SQL

Session 3

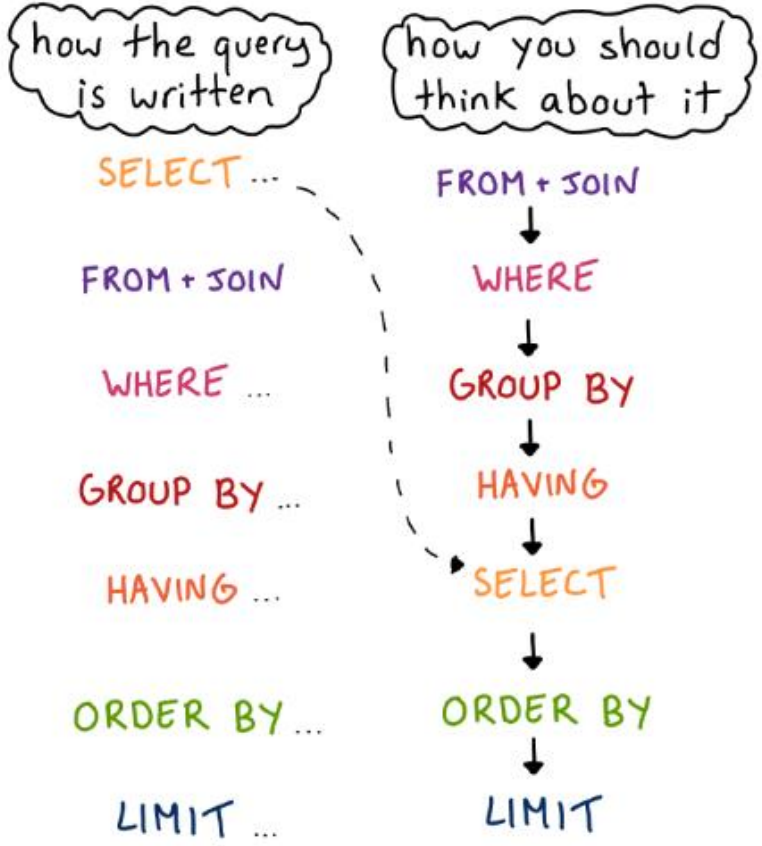
CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

- Subqueries
- DDL Command

Subqueries

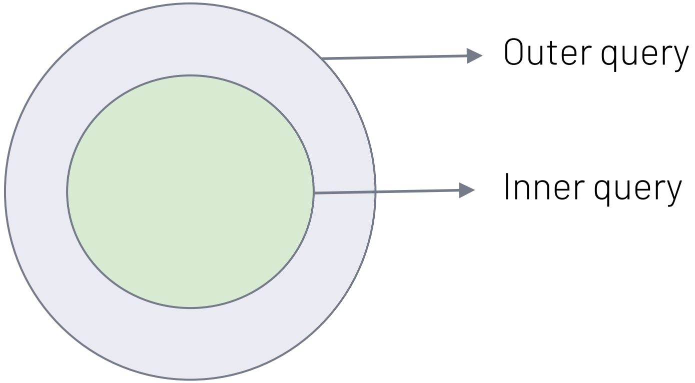
The query's steps don't happen in the order they're written:



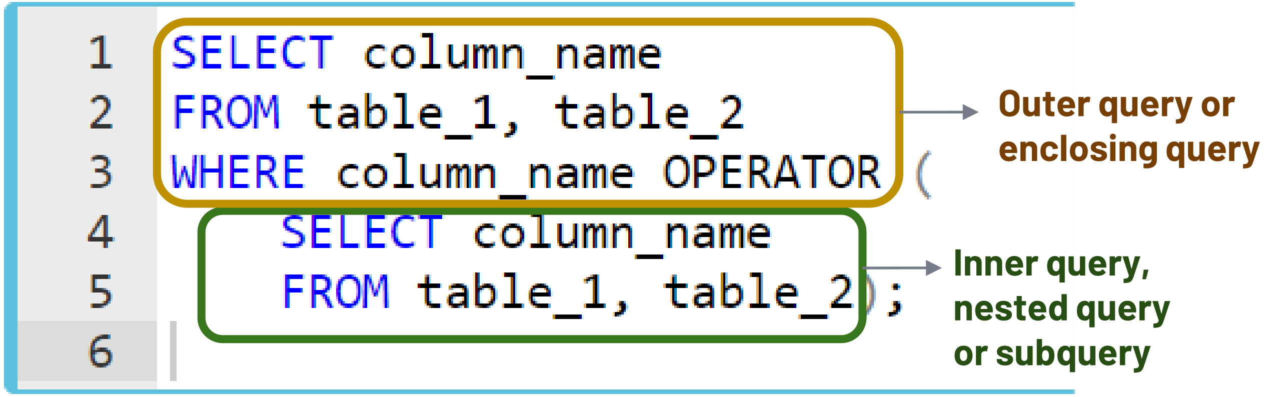
(In reality query execution is much more complicated than this. There are a lot of optimizations.)

Introduction

A subquery is a SELECT statement that is nested within another statement. The subquery is also called the inner query or nested query.



Syntax



```
SELECT column_name
FROM table_1, table_2
WHERE column_name OPERATOR (
    SELECT column_name
    FROM table_1, table_2);
```

Outer query or enclosing query

Inner query, nested query or subquery

- Subqueries are nested queries that provide data to the enclosing query.
- Subqueries can return individual values or a list of records
- Subqueries must be enclosed with parenthesis

Introduction

A subquery may be used in:

- SELECT clause
- FROM clause
- WHERE clause

Types of Subqueries

There are two main types of subqueries:

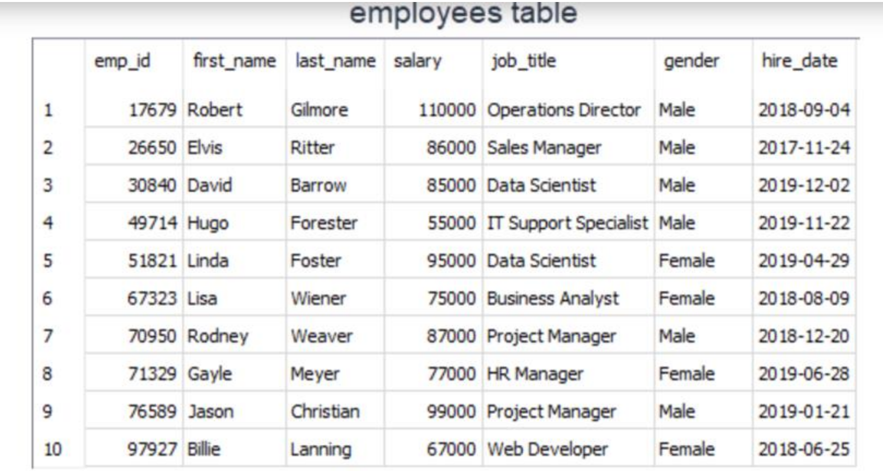
- Single-row subqueries
- Multiple-row subqueries

Single-row Subqueries

Single-row subqueries return one row with only one column and are typically used with single-row operators such as =, >, >=, <=, <>, != especially in WHERE clause.

Example

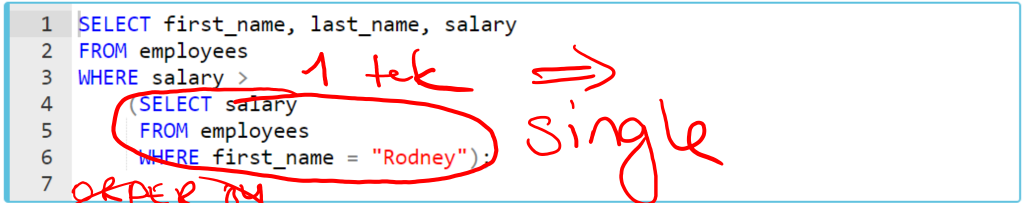
Find the employees who get paid more than Rodney Weaver



employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

query:

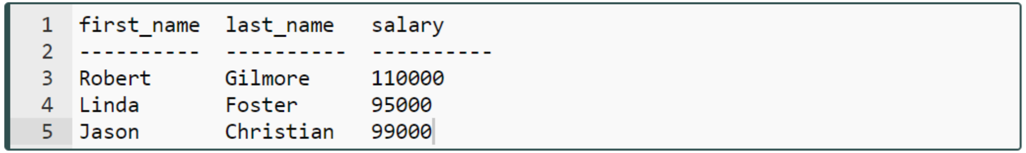


```
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (SELECT salary
    FROM employees
    WHERE first_name = "Rodney");
```

1 tek ⇒ single row

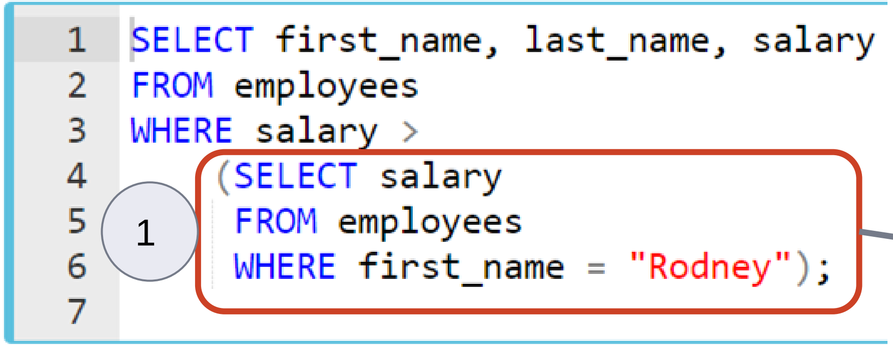
ORDER BY
limit

output:

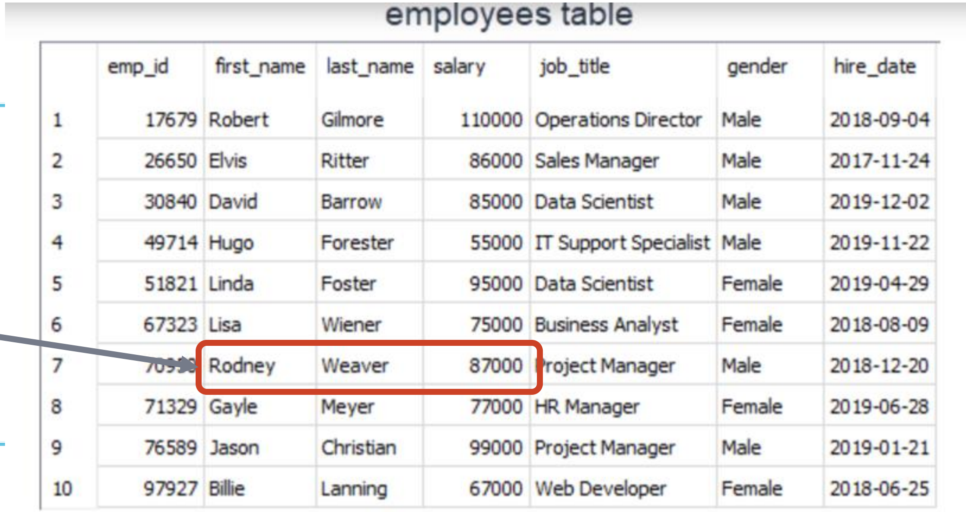


```
first_name  last_name   salary
----------  ----------  ----------
Robert      Gilmore     110000
Linda       Foster      95000
Jason       Christian   99000
```

Analyze the query-1



```
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (SELECT salary
     FROM employees
     WHERE first_name = "Rodney");
```

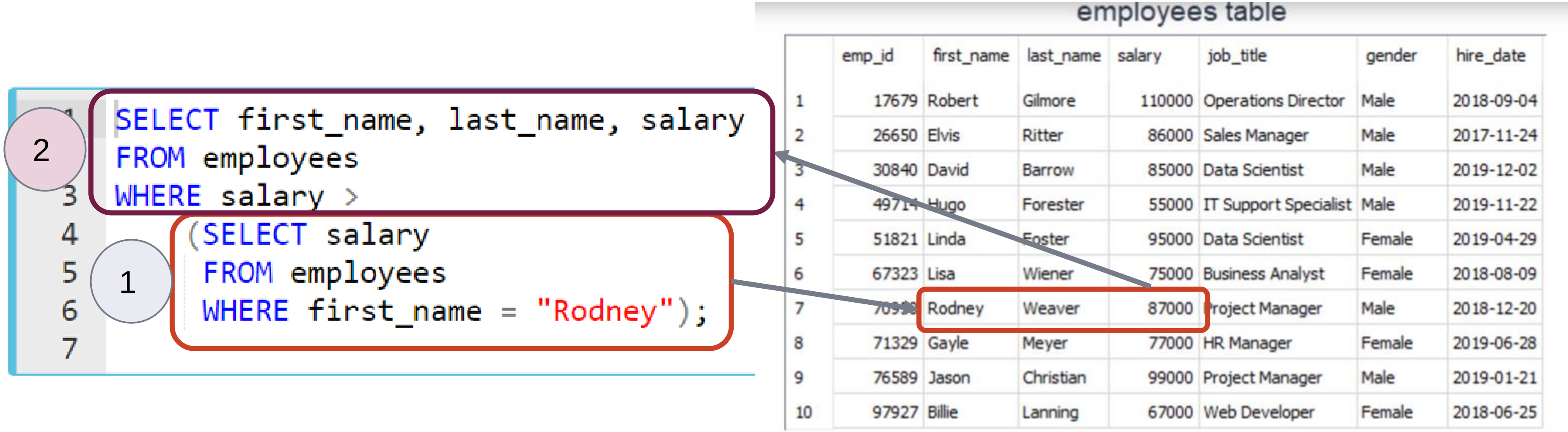


employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

1 The inner query is executed first and returns 87000 which is the salary of Rodney.

Analyze the query-2



```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (SELECT salary
     FROM employees
     WHERE first_name = "Rodney");
```

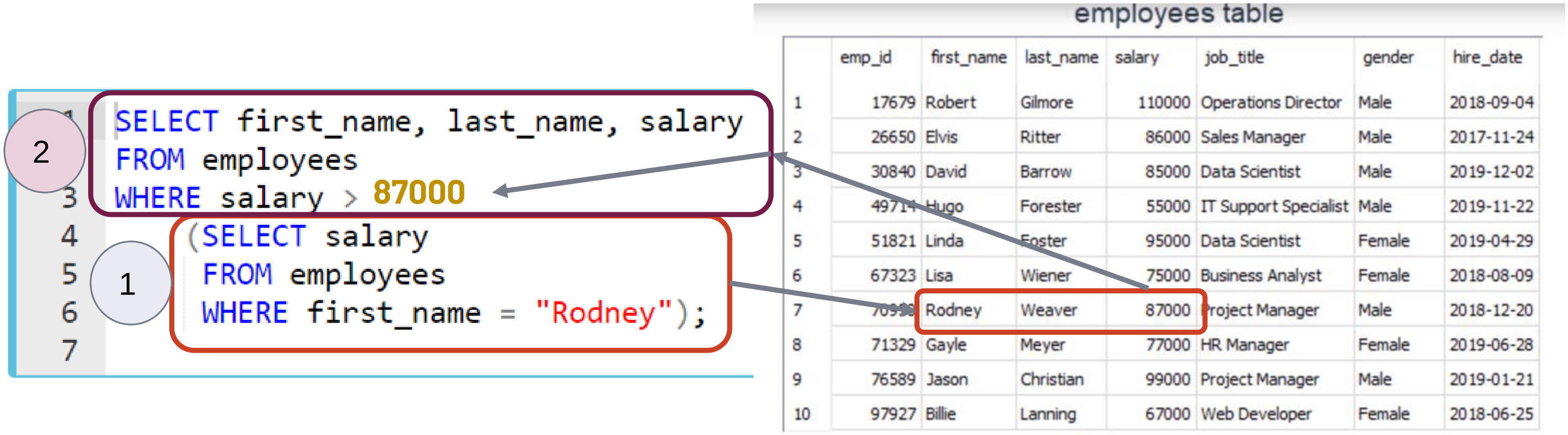
employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

1 The inner query is execute first and returns 87000 which is the salary of Rodney.

2 The value 87000 is passed to the outer query, in particular to the WHERE clause.

Analyze the query-3



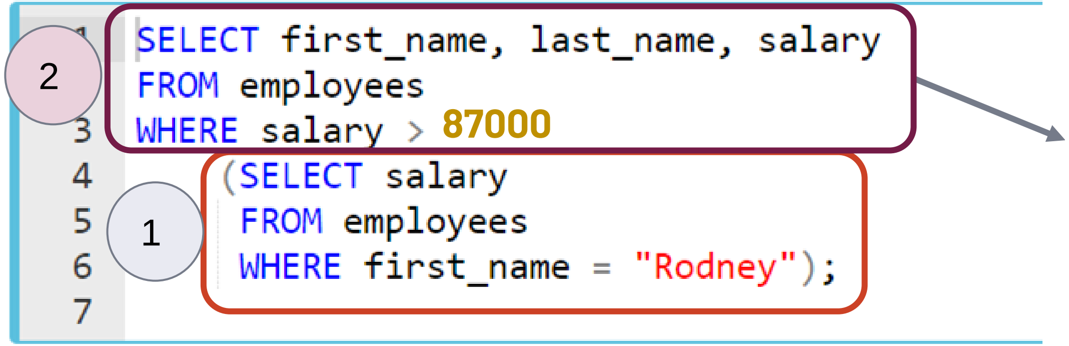
```
SELECT first_name, last_name, salary
FROM employees
WHERE salary > 87000
    (SELECT salary
     FROM employees
     WHERE first_name = "Rodney");
```

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

1 The inner query is execute first and returns 87000 which is the salary of Rodney.

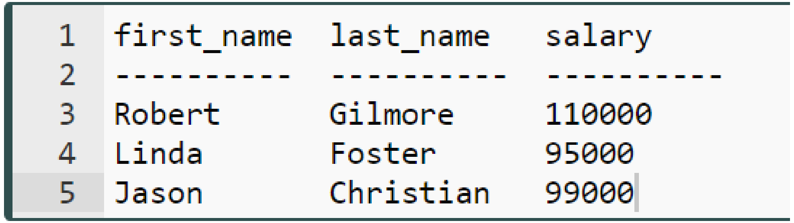
2 The value 87000 is passed to the outer query, in particular to the WHERE clause.

Analyze the query-4



```
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (SELECT salary
     FROM employees
     WHERE first_name = "Rodney");
```

output:



```
first_name  last_name   salary
----------  ----------  ----------
Robert      Gilmore     110000
Linda       Foster      95000
Jason       Christian   99000
```

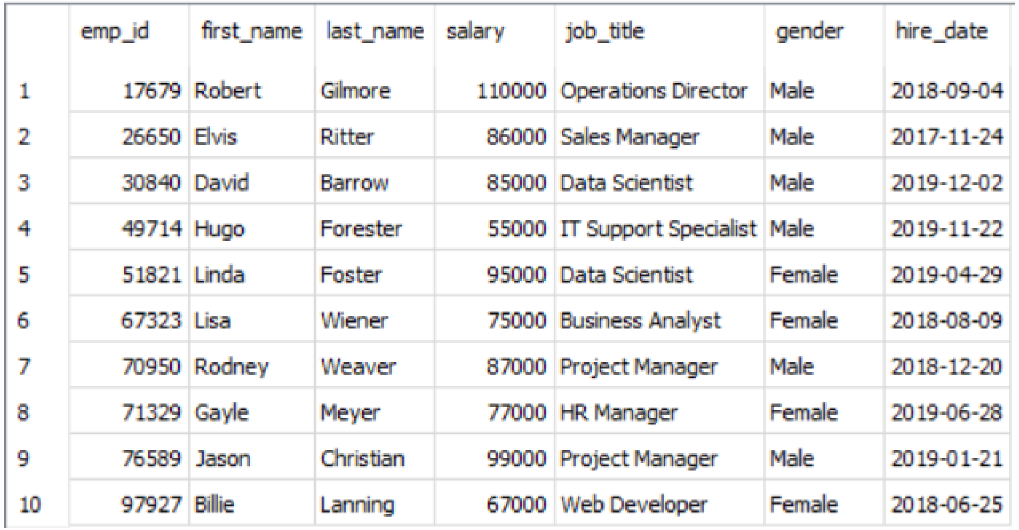
1 The inner query is execute first and returns 87000 which is the salary of Rodney.

2 The value 87000 is passed this value to the outer query, in particular to the WHERE clause.

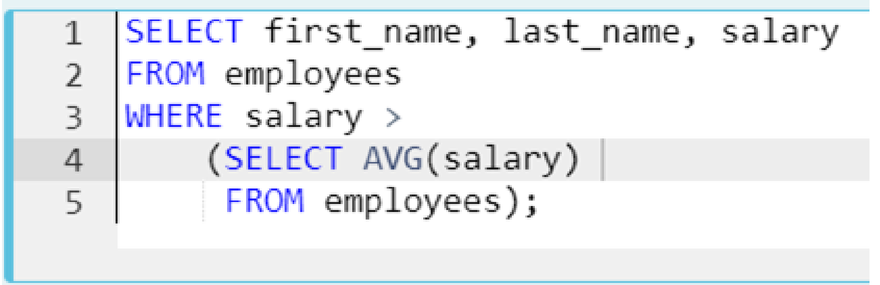
Example

Find out the employees who get paid more than the average salary

employees table



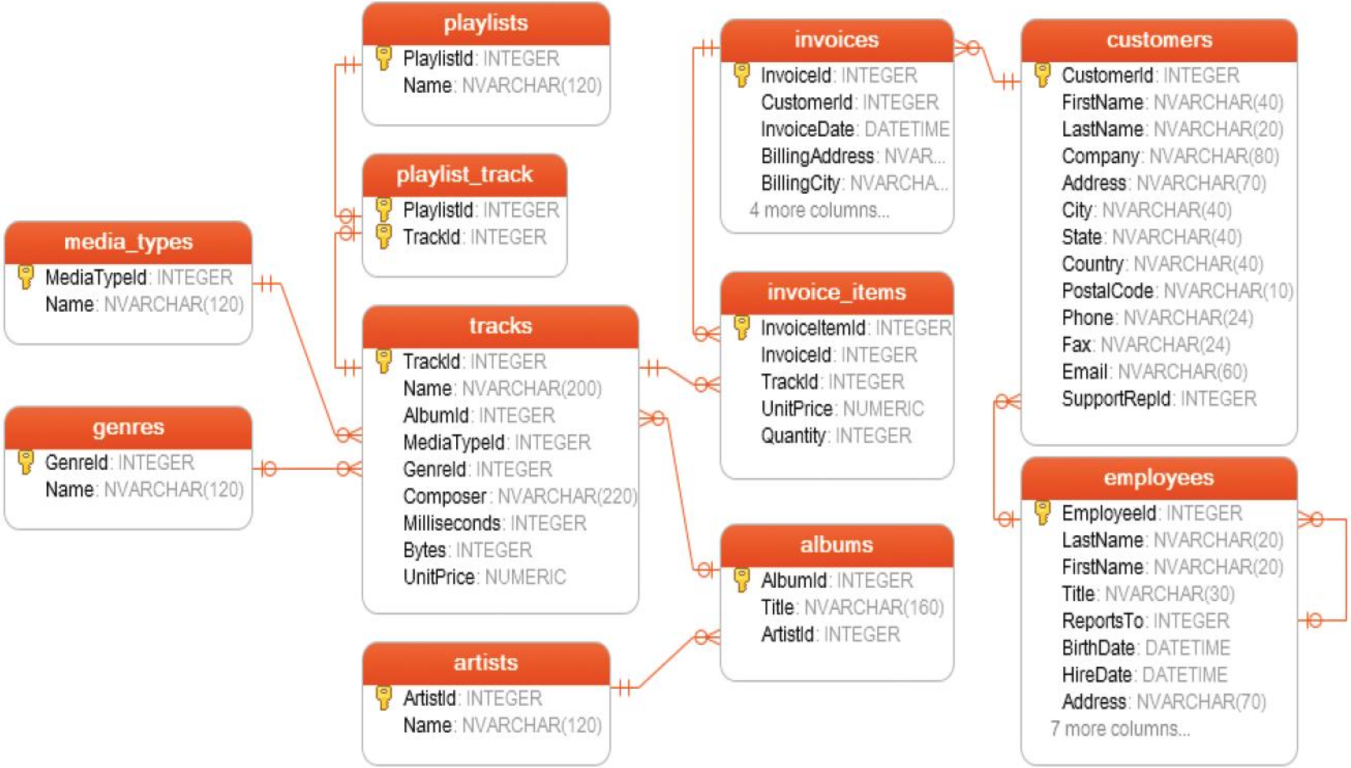
	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25



```
SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (SELECT AVG(salary)
     FROM employees);
```

Query Time

Retrieve track id, track name, album id info of the Album title ‘Faceless’. (**use : Subquery**)

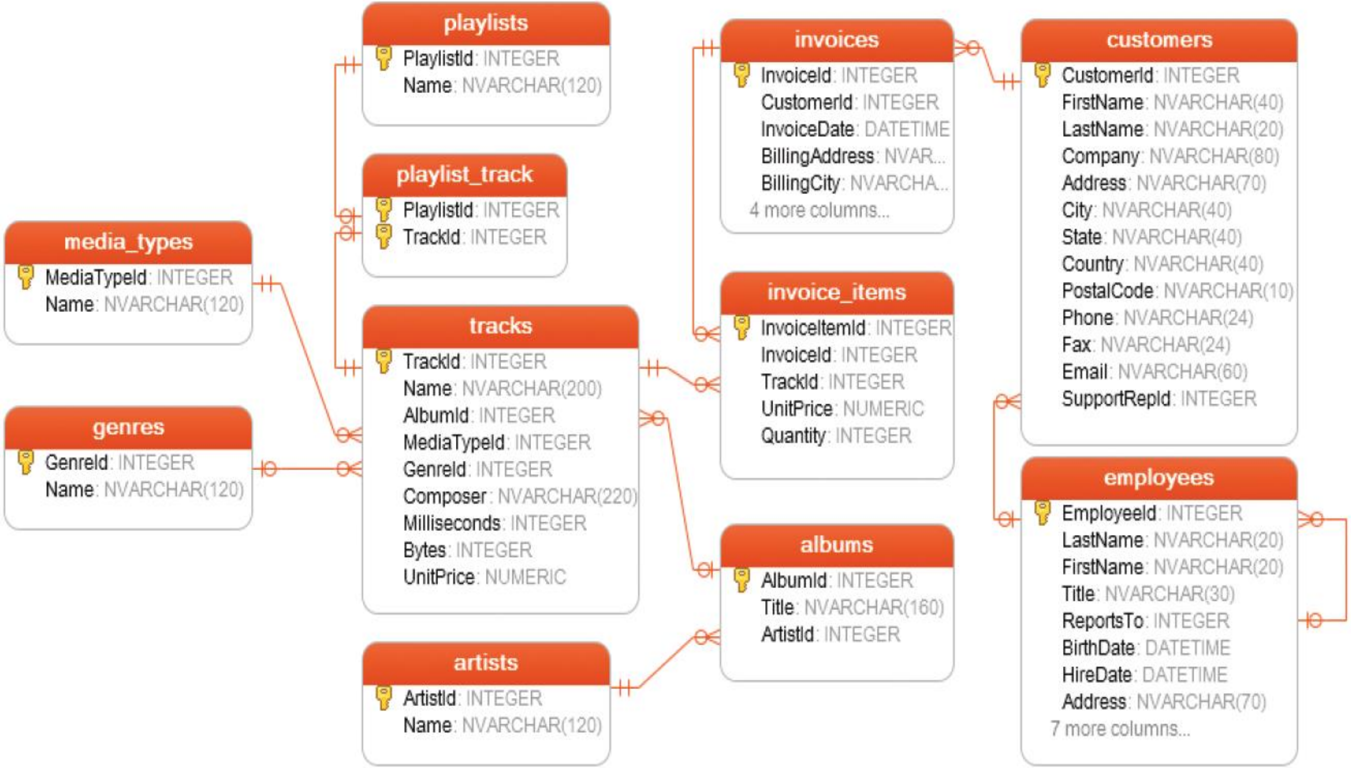


CLARUSWAY©
WAY TO REINVENT YOURSELF

PRO
TIP

Most queries using a join can be rewritten using a subquery (a query nested within another query), and most subqueries can be rewritten as joins.

Retrieve track id, track name, album id info of the Album title ‘Faceless’. (**use : Joins**)



playlists
- PlaylistId: INTEGER
- Name: NVARCHAR(120)

playlist_track
- PlaylistId: INTEGER
- TrackId: INTEGER

media_types
- MediaTypeId: INTEGER
- Name: NVARCHAR(120)

genres
- GenreId: INTEGER
- Name: NVARCHAR(120)

tracks
- TrackId: INTEGER
- Name: NVARCHAR(200)
- AlbumId: INTEGER
- MediaTypeId: INTEGER
- GenreId: INTEGER
- Composer: NVARCHAR(220)
- Milliseconds: INTEGER
- Bytes: INTEGER
- UnitPrice: NUMERIC

artists
- ArtistId: INTEGER
- Name: NVARCHAR(120)

invoices
- InvoiceId: INTEGER
- CustomerId: INTEGER
- InvoiceDate: DATETIME
- BillingAddress: NVAR...
- BillingCity: NVARCHA...
- 4 more columns...

invoice_items
- InvoiceItemId: INTEGER
- InvoiceId: INTEGER
- TrackId: INTEGER
- UnitPrice: NUMERIC
- Quantity: INTEGER

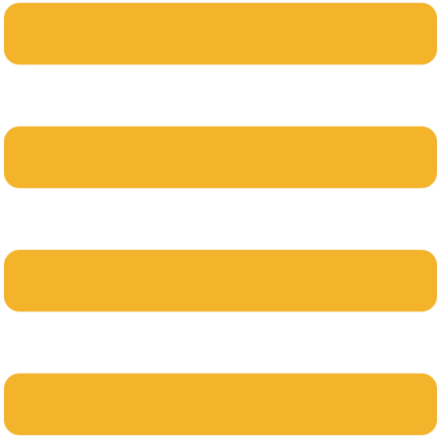
albums
- AlbumId: INTEGER
- Title: NVARCHAR(160)
- ArtistId: INTEGER

customers
- CustomerId: INTEGER
- FirstName: NVARCHAR(40)
- LastName: NVARCHAR(20)
- Company: NVARCHAR(80)
- Address: NVARCHAR(70)
- City: NVARCHAR(40)
- State: NVARCHAR(40)
- Country: NVARCHAR(40)
- PostalCode: NVARCHAR(10)
- Phone: NVARCHAR(24)
- Fax: NVARCHAR(24)
- Email: NVARCHAR(60)
- SupportRepId: INTEGER

employees
- EmployeeId: INTEGER
- LastName: NVARCHAR(20)
- FirstName: NVARCHAR(20)
- Title: NVARCHAR(30)
- ReportsTo: INTEGER
- BirthDate: DATETIME
- HireDate: DATETIME
- Address: NVARCHAR(70)
- 7 more columns...

Multiple-row Subqueries

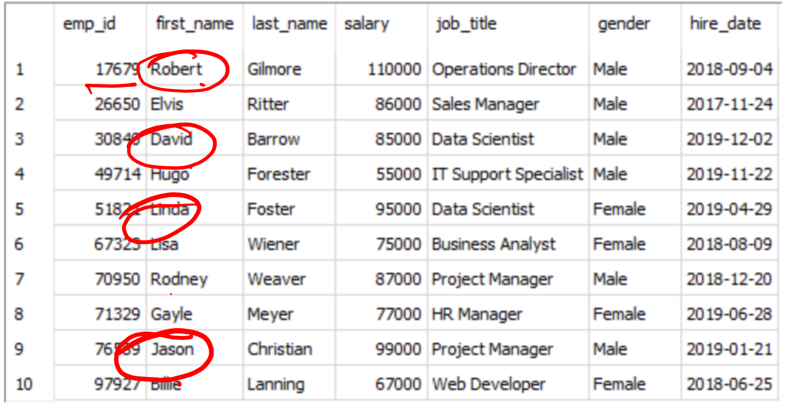
Multiple-row subqueries return sets of rows and are used with multiple-row operators such as **IN**, **NOT IN**, **ANY**, **ALL**.



Example

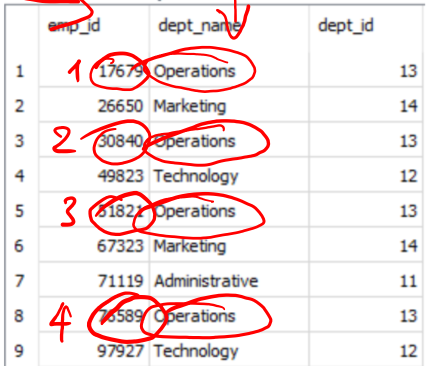
Find the employees (first name, last name from employees table) who work under the Operations department (departments table)

employees table



	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

departments table

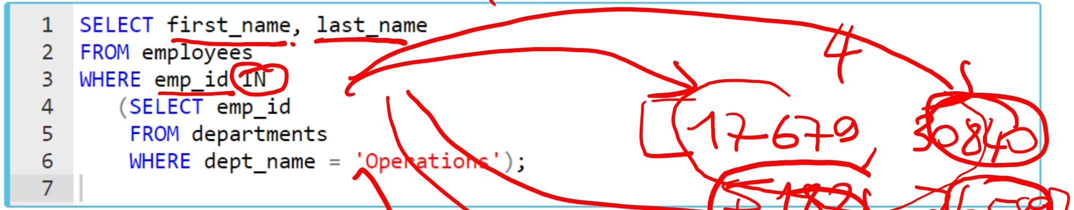


	emp_id	dept_name	dept_id
1	17679	Operations	13
2	26650	Marketing	14
3	30840	Operations	13
4	49823	Technology	12
5	51821	Operations	13
6	67323	Marketing	14
7	71119	Administrative	11
8	76589	Operations	13
9	97927	Technology	12

1 2 3 4

query:

NOT IN

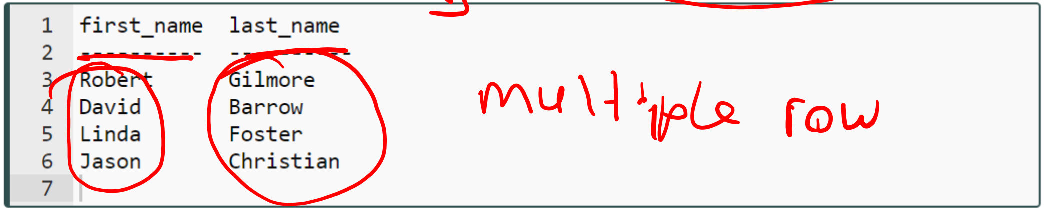


```
SELECT first_name, last_name
FROM employees
WHERE emp_id IN
   (SELECT emp_id
    FROM departments
    WHERE dept_name = 'Operations');

```

4

17679 30840
51821 76589

output:

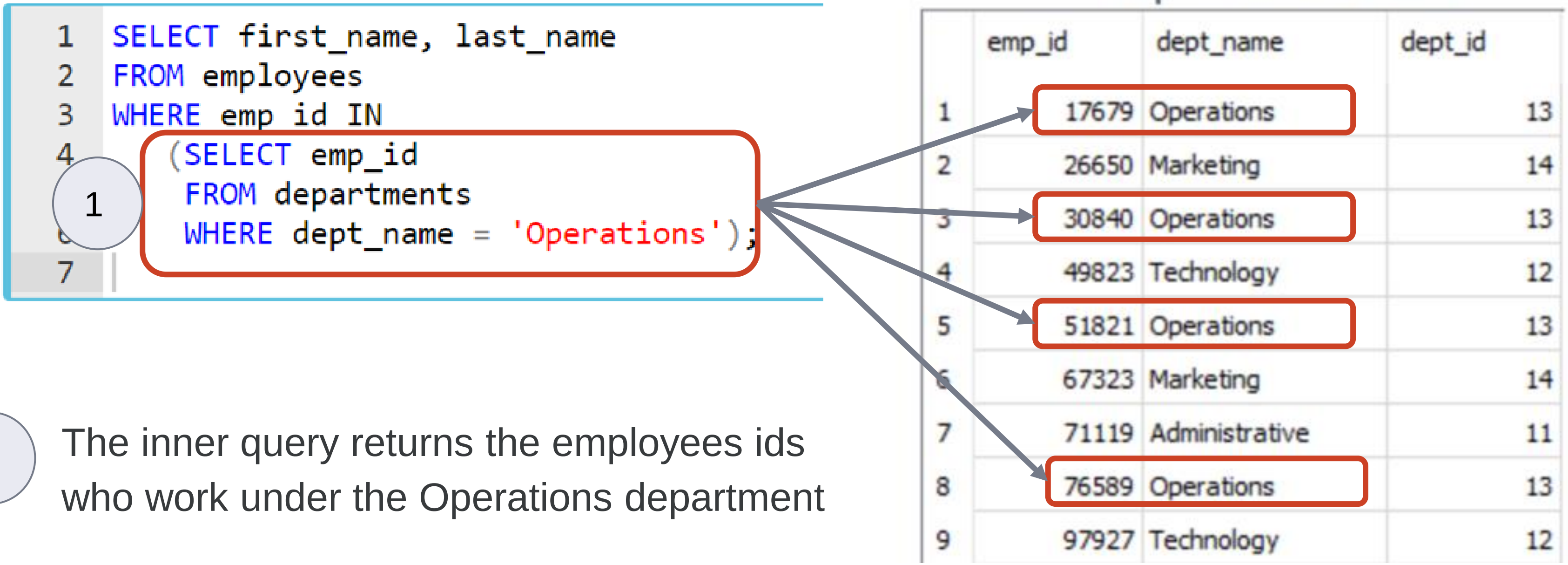


```
first_name  last_name
----------  ---------
Robert      Gilmore
David       Barrow
Linda       Foster
Jason       Christian

```

multiple row

CLARUSWAY©
WAY TO REINVENT YOURSELF

Analyze the query-1



```sql
SELECT first_name, last_name
FROM employees
WHERE emp_id IN
   (SELECT emp_id
    FROM departments
    WHERE dept_name = 'Operations');
```

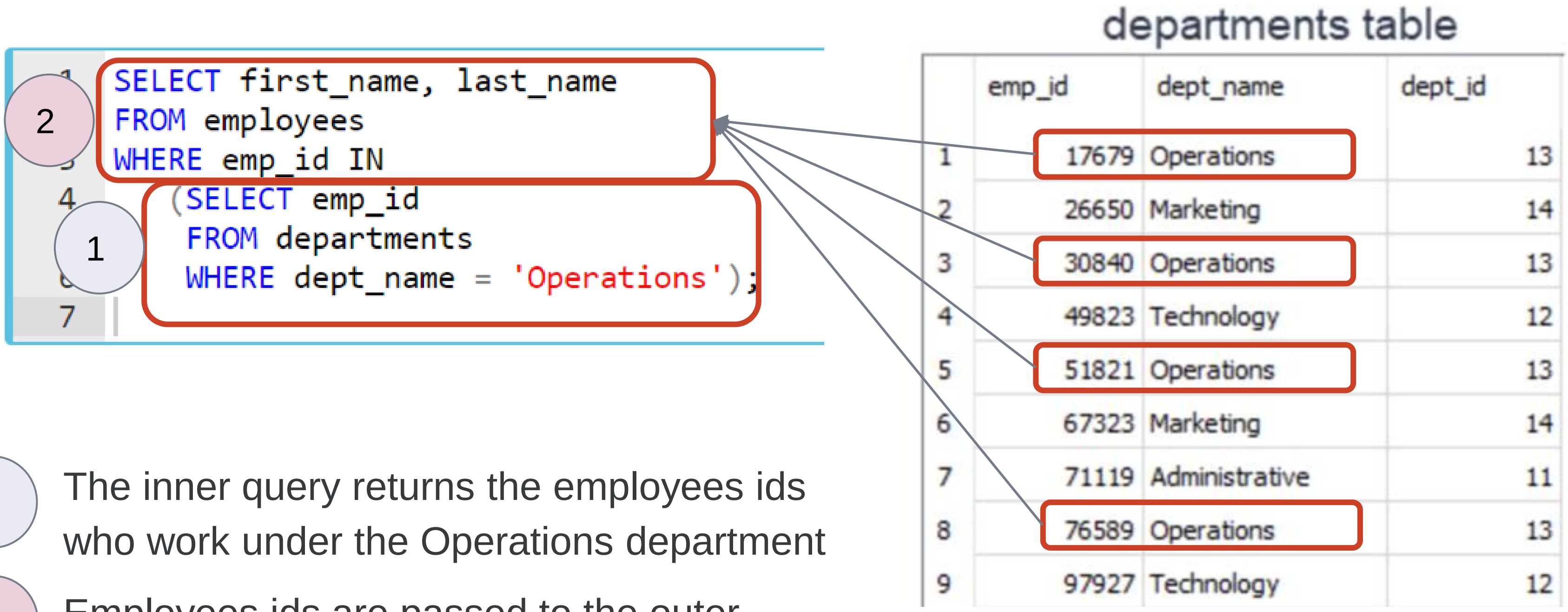
departments table

	emp_id	dept_name	dept_id
1	17679	Operations	13
2	26650	Marketing	14
3	30840	Operations	13
4	49823	Technology	12
5	51821	Operations	13
6	67323	Marketing	14
7	71119	Administrative	11
8	76589	Operations	13
9	97927	Technology	12

1 The inner query returns the employees ids who work under the Operations department

CLARUSWAY©
WAY TO REINVENT YOURSELF

Analyze the query-2



```sql
SELECT first_name, last_name
FROM employees
WHERE emp_id IN
    (SELECT emp_id
     FROM departments
     WHERE dept_name = 'Operations');
```

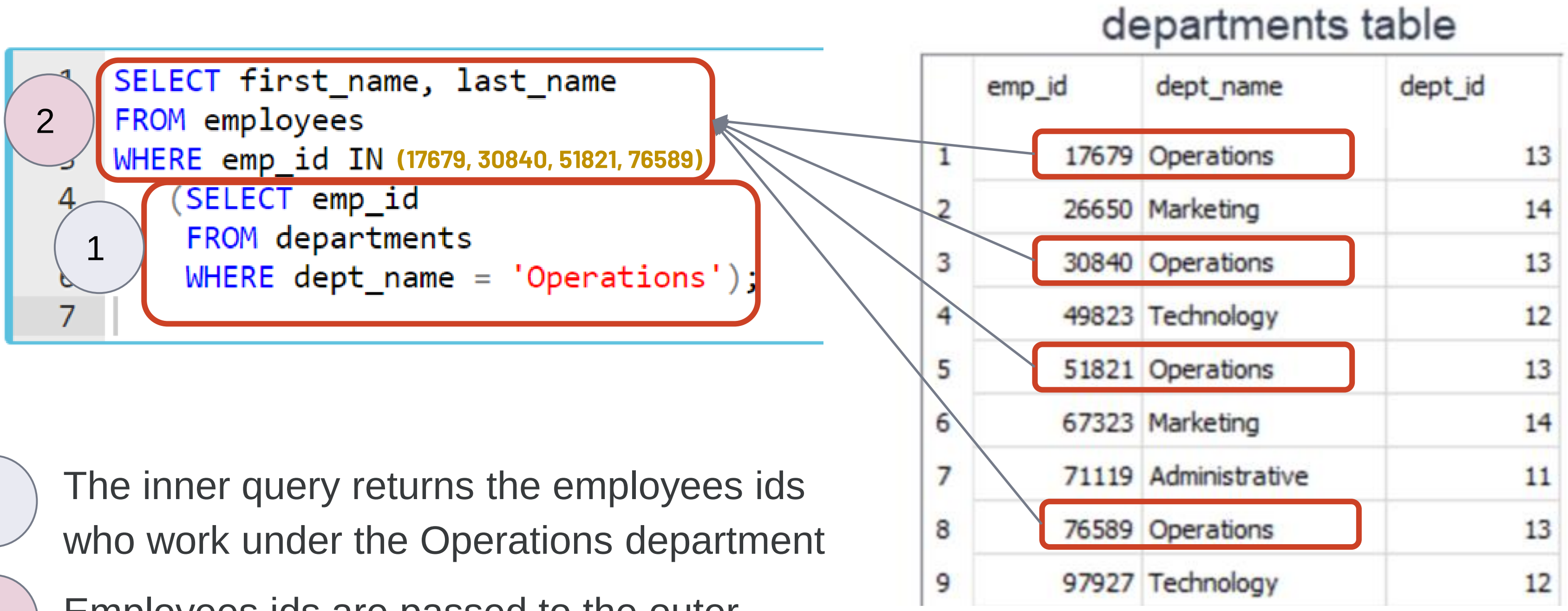
departments table

	emp_id	dept_name	dept_id
1	17679	Operations	13
2	26650	Marketing	14
3	30840	Operations	13
4	49823	Technology	12
5	51821	Operations	13
6	67323	Marketing	14
7	71119	Administrative	11
8	76589	Operations	13
9	97927	Technology	12

1. The inner query returns the employees ids who work under the Operations department

2. Employees ids are passed to the outer query.

Analyze the query-3



```sql
SELECT first_name, last_name
FROM employees
WHERE emp_id IN (17679, 30840, 51821, 76589)
   (SELECT emp_id
    FROM departments
    WHERE dept_name = 'Operations');
```

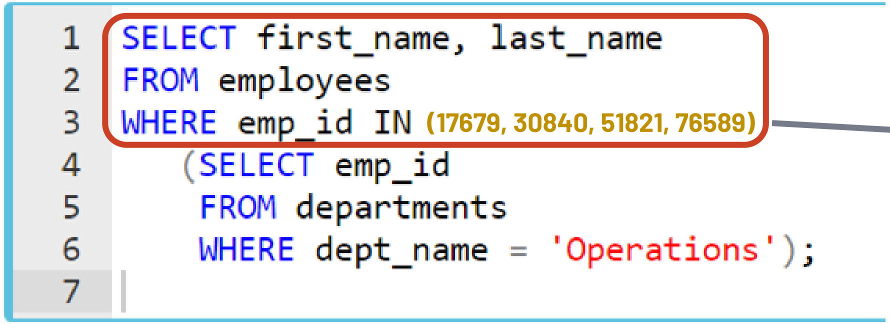
departments table

	emp_id	dept_name	dept_id
1	17679	Operations	13
2	26650	Marketing	14
3	30840	Operations	13
4	49823	Technology	12
5	51821	Operations	13
6	67323	Marketing	14
7	71119	Administrative	11
8	76589	Operations	13
9	97927	Technology	12

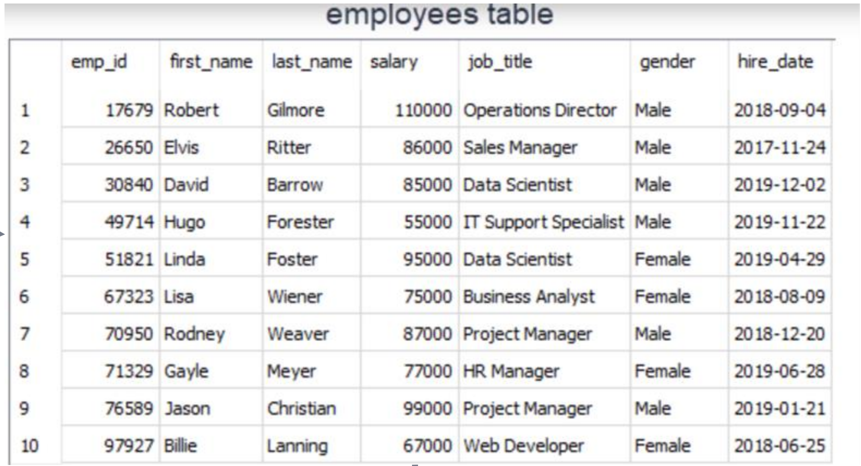
1 The inner query returns the employees ids who work under the Operations department

2 Employees ids are passed to the outer query.

Analyze the query-4



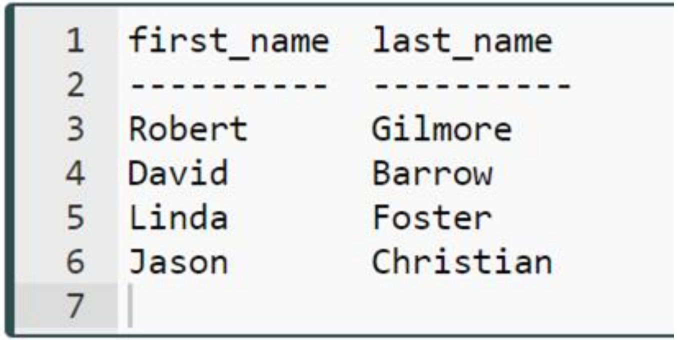
```
SELECT first_name, last_name
FROM employees
WHERE emp_id IN (17679, 30840, 51821, 76589)
    (SELECT emp_id
     FROM departments
     WHERE dept_name = 'Operations');
```



employees table

	emp_id	first_name	last_name	salary	job_title	gender	hire_date
1	17679	Robert	Gilmore	110000	Operations Director	Male	2018-09-04
2	26650	Elvis	Ritter	86000	Sales Manager	Male	2017-11-24
3	30840	David	Barrow	85000	Data Scientist	Male	2019-12-02
4	49714	Hugo	Forester	55000	IT Support Specialist	Male	2019-11-22
5	51821	Linda	Foster	95000	Data Scientist	Female	2019-04-29
6	67323	Lisa	Wiener	75000	Business Analyst	Female	2018-08-09
7	70950	Rodney	Weaver	87000	Project Manager	Male	2018-12-20
8	71329	Gayle	Meyer	77000	HR Manager	Female	2019-06-28
9	76589	Jason	Christian	99000	Project Manager	Male	2019-01-21
10	97927	Billie	Lanning	67000	Web Developer	Female	2018-06-25

output:

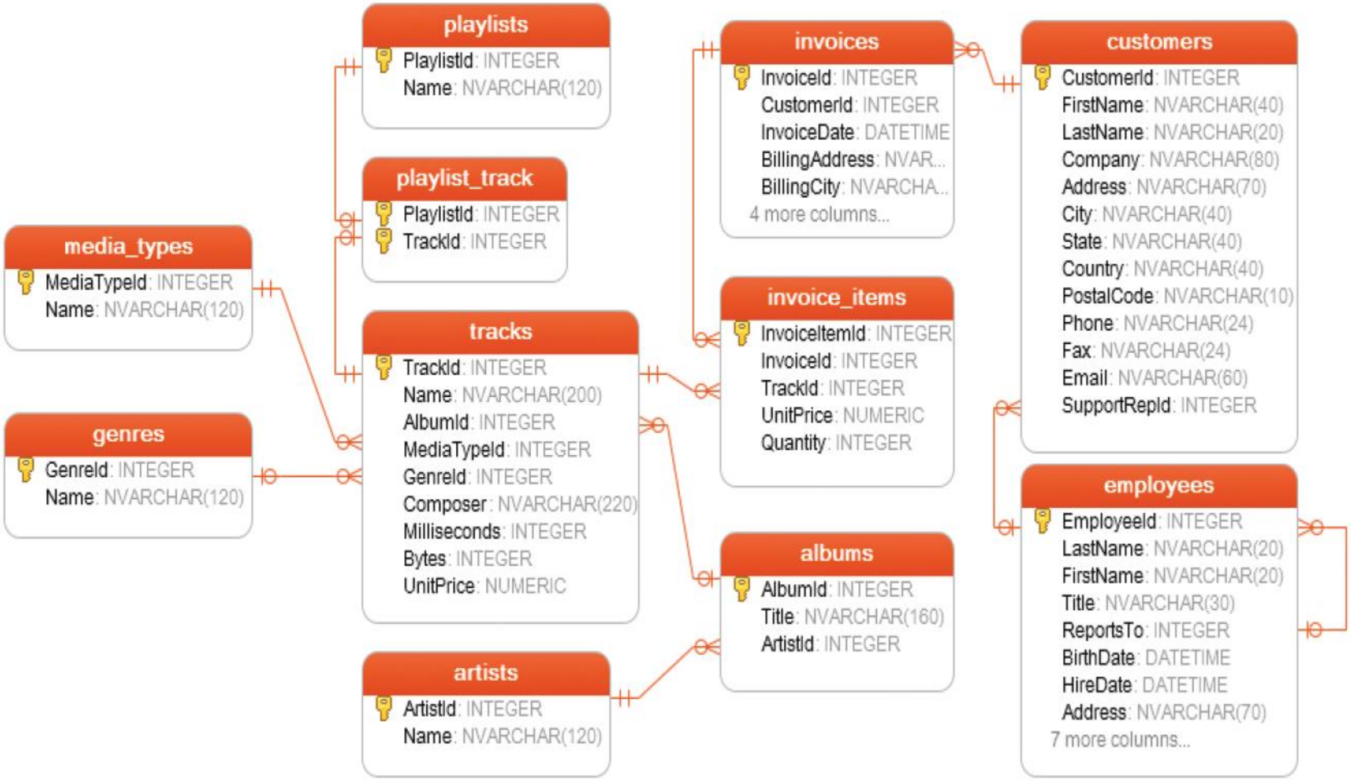


```
first_name  last_name
----------  ----------
Robert      Gilmore
David       Barrow
Linda       Foster
Jason       Christian
```

Outer query filters those employees ids and returns their first name and last name as a result set.

CLARUSWAY©
WAY TO REINVENT YOURSELF

Retrieve track id, track name, album id info of the Album title ‘Faceless’ and ‘Let There Be Rock’

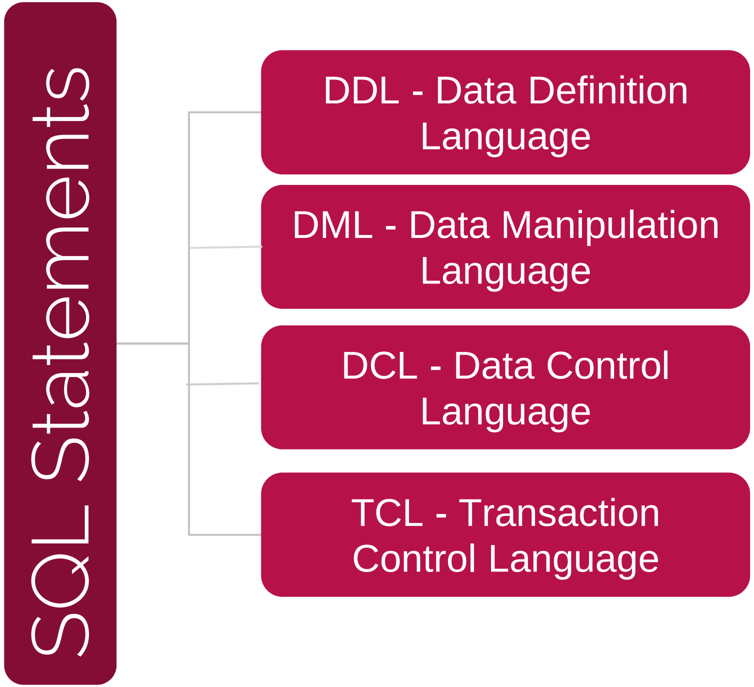


CLARUSWAY
WAY TO REINVENT YOURSELF

SQL

Session 4

Introduction



SQL Statements

DDL - Data Definition Language

DML - Data Manipulation Language

DCL - Data Control Language

TCL - Transaction Control Language

DDL Commands

Data Definition Language

- DDL specifies the database schema.
- Some statements used in DDL are **CREATE**, **ALTER**, **DROP**.
- DDL statements are typically used to set up and configure a new database before we insert data.

Data Manipulation Language

- Data Manipulation Language (DML) enables users to access or manipulate data.
- **INSERT**, **UPDATE**, **DELETE**, **SELECT*** are the statements used in DML.

* In some sources, SELECT statement is grouped into a different category called DQL (Data Query Language).

Data Control Language

- Data Control Language (DCL) is used to grant or revoke access control.
- Its statements are **REVOKE** and **GRANT**.

Transaction Control Language

- Transaction Control Language (TCL) controls the transactions of DML and DDL commands.
- Some statements in TCL are **COMMIT, ROLLBACK, SAVEPOINT**.

2 Data Types

Data Types

The data type of a column defines what value the column can hold: integer, character, date and time, binary, and so on.

Data Types

String

Date and Time

Numeric

String Data Types

The string data types are:

- CHAR
- VARCHAR
- BINARY
- VARBINARY
- BLOB
- TEXT
- ENUM
- SET

Date and Time Data Types

The date and time data types are:

- DATE
- DATETIME
- TIMESTAMP
- YEAR

Numeric Data Types

Integer Types (Exact Value)

- INTEGER or INT
- SMALLINT
- TINYINT
- MEDIUMINT
- BIGINT

Fixed-Point Types (Exact Value)

- DECIMAL
- NUMERIC

Floating-Point Types (Approximate Value)

- FLOAT
- DOUBLE

Data Types

PRO TIP

Data types might have different names in different database. And even if the name is the same, the size and other details may be different! Always check the documentation!

CREATE TABLE

When creating a table, we use **CREATE TABLE** statement.

Syntax of a Basic Create Table Statement

```
CREATE TABLE table_name
            (column_name1 data_type,
             column_name2 data_type);
```

CREATE TABLE-Example

```
CREATE TABLE employee
        (first_name VARCHAR(15),
         last_name VARCHAR(20),
         age INT,
         hire_date DATE);
```

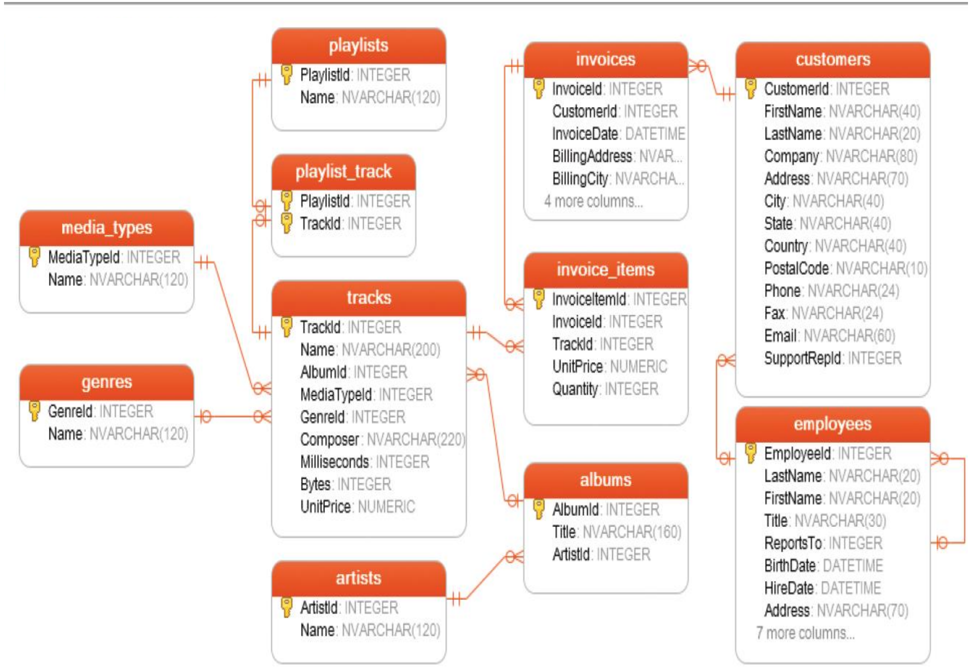
Note: Values in `VARCHAR` columns are variable-length strings. The length can be specified as a value from 0 to 65,535.

Query Time

Please add a table to your existing chinook database: The table name will be **leaves** we will use it ot keep record of the employees' annual or sick leaves

Column names:

- id
- employee_id
- start_date
- end_date



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

DROP TABLE

The DROP TABLE statement is used to drop an existing table in a database.

Syntax:

```
DROP TABLE table_name;

TRUNCATE TABLE table_name;
```

INSERT INTO

Syntax:

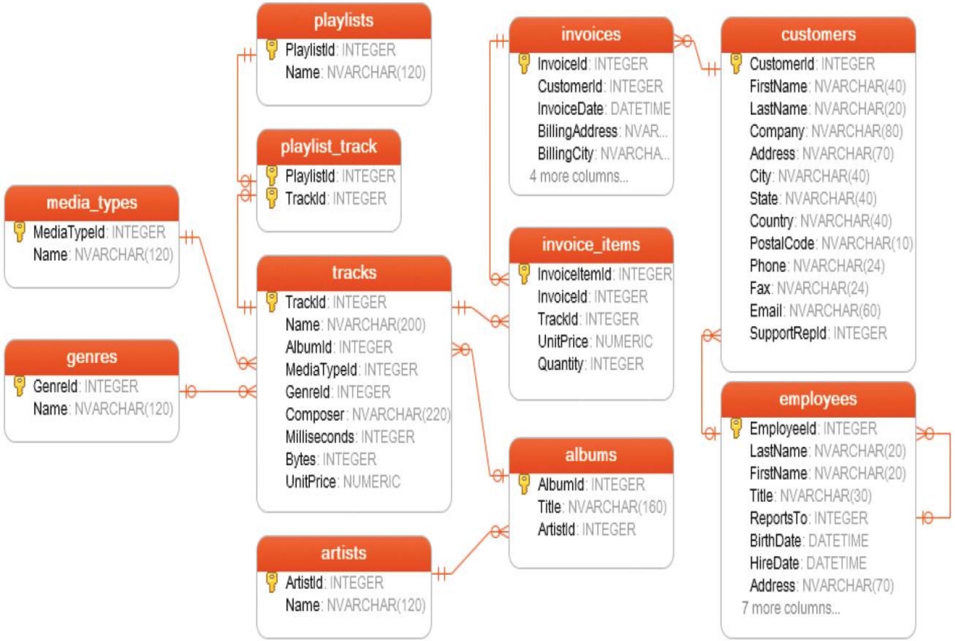
```
INSERT INTO table_name  (column1, column2 ,...)
               VALUES( value1, value2 ,....);
```

```
INSERT INTO table1 (column1,column2 ,...)
             VALUES
  (value1,value2 ,....),
  (value1,value2 ,....),
   ...
  (value1,value2 ,....);
```

Query Time

INSERT a record for an employee into leaves table

id INT,
employee_id INT,
start_date DATE,
end_date DATE



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

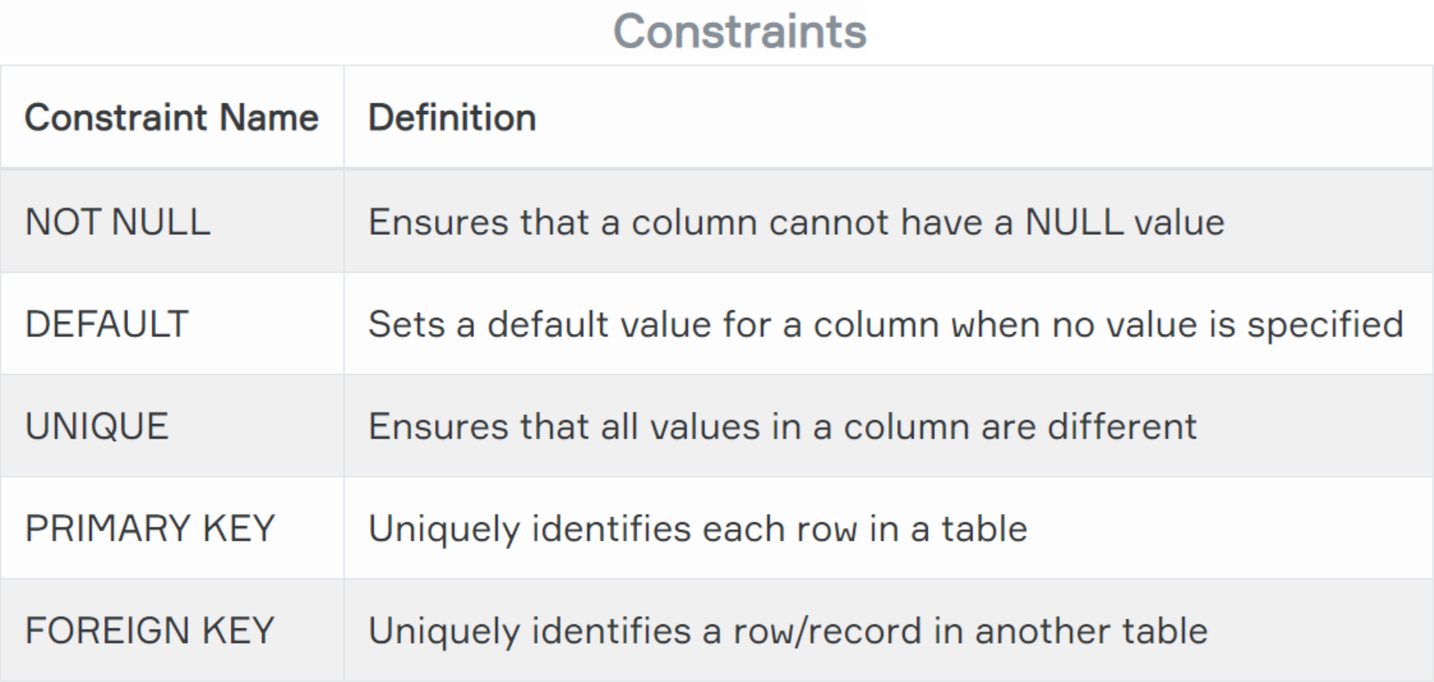
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Constraints

Constraints are the rules specified for data in a table. We can limit the type of data that will go into a table with the constraints. We can define the constraints with the CREATE TABLE statement or ALTER TABLE statement.

Constraints



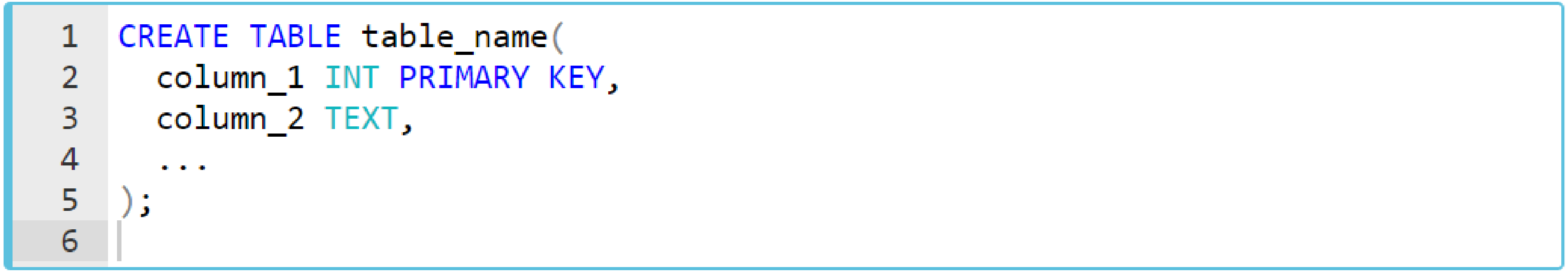
Constraints

Constraint Name	Definition
NOT NULL	Ensures that a column cannot have a NULL value
DEFAULT	Sets a default value for a column when no value is specified
UNIQUE	Ensures that all values in a column are different
PRIMARY KEY	Uniquely identifies each row in a table
FOREIGN KEY	Uniquely identifies a row/record in another table

Primary Key

The primary key is a column in our table that makes each row (aka, record) unique.

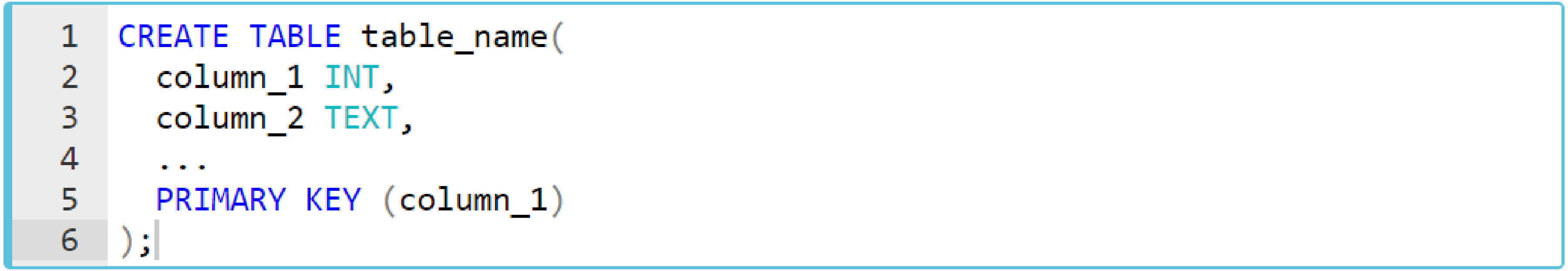
Syntax



```sql
CREATE TABLE table_name(
  column_1 INT PRIMARY KEY,
  column_2 TEXT,
  ...
);

```

Primary Key

Syntax (Alternative)



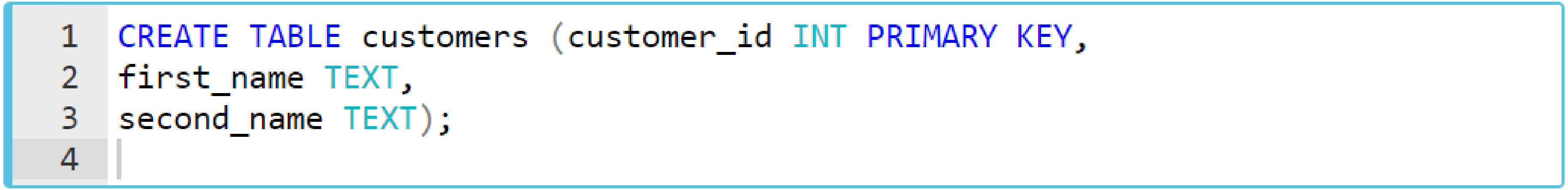
```sql
CREATE TABLE table_name(
  column_1 INT,
  column_2 TEXT,
  ...
  PRIMARY KEY (column_1)
);
```

Foreign Key

Foreign key is a column in a table that uniquely identifies each row of another table. That column refers to a primary key of another table. This creates a kind of link between the tables.

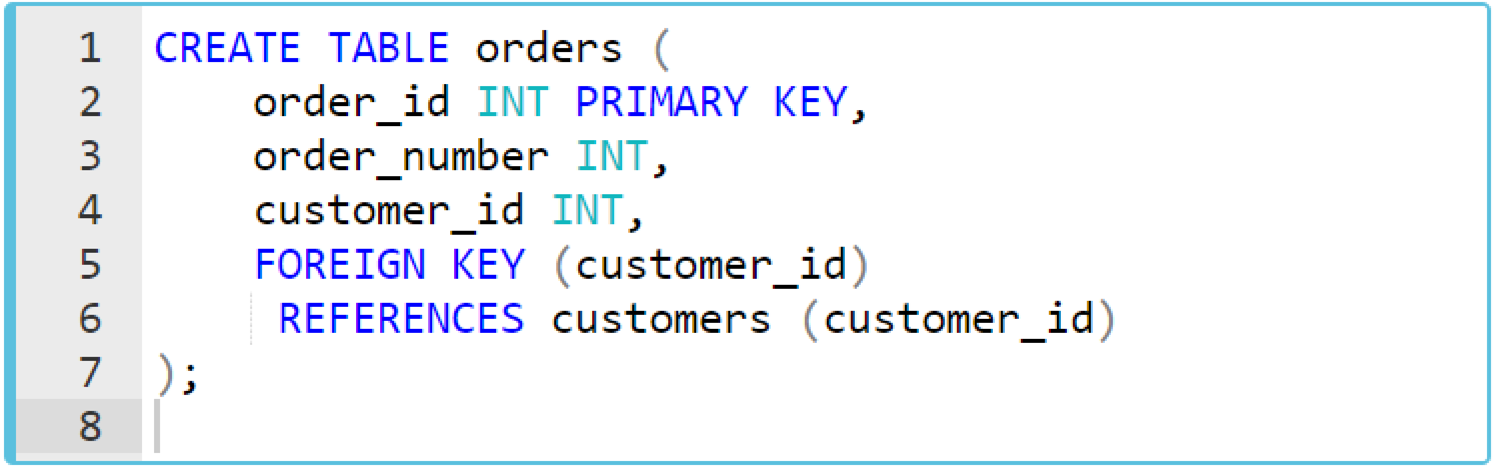
Foreign Key

customers



```sql
CREATE TABLE customers (customer_id INT PRIMARY KEY,
first_name TEXT,
second_name TEXT);

```

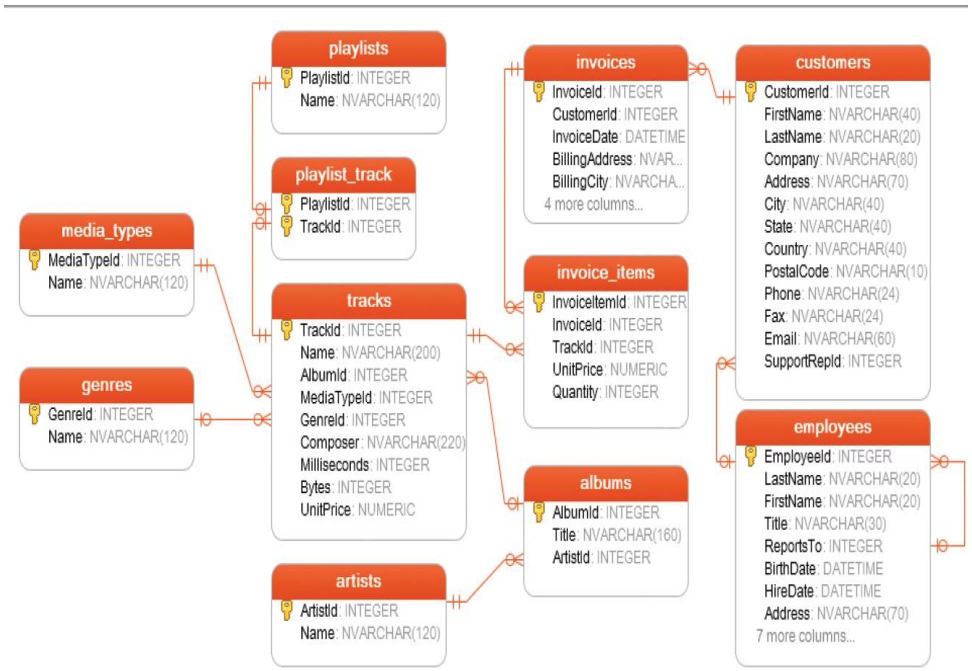
orders



```sql
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    order_number INT,
    customer_id INT,
    FOREIGN KEY (customer_id)
     REFERENCES customers (customer_id)
);

```

Query Time

Try to insert a record in albums table with an ArtistID=10000 and AlbumID=347



playlists
- PlaylistId: INTEGER
- Name: NVARCHAR(120)

playlist_track
- PlaylistId: INTEGER
- TrackId: INTEGER

media_types
- MediaTypeId: INTEGER
- Name: NVARCHAR(120)

genres
- GenreId: INTEGER
- Name: NVARCHAR(120)

tracks
- TrackId: INTEGER
- Name: NVARCHAR(200)
- AlbumId: INTEGER
- MediaTypeId: INTEGER
- GenreId: INTEGER
- Composer: NVARCHAR(220)
- Milliseconds: INTEGER
- Bytes: INTEGER
- UnitPrice: NUMERIC

artists
- ArtistId: INTEGER
- Name: NVARCHAR(120)

invoices
- InvoiceId: INTEGER
- CustomerId: INTEGER
- InvoiceDate: DATETIME
- BillingAddress: NVAR...
- BillingCity: NVARCHA...
- 4 more columns...

invoice_items
- InvoiceItemId: INTEGER
- InvoiceId: INTEGER
- TrackId: INTEGER
- UnitPrice: NUMERIC
- Quantity: INTEGER

albums
- AlbumId: INTEGER
- Title: NVARCHAR(160)
- ArtistId: INTEGER

customers
- CustomerId: INTEGER
- FirstName: NVARCHAR(40)
- LastName: NVARCHAR(20)
- Company: NVARCHAR(80)
- Address: NVARCHAR(70)
- City: NVARCHAR(40)
- State: NVARCHAR(40)
- Country: NVARCHAR(40)
- PostalCode: NVARCHAR(10)
- Phone: NVARCHAR(24)
- Fax: NVARCHAR(24)
- Email: NVARCHAR(60)
- SupportRepId: INTEGER

employees
- EmployeeId: INTEGER
- LastName: NVARCHAR(20)
- FirstName: NVARCHAR(20)
- Title: NVARCHAR(30)
- ReportsTo: INTEGER
- BirthDate: DATETIME
- HireDate: DATETIME
- Address: NVARCHAR(70)
- 7 more columns...

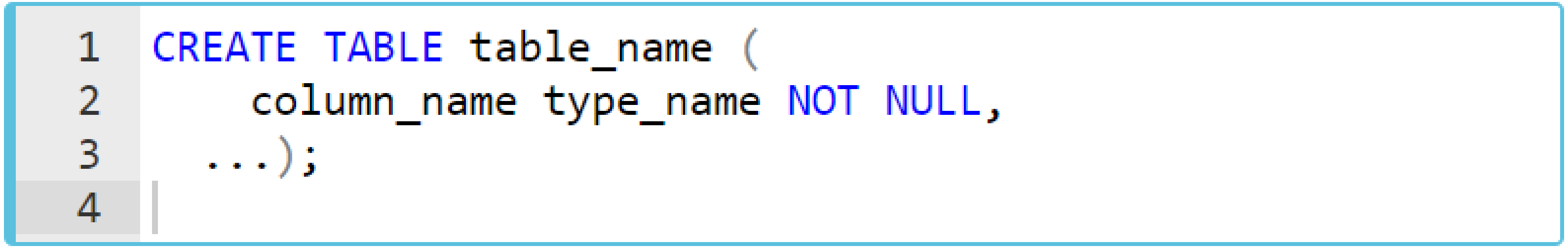
Not Null

A column can include NULL values. A NULL value is a special value that means the value is unknown or does not exist.

All columns (except primary key's column) in a table can hold NULL values unless we explicitly specify NOT NULL constraints.

Not Null

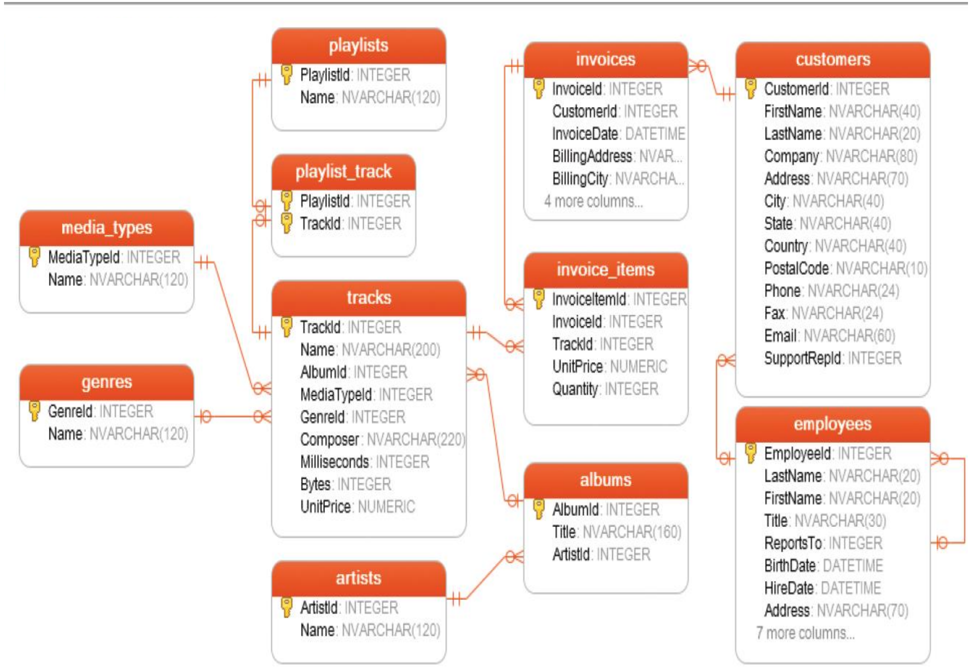
Syntax



```
CREATE TABLE table_name (
    column_name type_name NOT NULL,
  ...);

```

Query Time

Try to insert a record in albums table without a title value



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Query Time

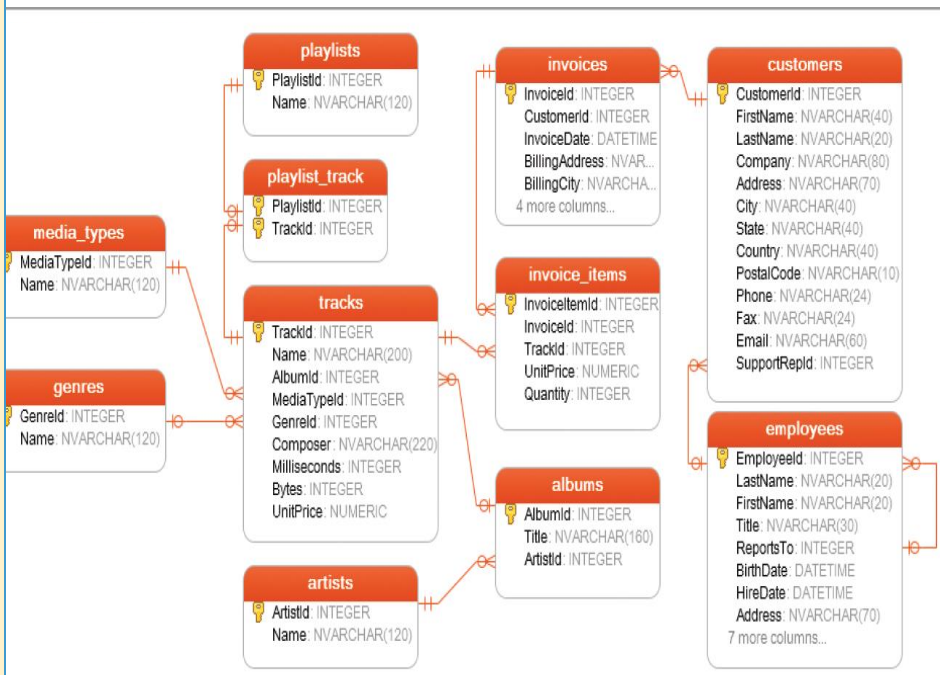
Please drop the table as you've just created writing

DROP TABLE leaves;

Then, recreate the leaves table adding constraints as below:

Column names:

- id -> PRIMARY KEY , AUTOINC
- employee_id -> FOREING KEY
- start_date -> NOT NULL
- end_date -> NOT NULL



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

genres
GenreId: INTEGER
Name: NVARCHAR(120)

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

4

ALTER TABLE

ALTER TABLE

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
It is also used to add and drop various constraints on an existing table.

To add a column in a table, use the following syntax:

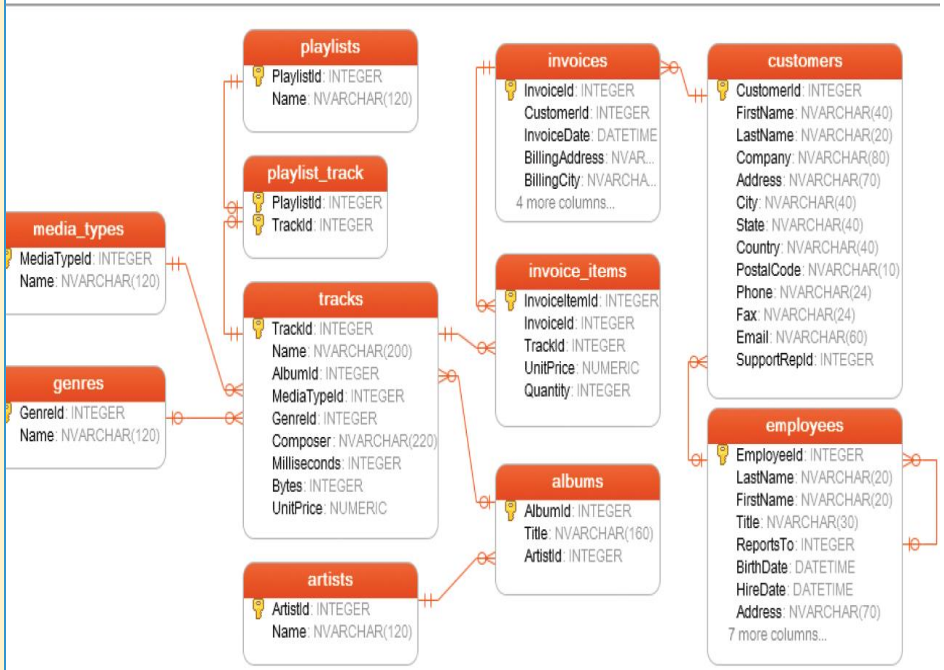
```
ALTER TABLE table_name
ADD  column_name data_type;
```

Query Time

Alter the table name to employee_leaves first. (Google this one know if you don't know)

Then add a column to your leaves table named "leave_type".

We will use type of leaves such as "annual leave", "sick leave" and etc.



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

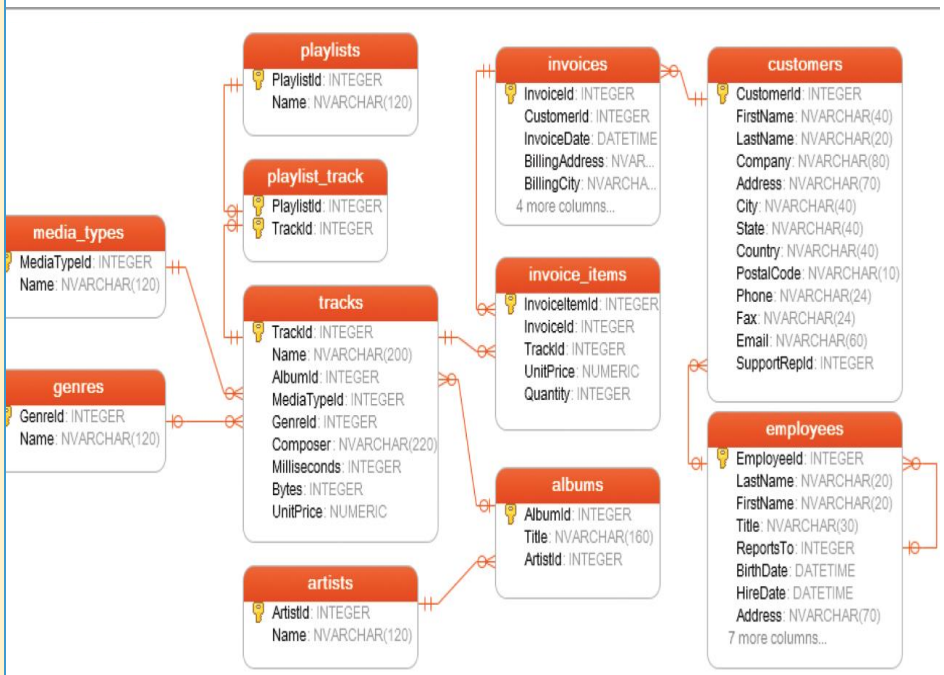
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Query Time

INSERT 3 new records to employee_leaves table.

You can use “annual_leave”, sick_leave” and etc for leave type



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

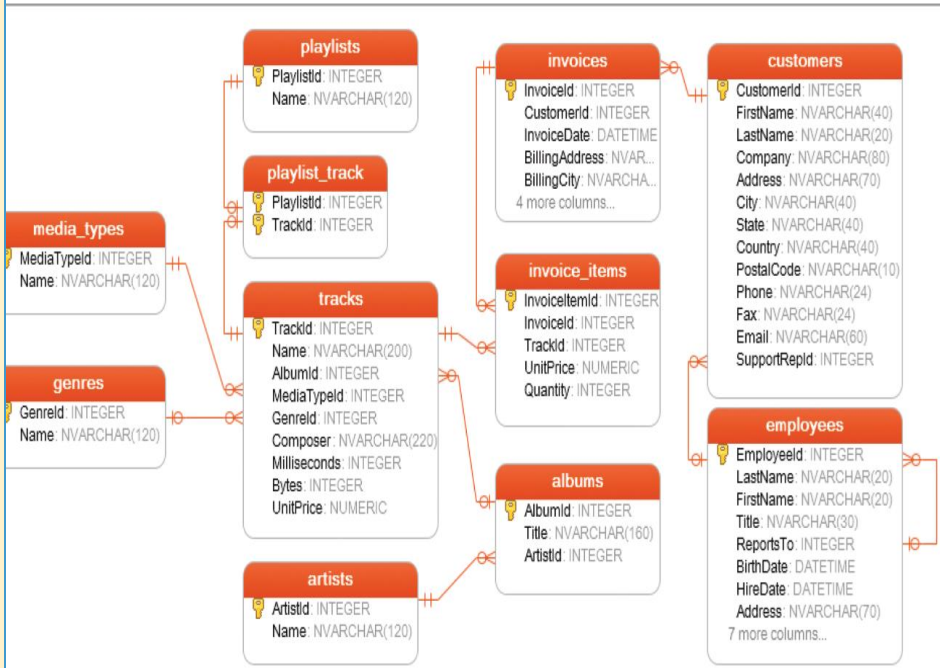
Query Time

Now add another table leave_types with

id -> PK AUTOINC
leave_name ->TEXT

And make the column in employee_leaves table as FOREIGN KEY

ADD 3 records to leave_types and employee_leaves table



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

genres
GenreId: INTEGER
Name: NVARCHAR(120)

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

ALTER TABLE

To delete a column in a table, use the following syntax:

```
ALTER TABLE table_name
DROP  column_name;
```

To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
MODIFY COLUMN  column_name data_type;
```

***Not works in SQLite but you can use them in other RDBMS**

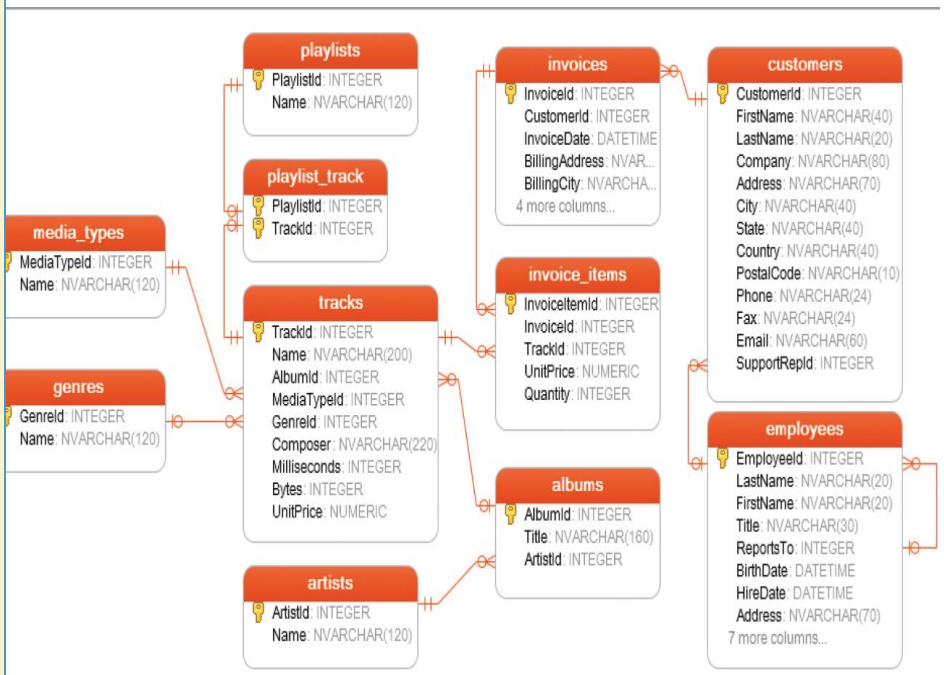
Query Time

Copy the entire table to a new table using csv import method

drop a column (you can't!)

change data type of a column

drop table



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

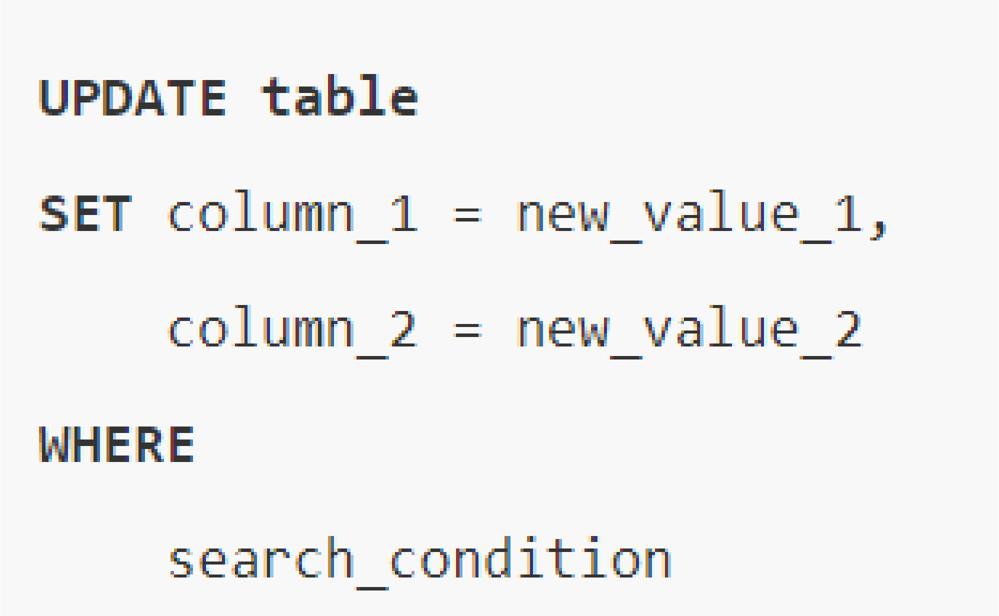
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

UPDATE TABLE



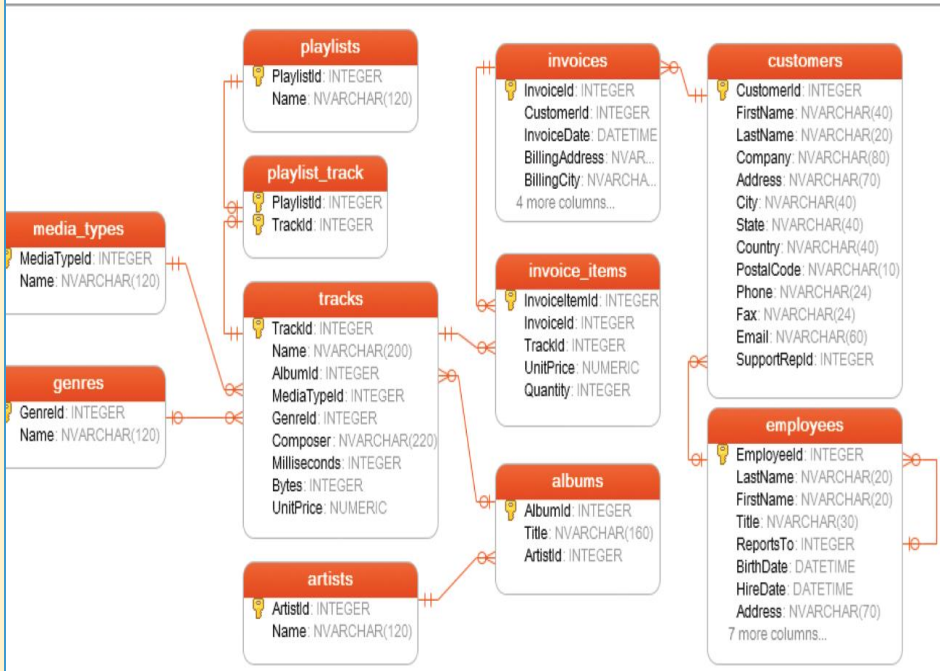
```
UPDATE table

SET column_1 = new_value_1,

    column_2 = new_value_2

WHERE

    search_condition
```

Query Time

Change the name of Annual leave to Marriage Leave in leave_types table

Change the start and end date values of a record in employee_leaves table

google sqlite date add



playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)

playlist_track
PlaylistId: INTEGER
TrackId: INTEGER

media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)

genres
GenreId: INTEGER
Name: NVARCHAR(120)

tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

artists
ArtistId: INTEGER
Name: NVARCHAR(120)

invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

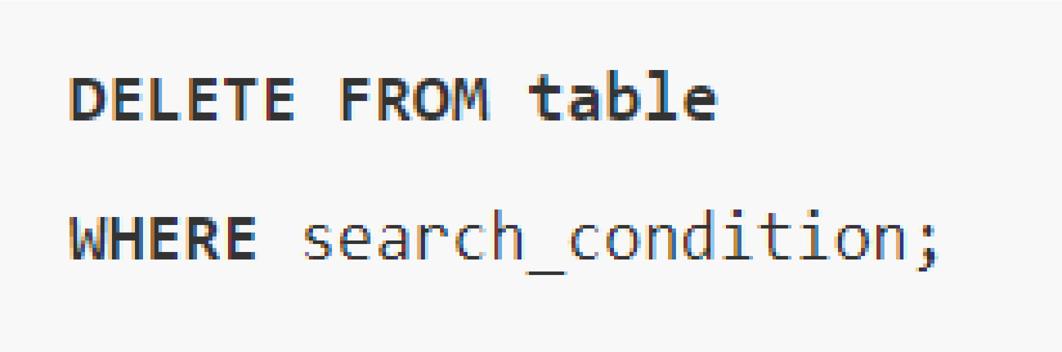
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

DELETE

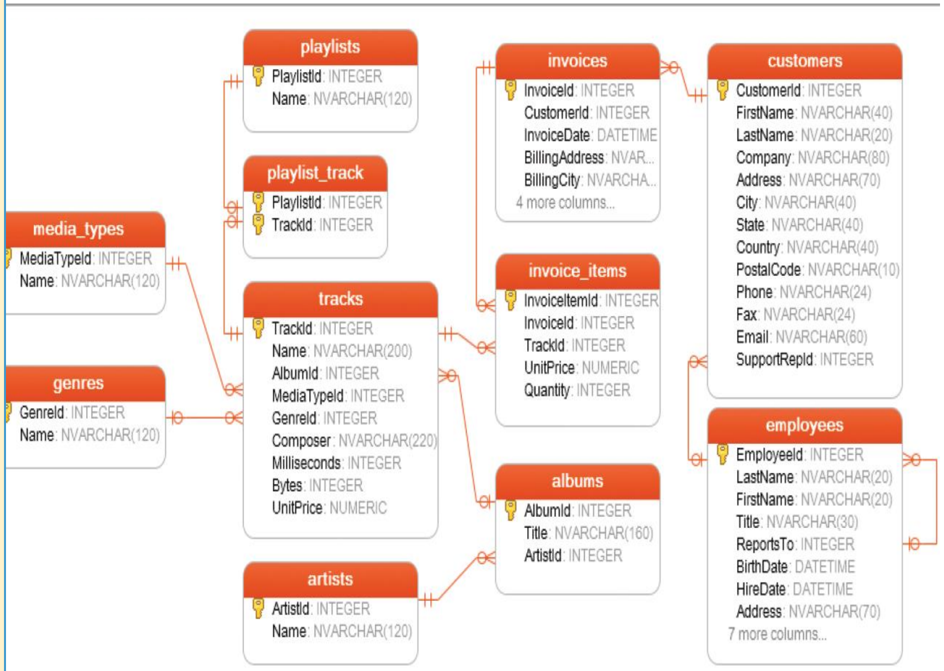


```
DELETE FROM table

WHERE search_condition;
```

Query Time

Delete a record from leave types table

Delete a record from employee leaves table



playlists
- PlaylistId: INTEGER
- Name: NVARCHAR(120)

playlist_track
- PlaylistId: INTEGER
- TrackId: INTEGER

media_types
- MediaTypeId: INTEGER
- Name: NVARCHAR(120)

genres
- GenreId: INTEGER
- Name: NVARCHAR(120)

tracks
- TrackId: INTEGER
- Name: NVARCHAR(200)
- AlbumId: INTEGER
- MediaTypeId: INTEGER
- GenreId: INTEGER
- Composer: NVARCHAR(220)
- Milliseconds: INTEGER
- Bytes: INTEGER
- UnitPrice: NUMERIC

artists
- ArtistId: INTEGER
- Name: NVARCHAR(120)

invoices
- InvoiceId: INTEGER
- CustomerId: INTEGER
- InvoiceDate: DATETIME
- BillingAddress: NVAR...
- BillingCity: NVARCHA...
- 4 more columns...

invoice_items
- InvoiceItemId: INTEGER
- InvoiceId: INTEGER
- TrackId: INTEGER
- UnitPrice: NUMERIC
- Quantity: INTEGER

albums
- AlbumId: INTEGER
- Title: NVARCHAR(160)
- ArtistId: INTEGER

customers
- CustomerId: INTEGER
- FirstName: NVARCHAR(40)
- LastName: NVARCHAR(20)
- Company: NVARCHAR(80)
- Address: NVARCHAR(70)
- City: NVARCHAR(40)
- State: NVARCHAR(40)
- Country: NVARCHAR(40)
- PostalCode: NVARCHAR(10)
- Phone: NVARCHAR(24)
- Fax: NVARCHAR(24)
- Email: NVARCHAR(60)
- SupportRepId: INTEGER

employees
- EmployeeId: INTEGER
- LastName: NVARCHAR(20)
- FirstName: NVARCHAR(20)
- Title: NVARCHAR(30)
- ReportsTo: INTEGER
- BirthDate: DATETIME
- HireDate: DATETIME
- Address: NVARCHAR(70)
- 7 more columns...

THANKS!

Any questions?

You can find me at: