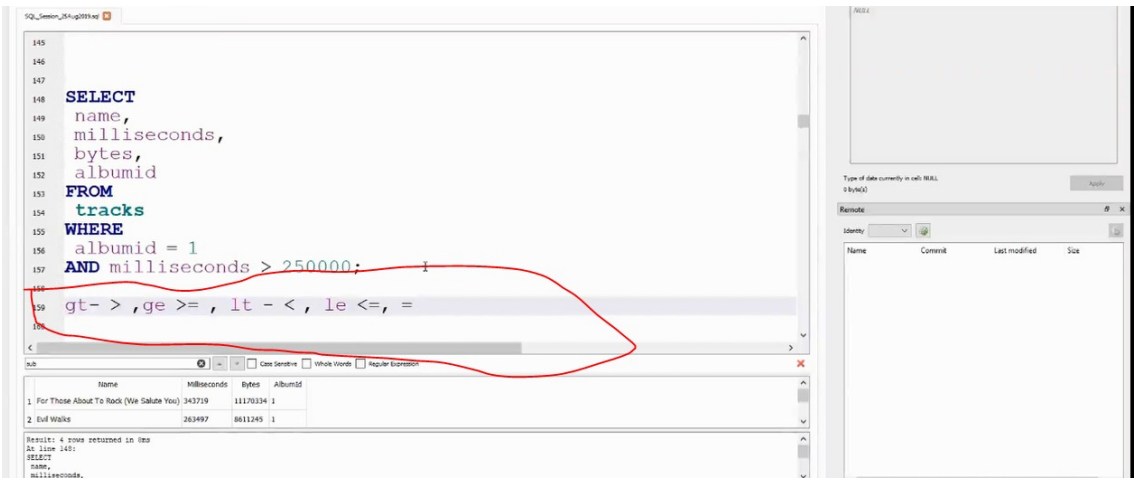


And will apply to conditions inside where statement





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

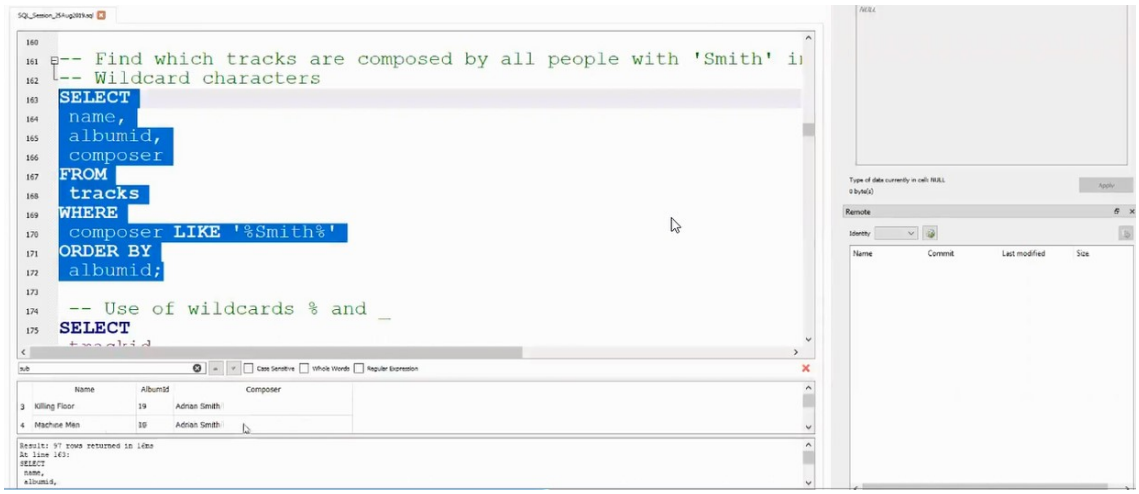


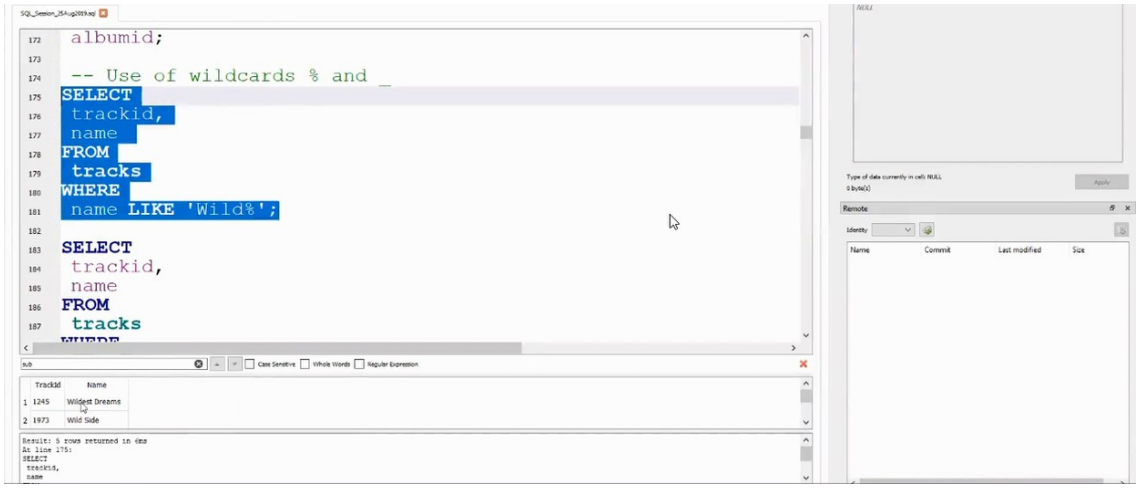
To select data rows which contains part of the string (% signs can be used)

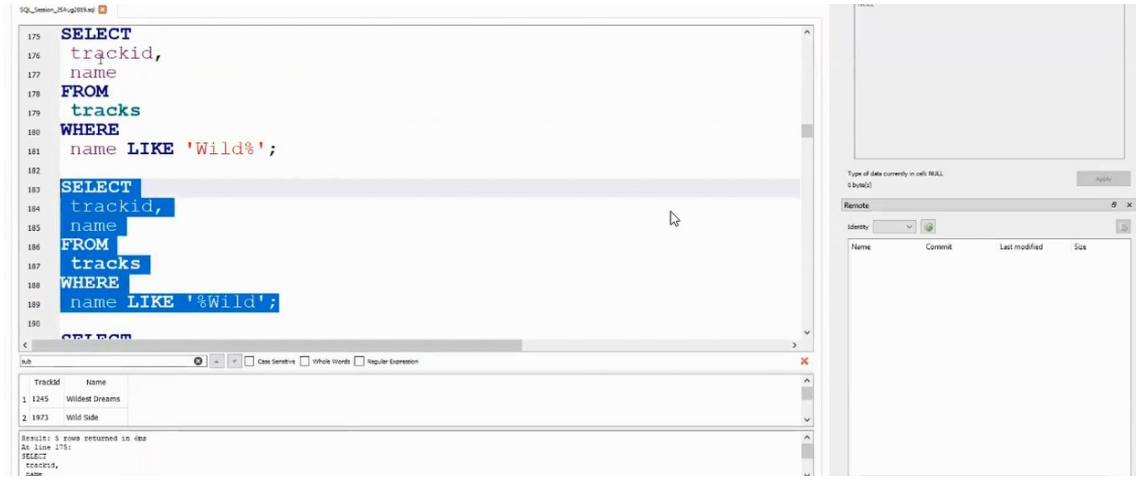
% before word – another words/letters are present before this word

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

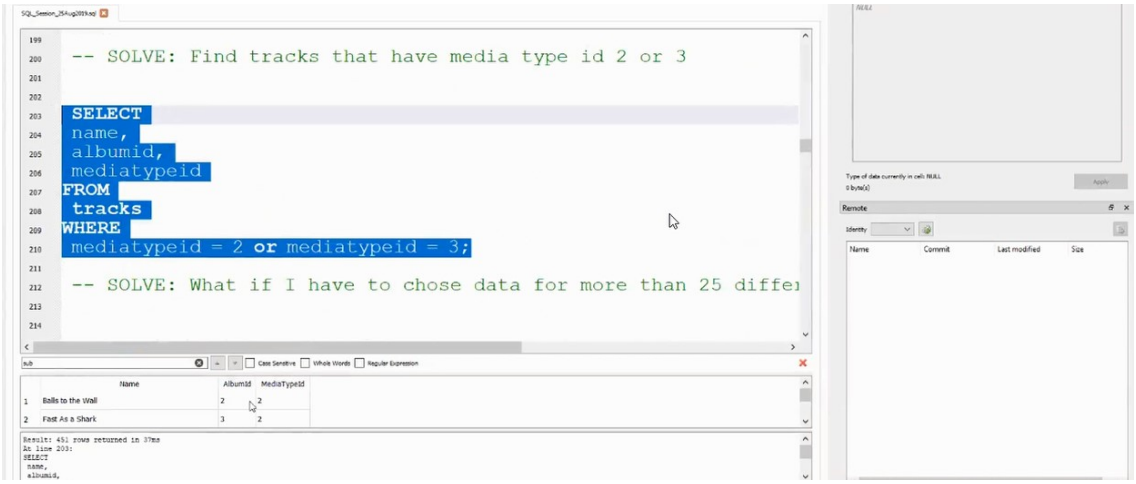
% on both side – another words/letters are present before & after this word



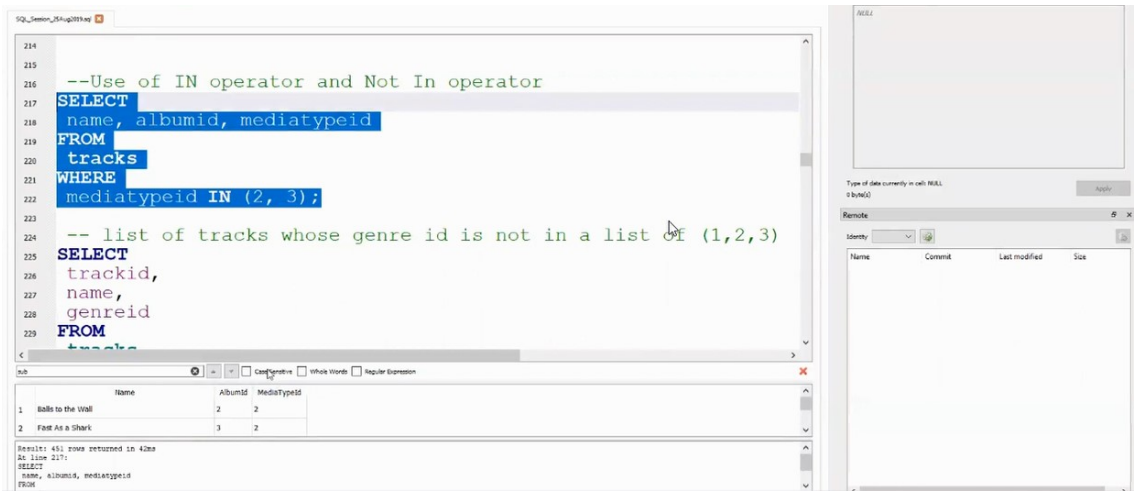




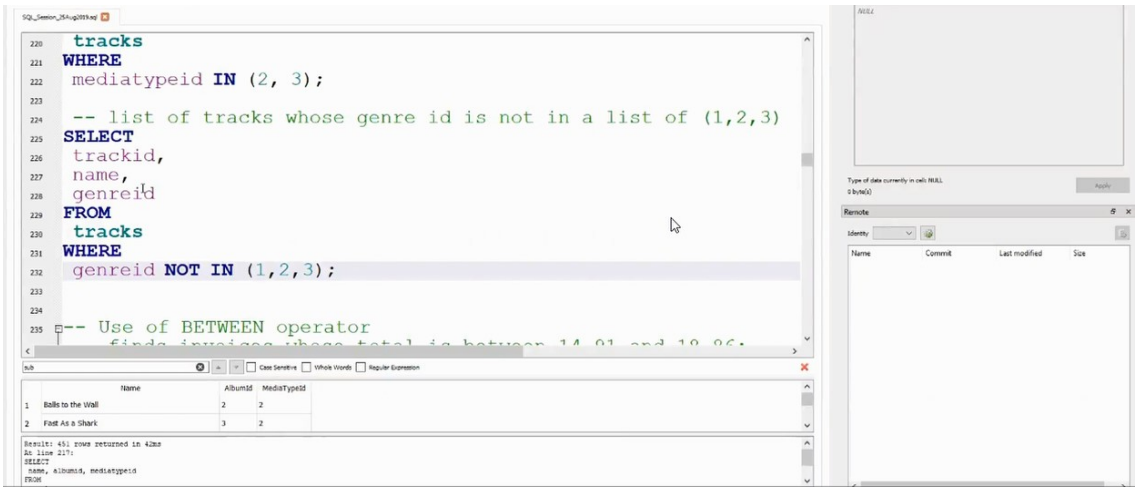
Or condition inside where



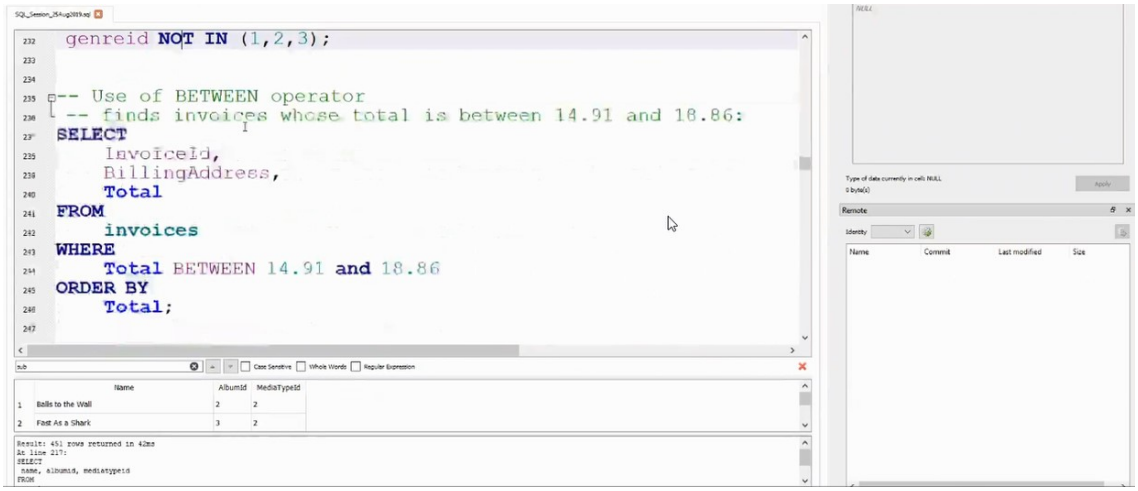
In condition – alternate for or condition (if applied on single column)



Notin condition – opposite of in

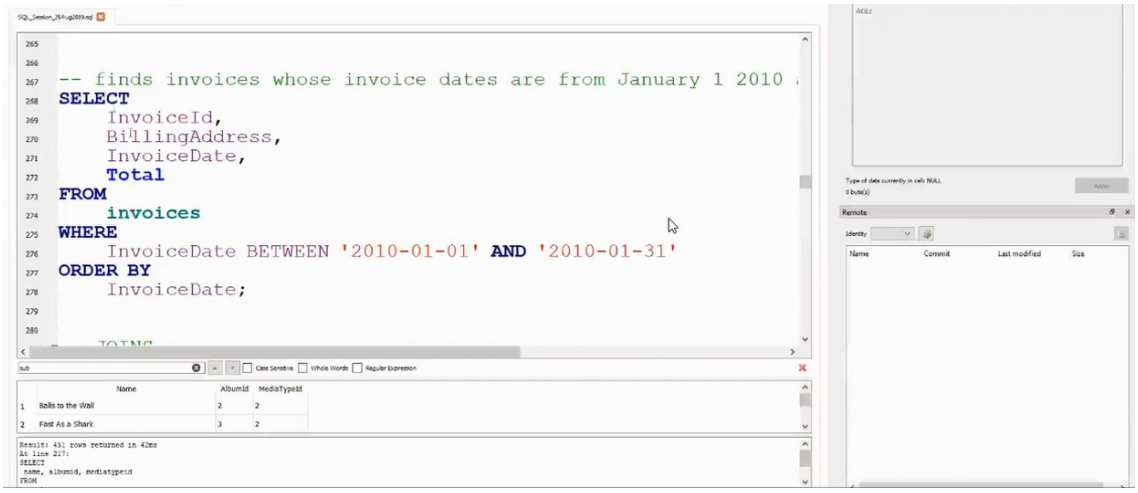


Between condition

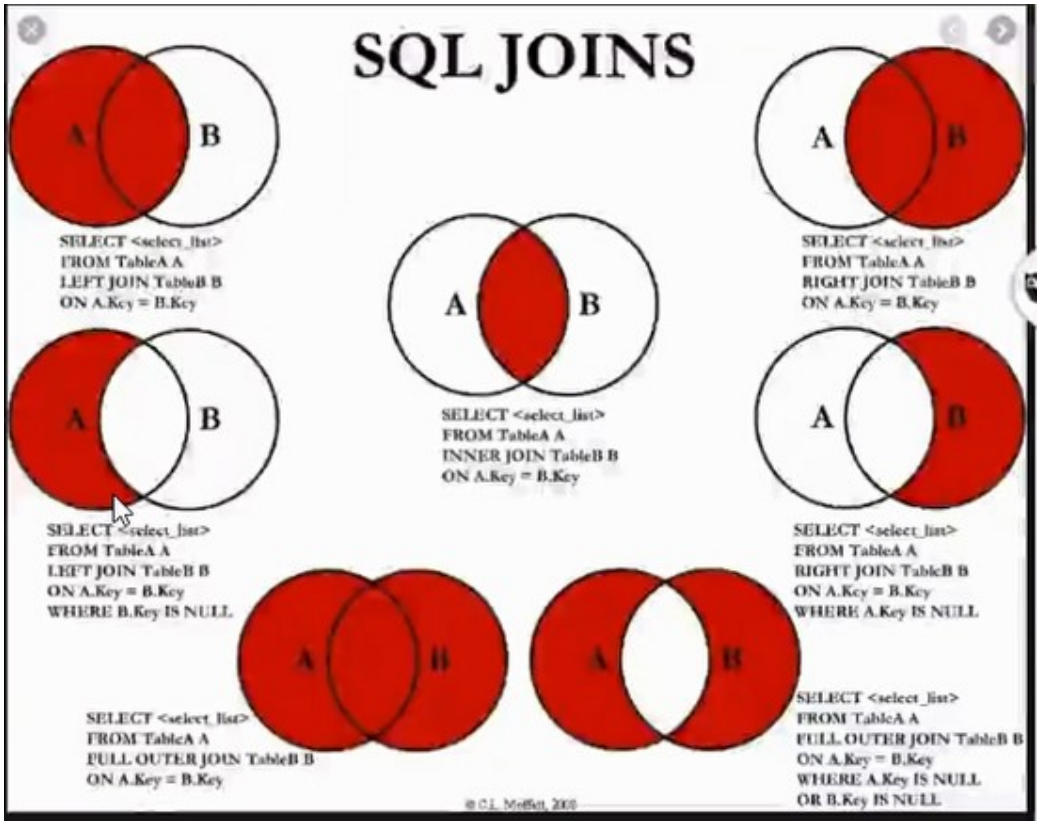


notbetween condition

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

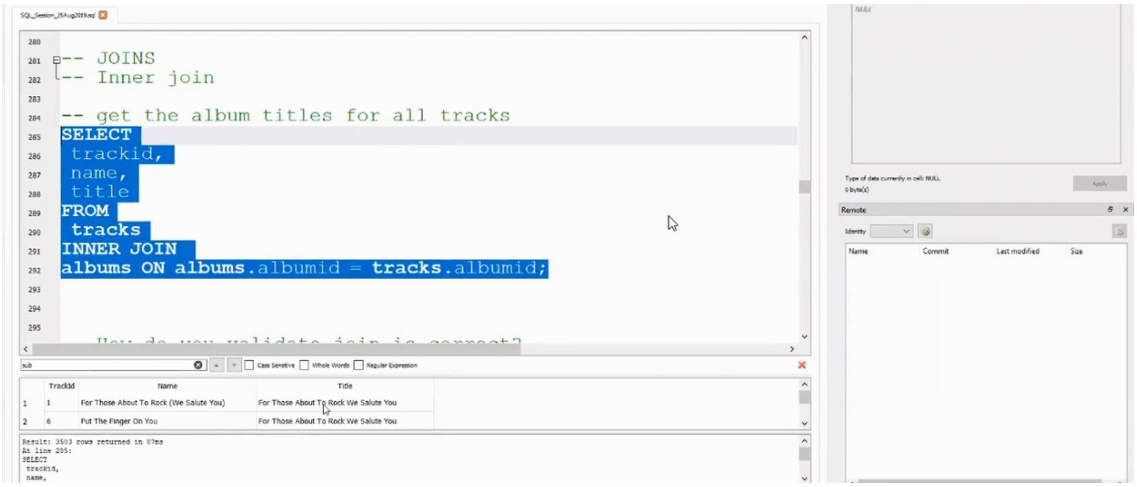
Full Outer Join

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

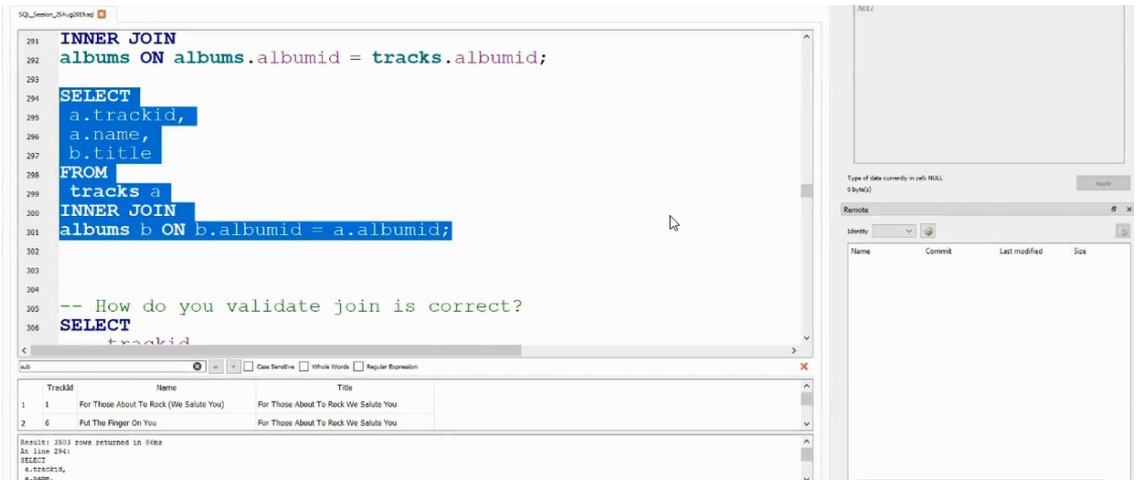
Inner join on tracks and album

Trackid and name from tracks table

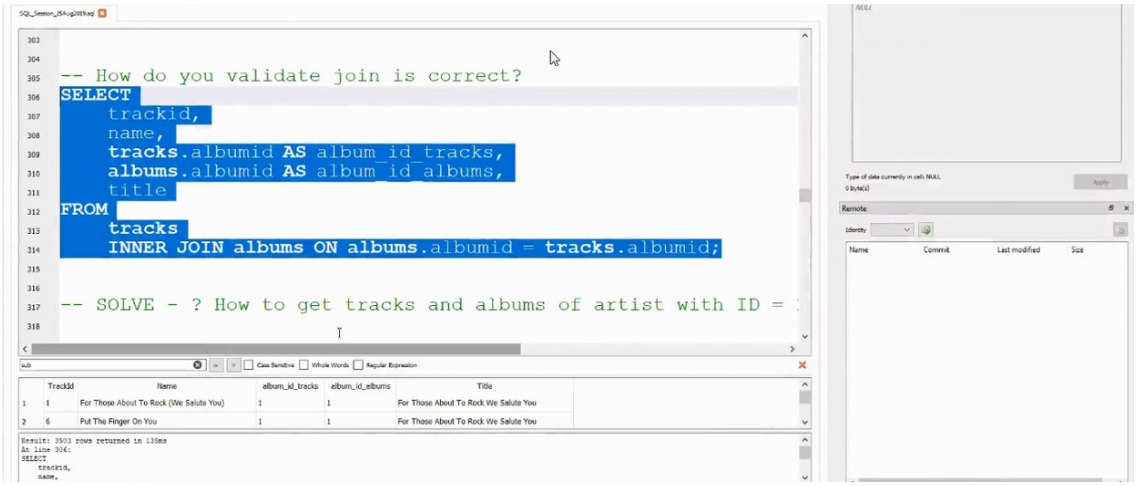
Title from album table



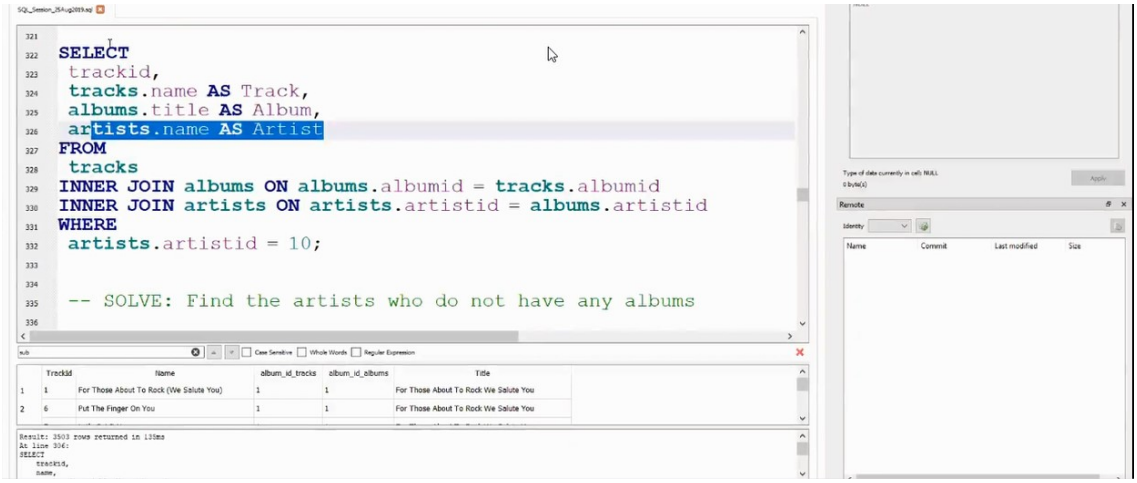
Usage of alias for table names

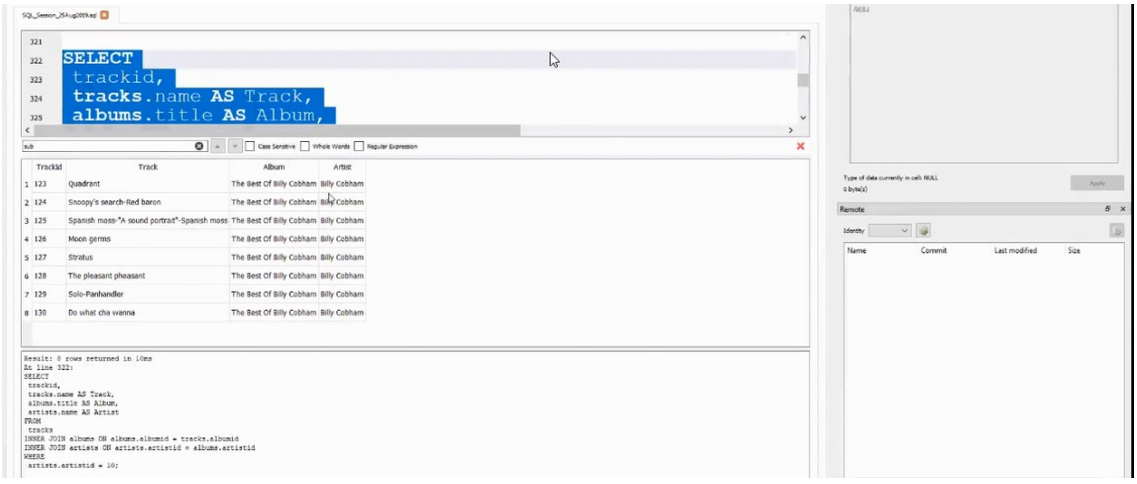


Getting albumids as well from both tables (to validate the join)

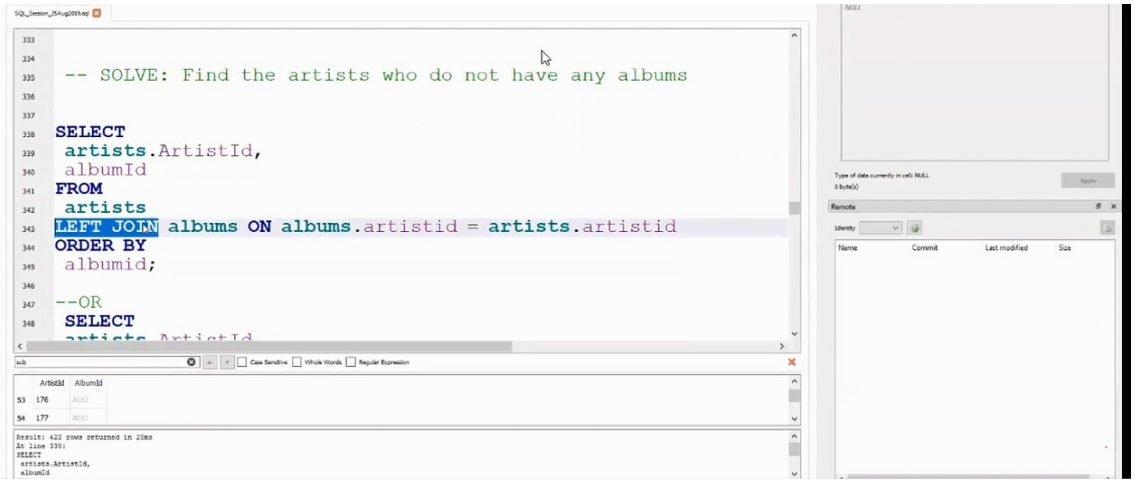


Joining multiple tables





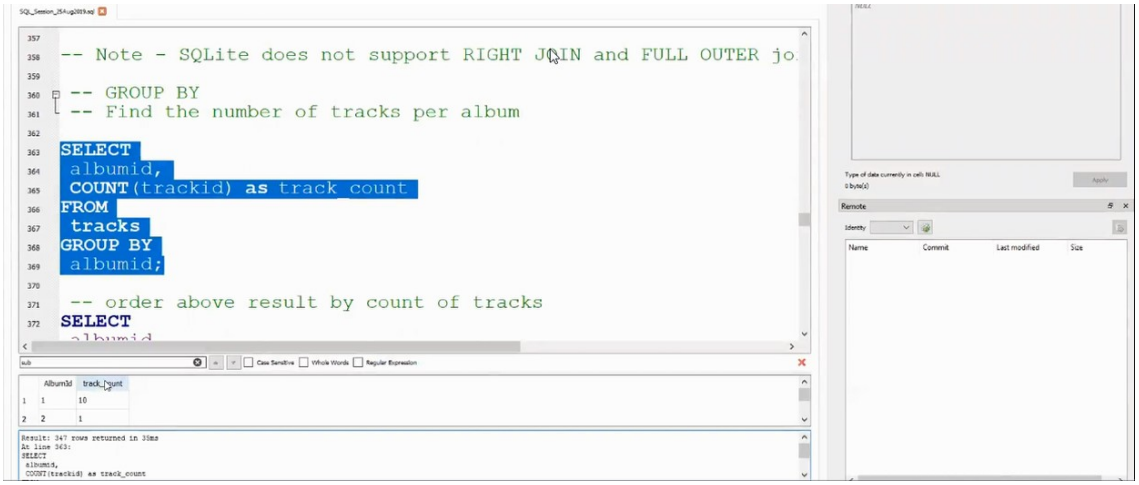
```
-- SOLVE: Find the artists who do not have any albums
```



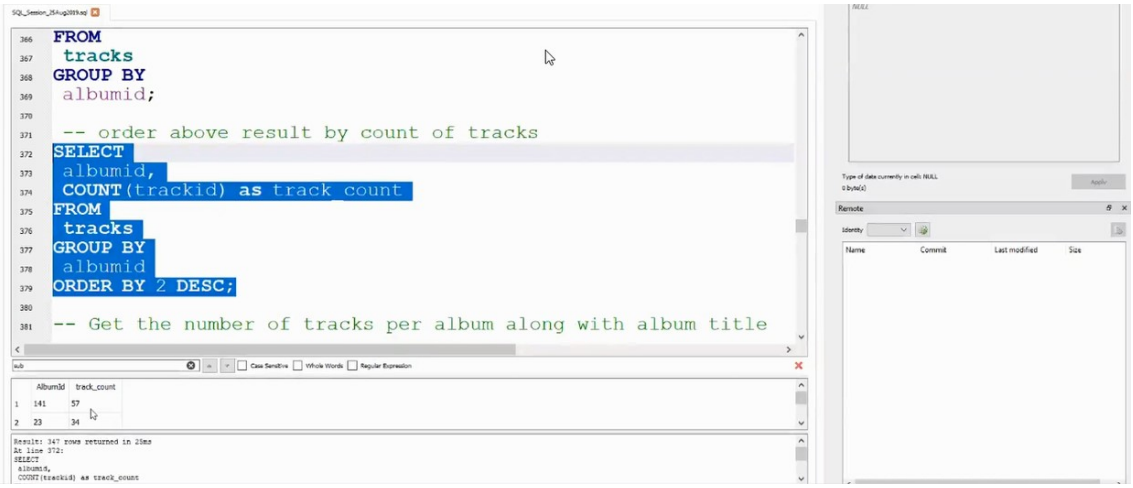
Alternate way

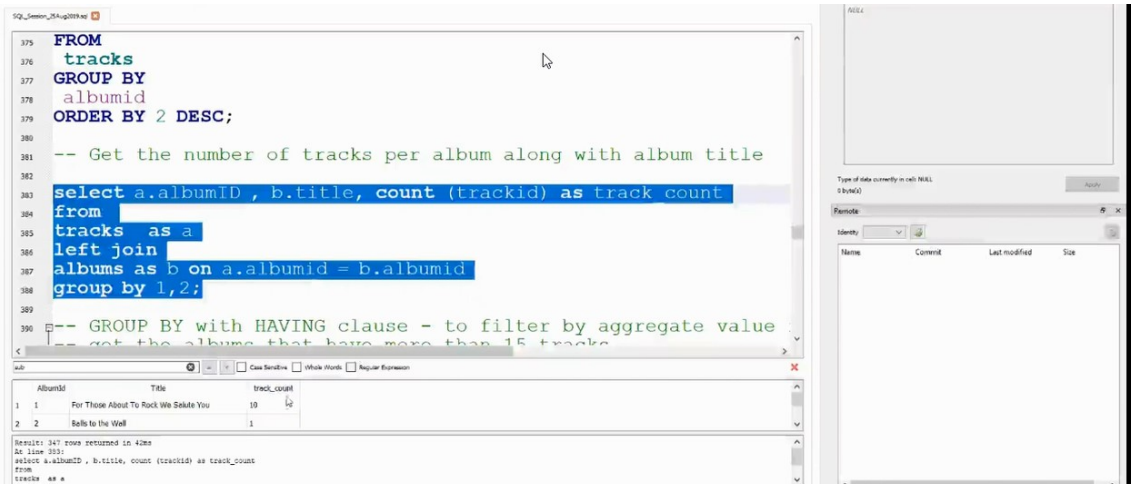

```
-- Note - SQLite does not support RIGHT JOIN and FULL OUTER jo
```

Groupby (like pivot table in excel)

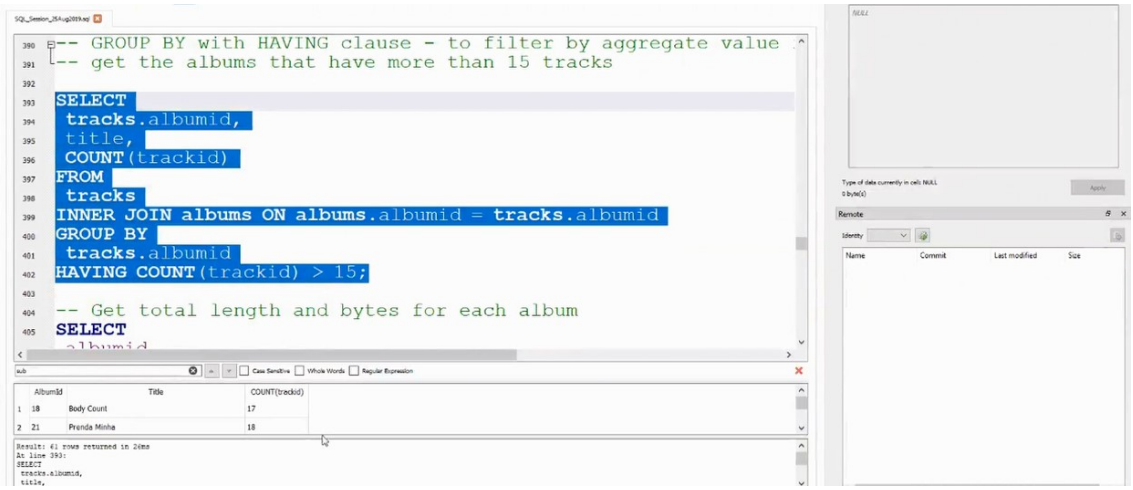


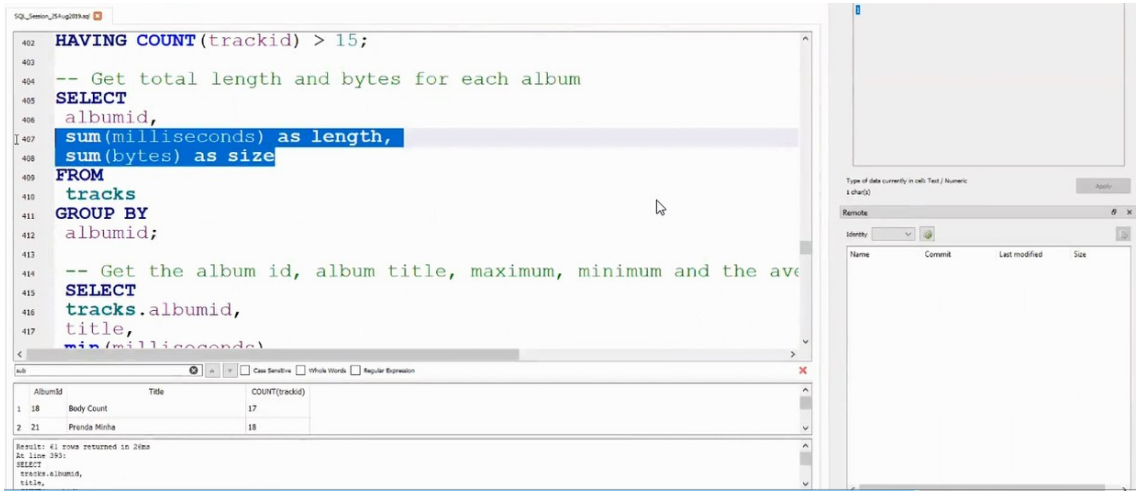
Sorting on top of groupby





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

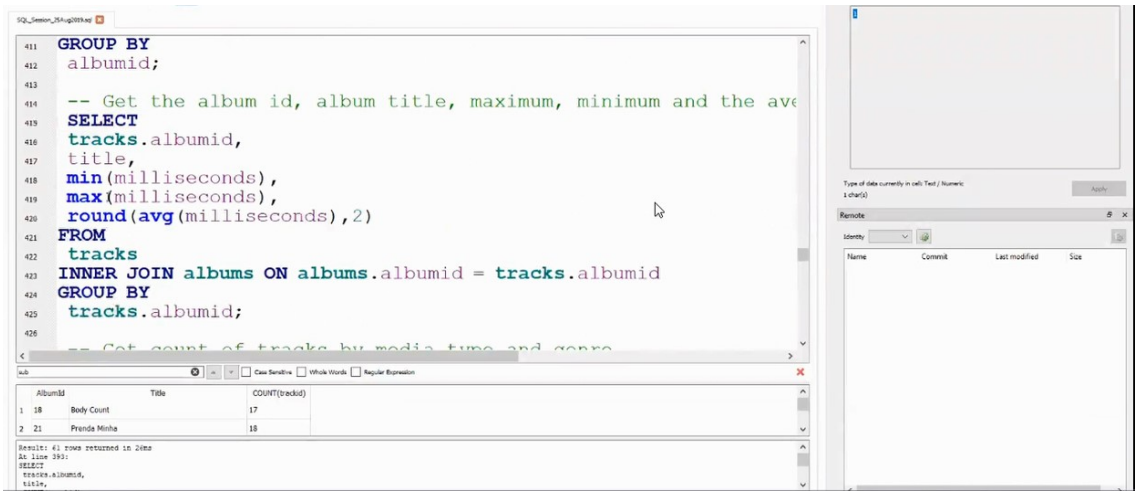




```
HAVING COUNT(trackid) > 15;

-- 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,
 min(milliseconds)
```

	AlbumId	Title	COUNT(trackid)
1	18	Body Count	17
2	21	Prenda Minha	18

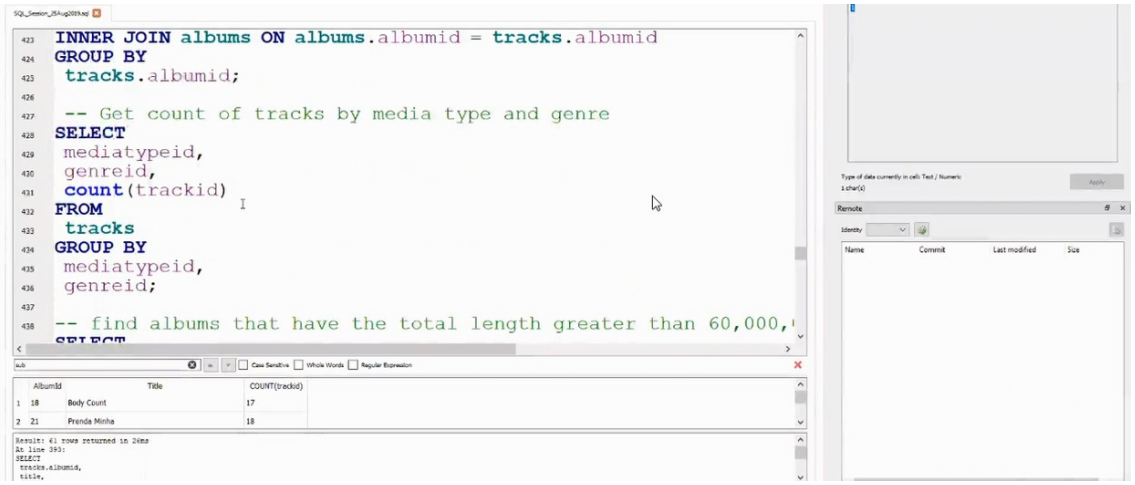
```
Result: 41 rows returned in 24ms
At line 393:
SELECT
 tracks.albumid,
 title,
```



```
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;

 -- Get count of tracks by media type and genre
```

	AlbumId	Title	COUNT(trackid)
1	18	Body Count	17
2	21	Prenda Minha	18

```
Result: 41 rows returned in 24ms
At line 393:
SELECT
 tracks.albumid,
 title,
```

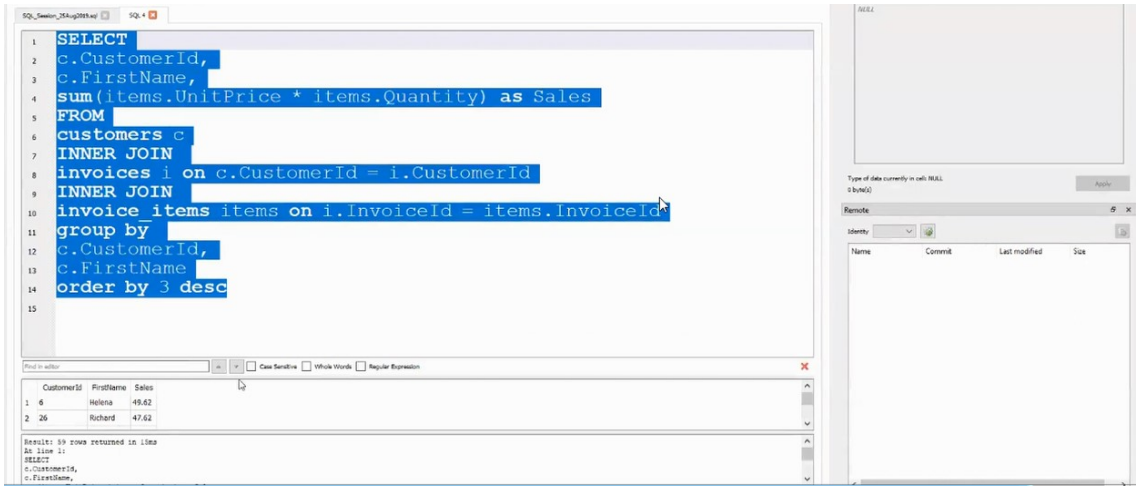


```
INNER JOIN albums ON albums.albumid = tracks.albumid
GROUP BY
 tracks.albumid;

 -- 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 24ms
At line 393:
SELECT
 tracks.albumid,
 title,
```

Top 5 customers by sales?

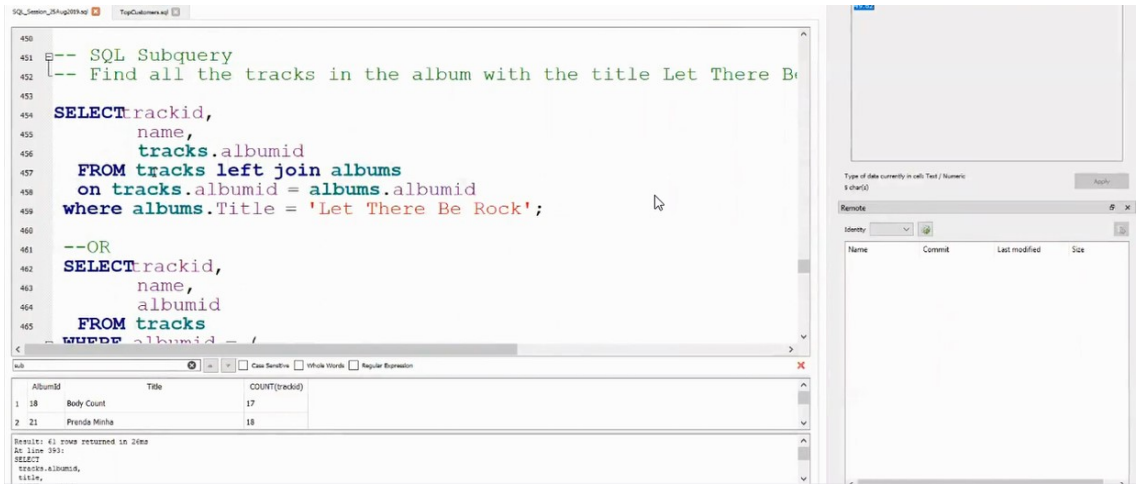


```
SELECT
c.CustomerId,
c.FirstName,
sum(items.UnitPrice * items.Quantity) as Sales
FROM
customers c
INNER JOIN
invoices i on c.CustomerId = i.CustomerId
INNER JOIN
invoice_items items on i.InvoiceId = items.InvoiceId
group by
c.CustomerId,
c.FirstName
order by 3 desc
```

	CustomerId	FirstName	Sales
1	6	Helena	49.62
2	26	Richard	47.62

```
Result: 59 rows returned in 18ms
At line 1:
SELECT
c.CustomerId,
c.FirstName,
```

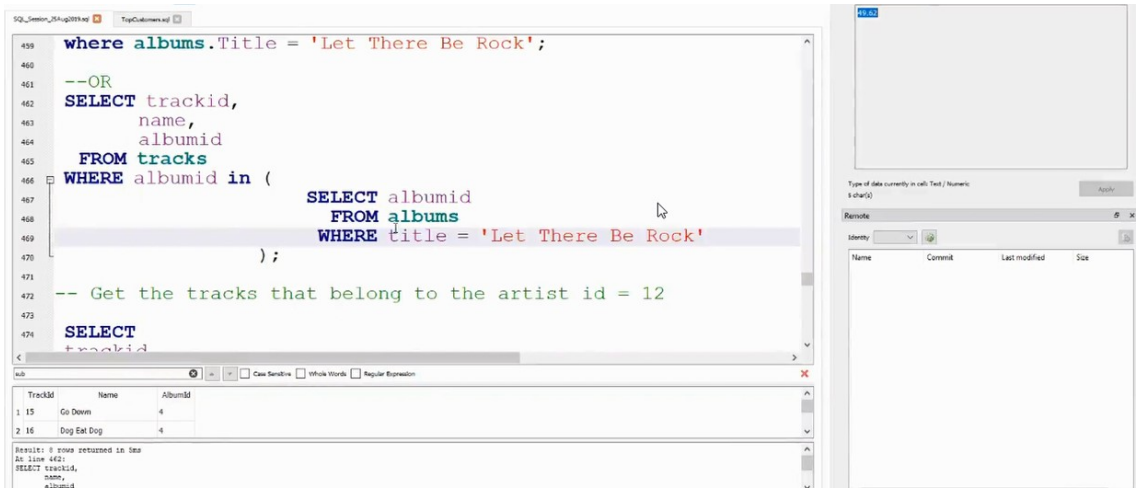
SQL Subqueries



```
-- SQL Subquery
-- Find all the tracks in the album with the title Let There B

SELECT trackid,
       name,
       tracks.albumid
  FROM tracks left join albums
  on tracks.albumid = albums.albumid
 where albums.Title = 'Let There Be Rock';

 --OR
 SELECT trackid,
       name,
       albumid
  FROM tracks
 WHERE albumid = (
```

	AlbumId	Title	COUNT(trackid)
1	18	Body Count	17
2	21	Prenda Minha	18

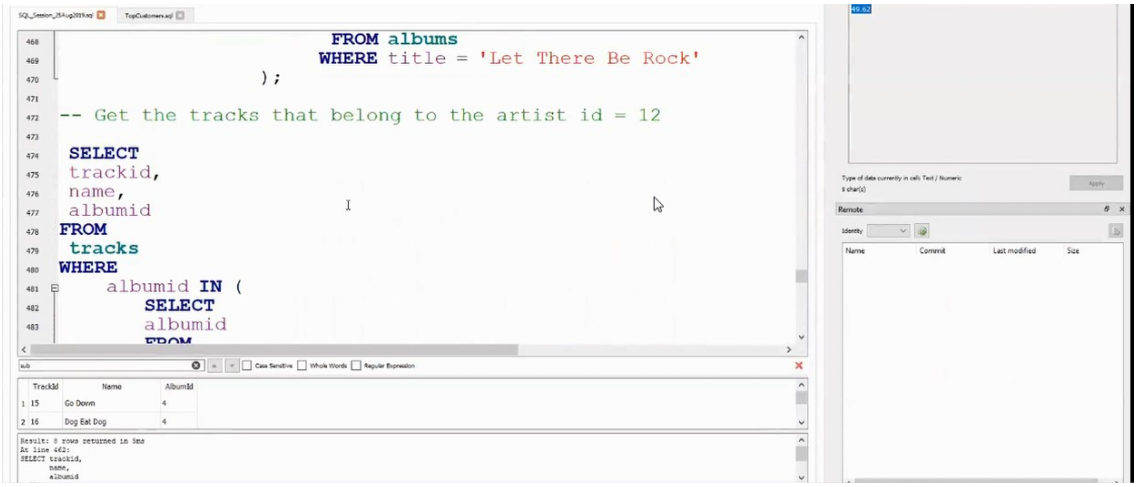
```
Result: 41 rows returned in 24ms
At line 393:
SELECT
 tracks.albumid,
 title,
```



```
where albums.Title = 'Let There Be Rock';

 --OR
 SELECT trackid,
       name,
       albumid
  FROM tracks
 WHERE albumid in (
                    SELECT albumid
                      FROM albums
                     WHERE title = 'Let There Be Rock'
                  );

-- Get the tracks that belong to the artist id = 12

 SELECT
 trackid
```

	TrackId	Name	AlbumId
1	15	Go Down	4
2	16	Dog Eat Dog	4

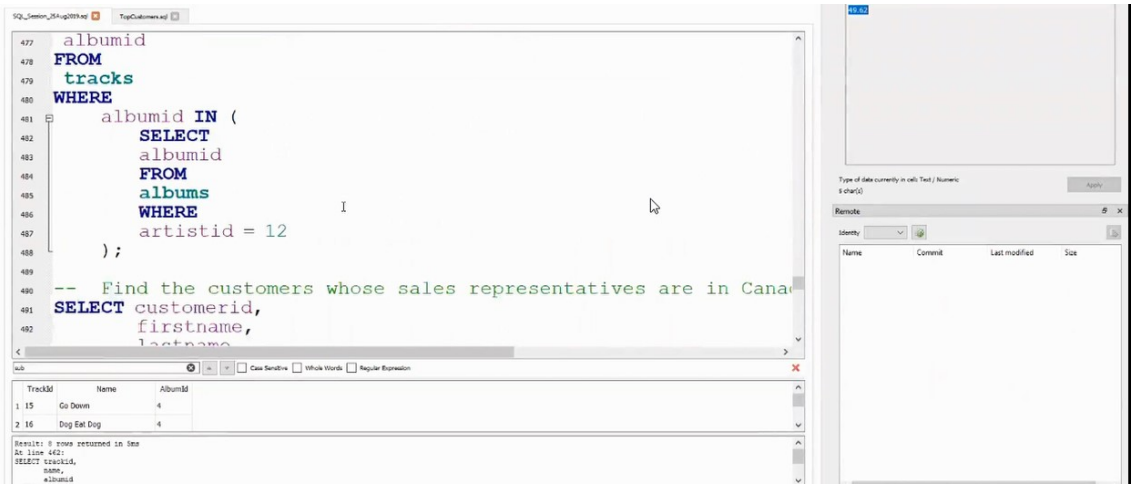
```
Result: 8 rows returned in 8ms
At line 462:
SELECT trackid,
       name,
       albumid
```



```
                        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

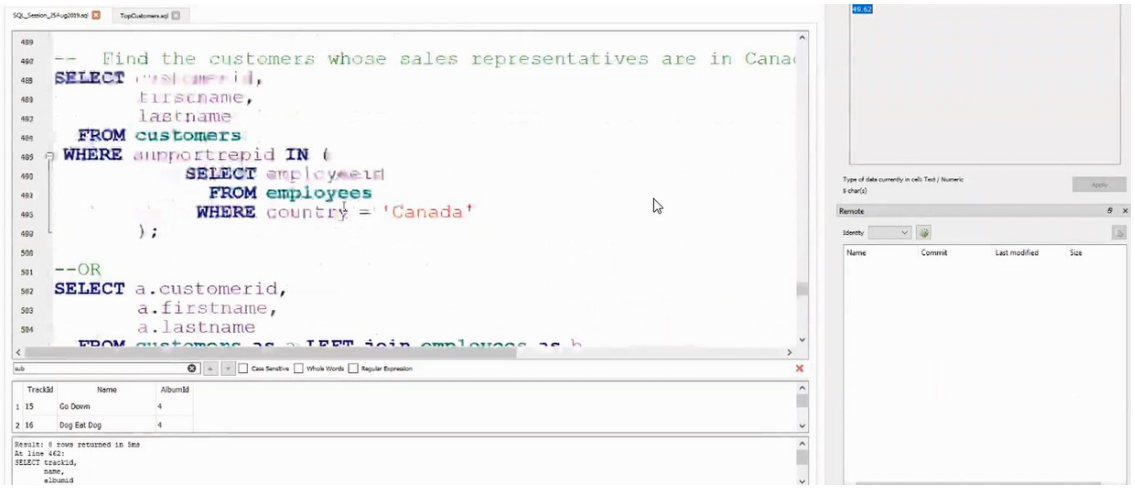
```
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
    name,
    albumid
```



```
 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

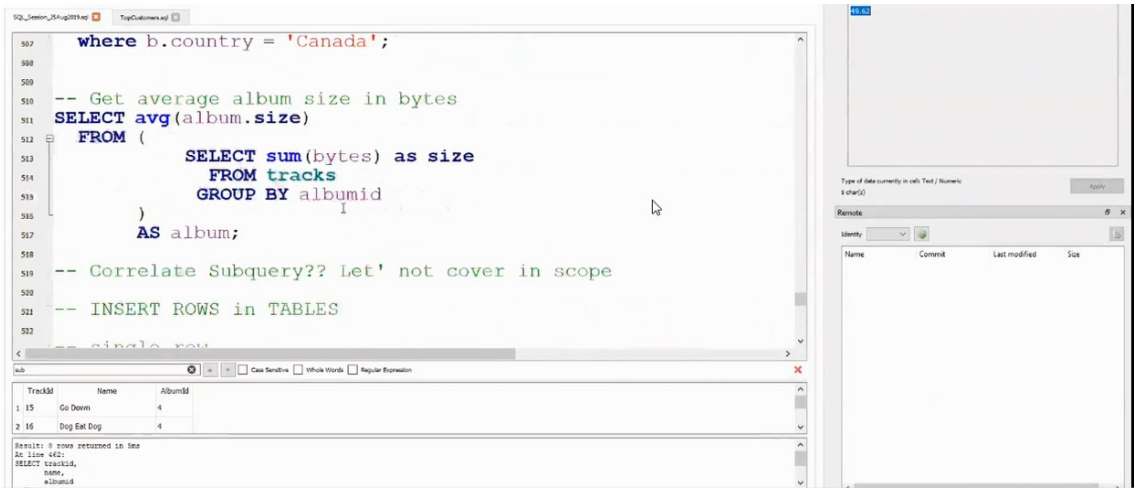
```
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
    name,
    albumid
```



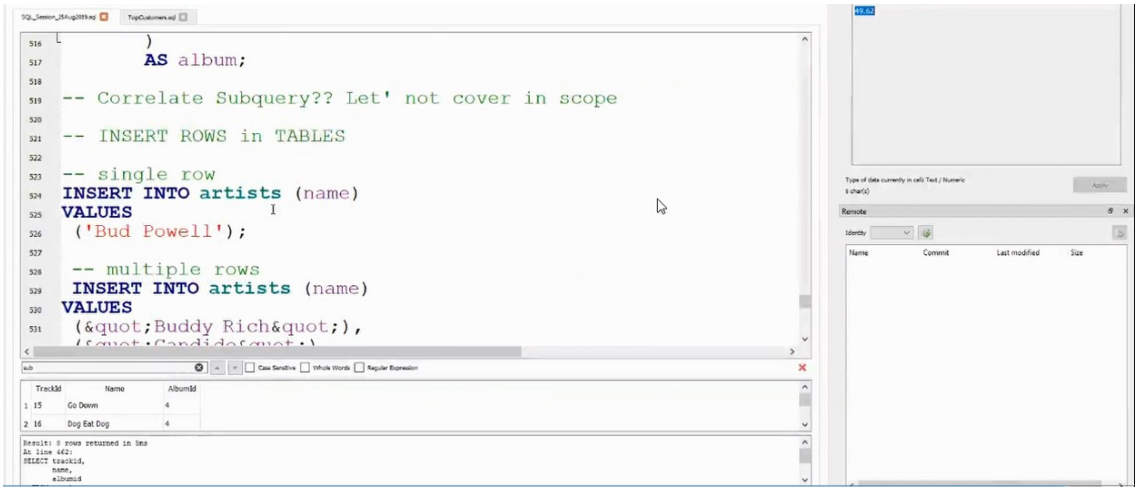
```
--  Find the customers whose sales representatives are in Cana
SELECT customerid,
        firstname,
        lastname
  FROM customers
 WHERE supportrepid IN (
            SELECT employeeid
              FROM employees
             WHERE country = 'Canada'
        );

--OR
SELECT a.customerid,
        a.firstname,
        a.lastname
```

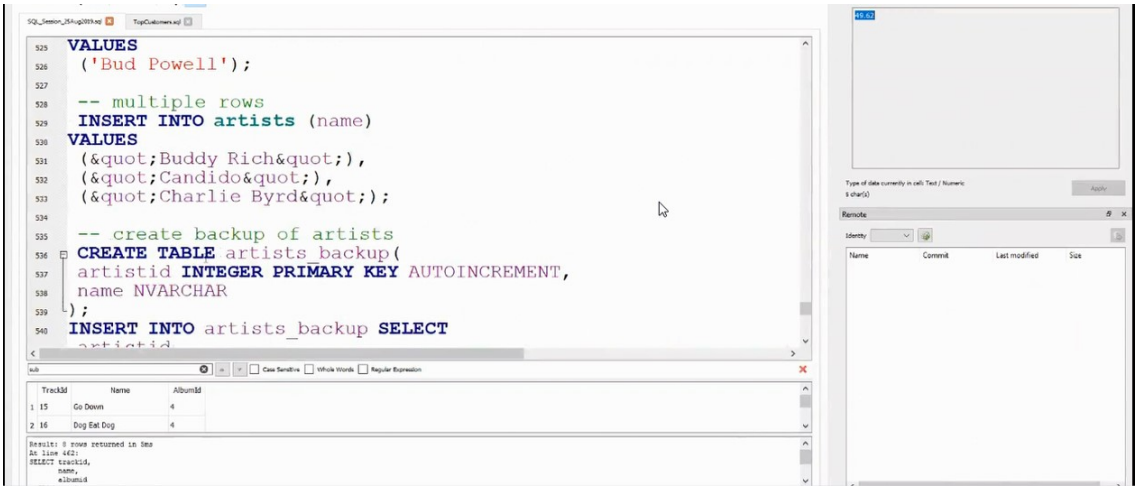
	TrackId	Name	AlbumId
1	15	Go Down	4
2	16	Dog Eat Dog	4

```
Result: 8 rows returned in 5ms
At line 462:
SELECT trackid,
    name,
    albumid
```

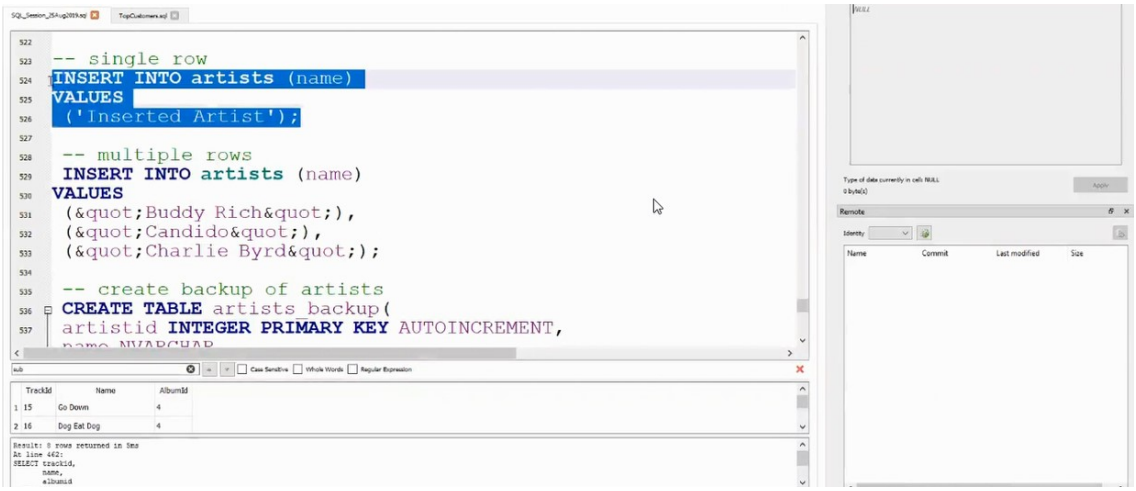


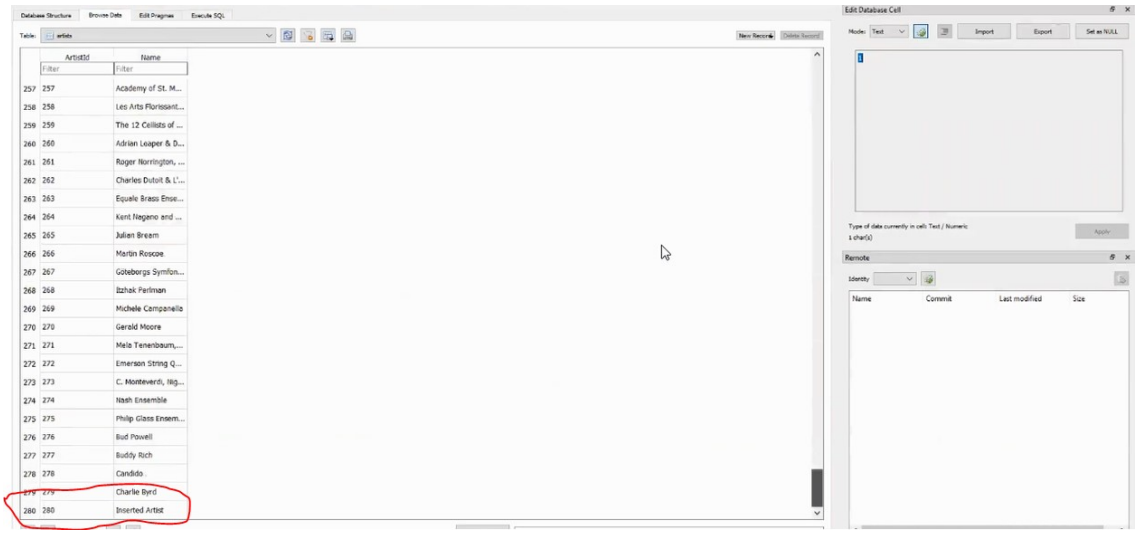
Correlate subquery



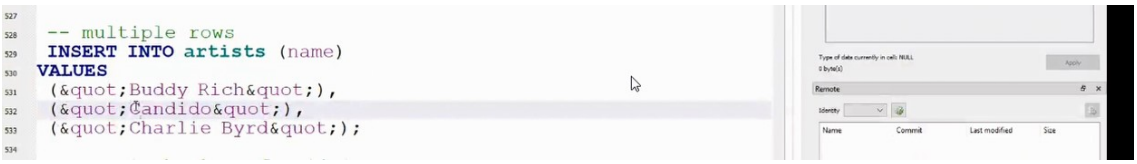


To insert values in a table





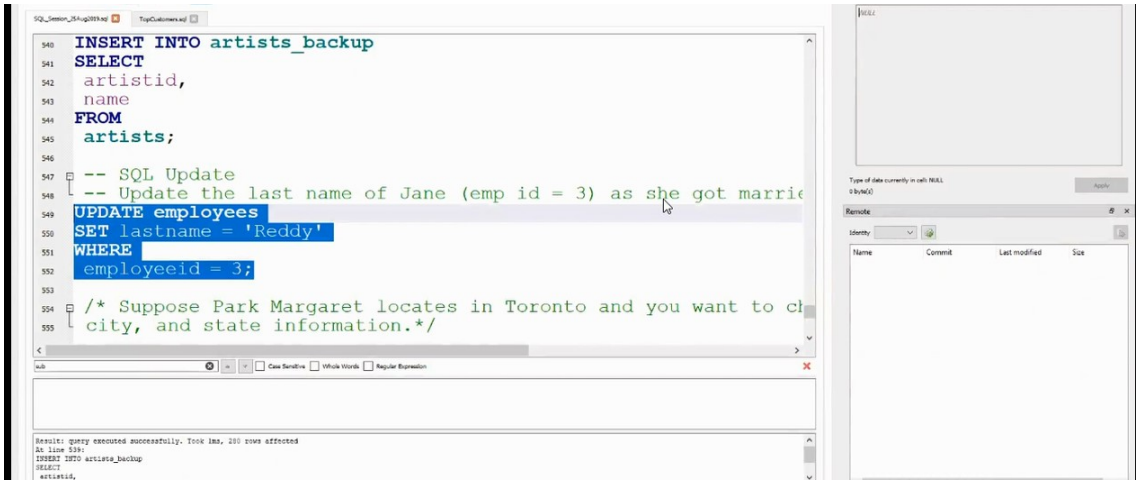
Insert multiple values in a table

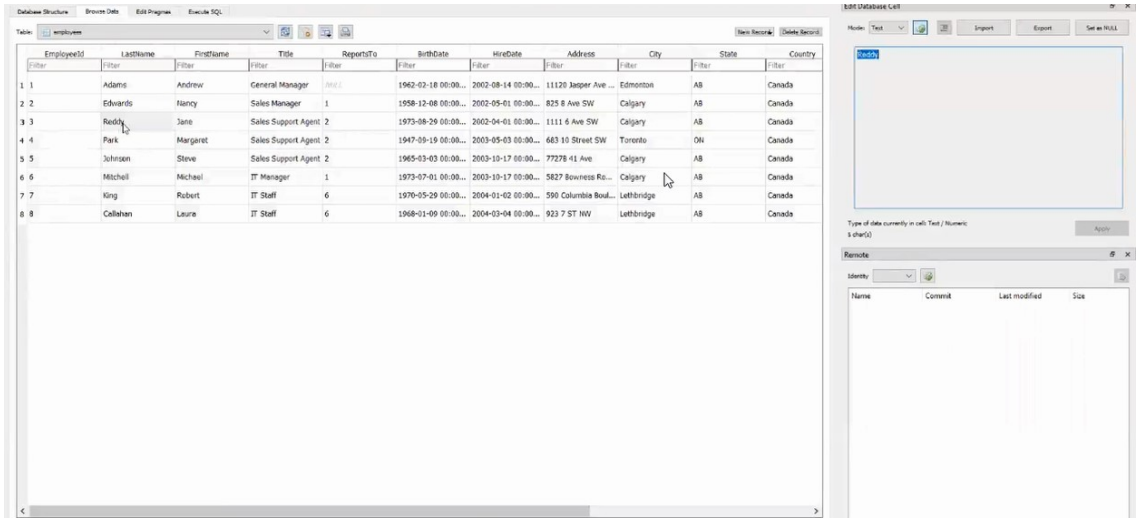


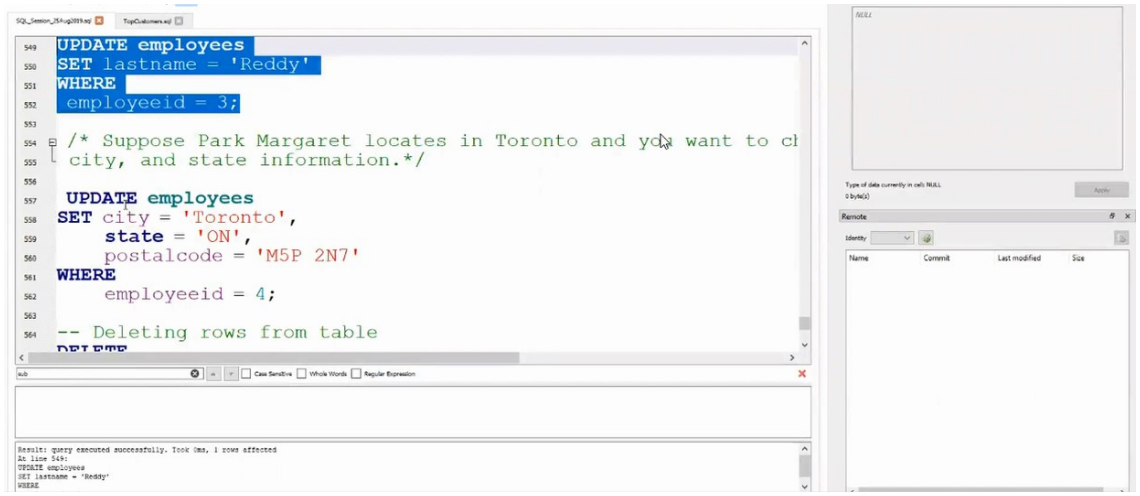
Create new table – artists backup is table created from artist table

New rows can be inserted into new table from original tables

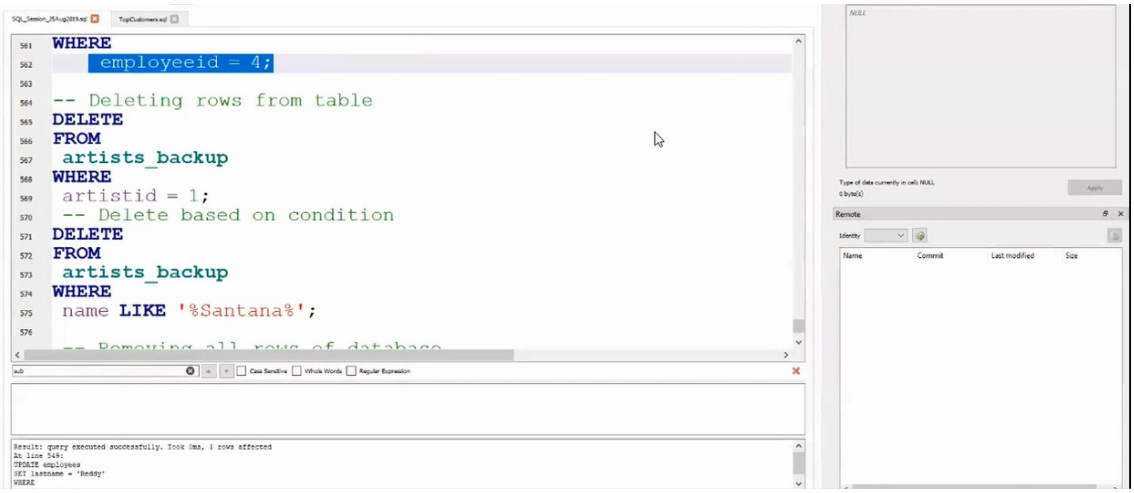
Update values in a table



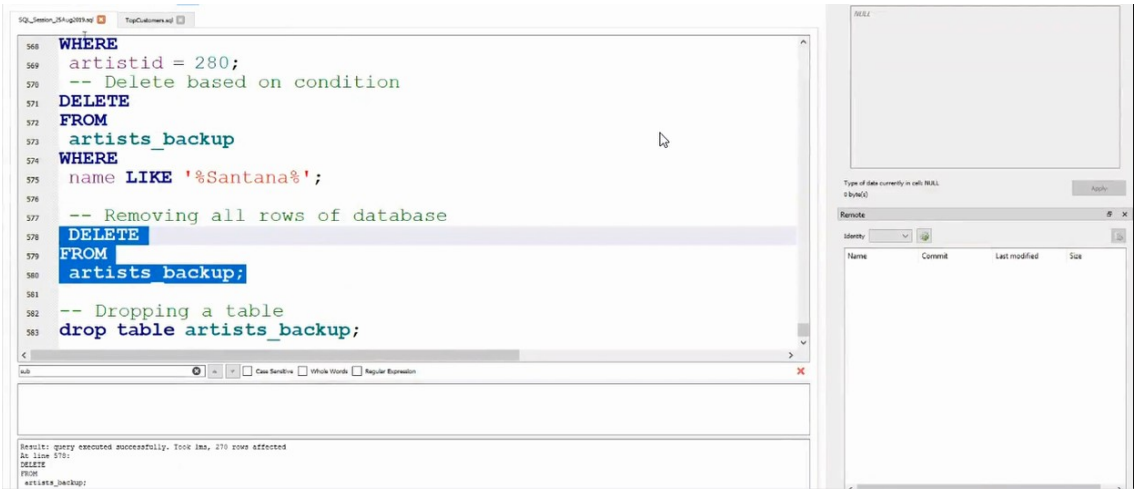




Deleting rows



Delete all the rows and dropping a table



Export table as csv