



SQL

Session 3

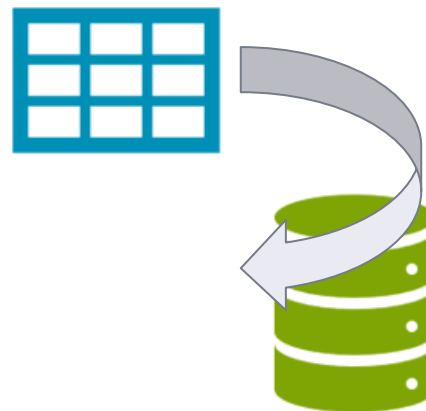


Table of Contents



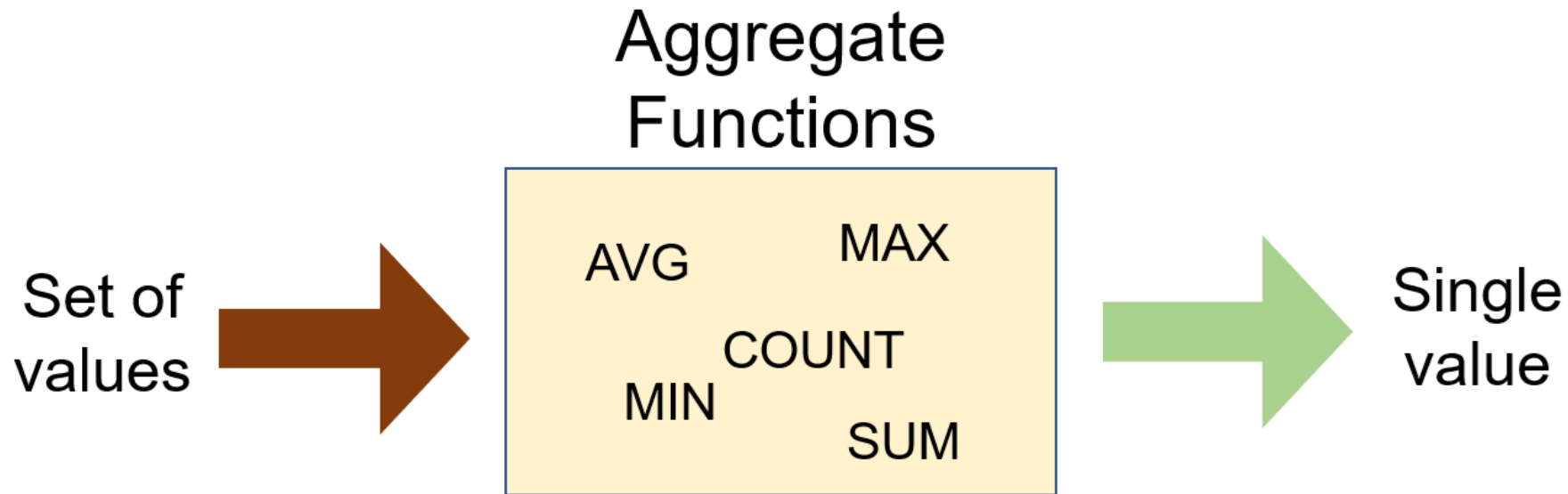
- ▶ What are type of Aggregate Functions, why do we need them?
- ▶ Group By Clause
- ▶ What are type of Joins in SQL, why do we need them



Aggregate Functions



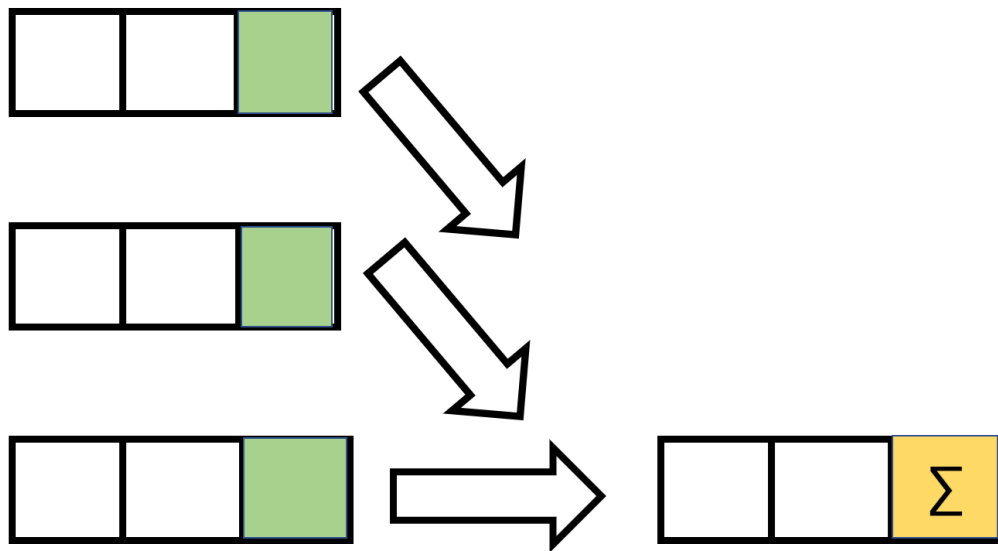
What is an aggregate function?



What is an aggregate function?



Aggregate Functions



SUM and AVG → numeric values

MIN, MAX, COUNT → numeric & non-numeric (strings, date, etc.)

We will learn **GROUP BY** clause and **HAVING** clause later.

What is **NULL**?



What is NULL?

NULL means no data and is a special value in SQL. It shows us that a piece of information is unknown or missing or not applicable.

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer
1	For Those About To Rock (We Salute...)	1	1	1	Angus Young, Malcolm Young, Brian ...
2	Balls to the Wall	2	2	1	NULL
3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U. Dirkschneide...
4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel, S. ...
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel
6	Put The Finger On You	1	1	1	Angus Young, Malcolm Young, Brian ...
7	Let's Get It Up	1	1	1	Angus Young, Malcolm Young, Brian ...



- NULL value represents the unknown value or missing value or not applicable.
- NULL is not equal to zero or empty string.
- NULL is not equal to itself.



2

COUNT Function



COUNT Function



We use **COUNT** function to count the numbers of records (a.k.a row) in a table.

Syntax

```
1 SELECT COUNT(column_name)
2 FROM table_name;
3 |
```

COUNT Function



How many students have enrolled the courses?

student_info table

	student_id	first_name	last_name	gender	state	county	field	start_date
1	110028	Michael	Crawford	M	Virginia	Albemarle	DevOps	2019-07-19
2	110078	Olivia	Smith	F	West Virginia	Tucker	Back-End Developer	2019-03-11
3	110080	Amelia	Anderson	F	West Virginia	Webster	Data Analysis	2019-04-25
4	110081	Megan	King	F	West Virginia	Kanawha	Back-End Developer	2019-06-07
5	110091	Richard	Morgan	M	Virginia	Prince William	DevOps	2019-06-28
6	110095	Hugo	Wallace	M	Virginia	Accomack	Data Analysis	2019-06-16
7	120001	Eleanor	Johnson	F	West Virginia	Wyoming	Front-End Developer	2019-03-22
8	120011	Oliver	Taylor	M	West Virginia	Hancock	Data Analysis	2019-04-14
9	120033	Lucas	Parker	M	West Virginia	Wayne	Data Science	2019-05-19
10	120048	Robert	Cox	M	Virginia	Accomack	Back-End Developer	2019-06-07
11	120087	Bill	Tucker	M	Virginia	Halifax	Data Analysis	2019-06-16
12	130558	Olivia	Brown	F	West Virginia	Kanawha	Quality Assurance (QA)	2019-04-02
13	130560	Joseph	Lee	M	West Virginia	Mingo	Quality Assurance (QA)	2019-05-08
14	130646	Jack	Rogers	M	Virginia	Prince William	Data Science	2019-05-19

query :

```
1 SELECT COUNT(first_name)
2 FROM student_info;
3
```

output :

```
1 COUNT(first_name)
2 -----
3 32
4
```



COUNT Function



There is another special character returning the number of rows in a table. That is * character. Use it inside the COUNT function as **COUNT (*)**.

▶ COUNT Function



An important point for **COUNT(*)** function is that the result table includes **NULL**. If you want the number of non-null values, use the syntax:
COUNT(column_name).



▶ AS (Alias) Keyword

We can customize the column name or table name using **AS** keyword. AS is used to rename a column or table with an alias.

This is the syntax for aliasing a column name:

column_name [AS] alias_name

This is the syntax for aliasing a table name:

table_name [AS] alias_name

▶ AS (Alias) Keyword



PRO
TIP

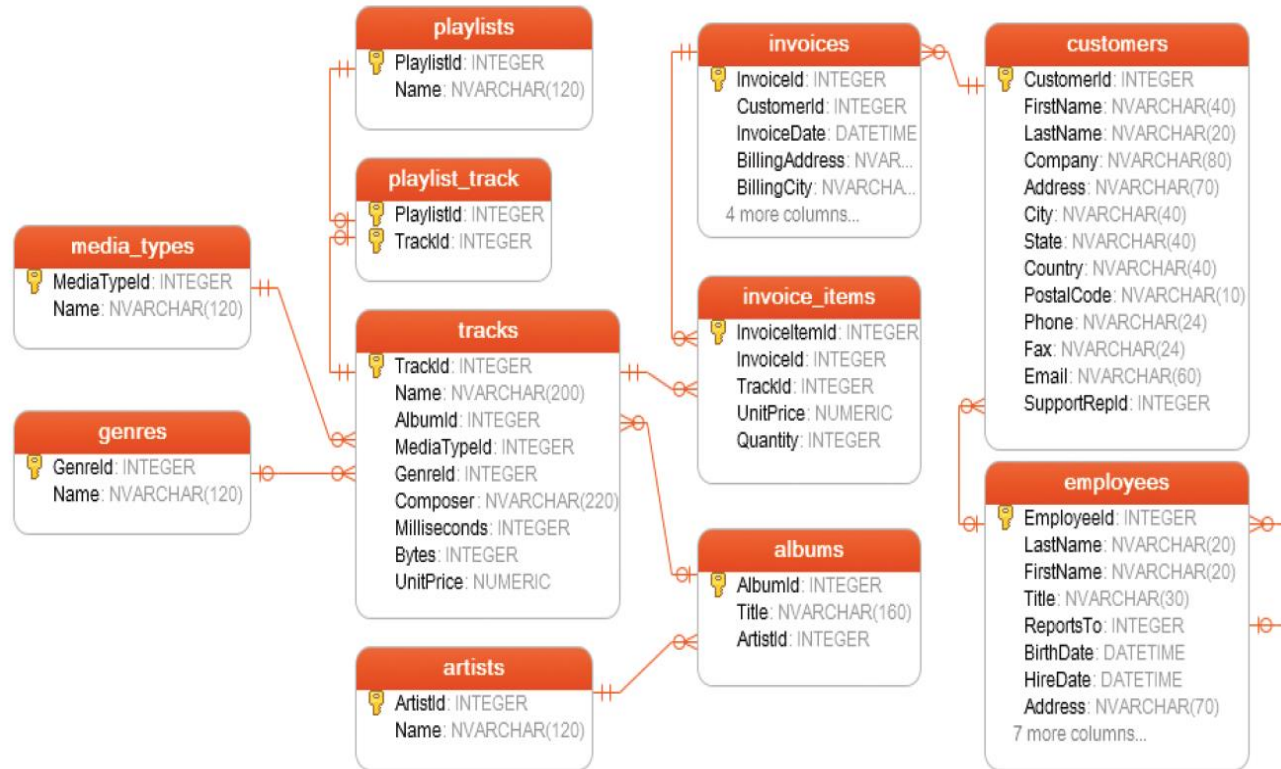
AS keyword is optional. Most programmers specify the AS keyword when aliasing a column name, but not when aliasing a table name.



Query Time



How many invoices are in the digital music store?





3

COUNT DISTINCT



COUNT DISTINCT



In some cases, we may want unique values. In those cases, we use **COUNT DISTINCT** function.

Syntax

COUNT (DISTINCT column_name)

COUNT DISTINCT



How many unique fields are there in the student_info table?

student_info table

	student_id	first_name	last_name	gender	state	county	field	start_date
1	110028	Michael	Crawford	M	Virginia	Albemarle	DevOps	2019-07-19
2	110078	Olivia	Smith	F	West Virginia	Tucker	Back-End Developer	2019-03-11
3	110080	Amelia	Anderson	F	West Virginia	Webster	Data Analysis	2019-04-25
4	110081	Megan	King	F	West Virginia	Kanawha	Back-End Developer	2019-06-07
5	110091	Richard	Morgan	M	Virginia	Prince William	DevOps	2019-06-28
6	110095	Hugo	Wallace	M	Virginia	Accomack	Data Analysis	2019-06-16
7	120001	Eleanor	Johnson	F	West Virginia	Wyoming	Front-End Developer	2019-03-22
8	120011	Oliver	Taylor	M	West Virginia	Hancock	Data Analysis	2019-04-14
9	120033	Lucas	Parker	M	West Virginia	Wayne	Data Science	2019-05-19
10	120048	Robert	Cox	M	Virginia	Accomack	Back-End Developer	2019-06-07
11	120087	Bill	Tucker	M	Virginia	Halifax	Data Analysis	2019-06-16
12	130558	Olivia	Brown	F	West Virginia	Kanawha	Quality Assurance (QA)	2019-04-02
13	130560	Joseph	Lee	M	West Virginia	Mingo	Quality Assurance (QA)	2019-05-08
14	130646	Jack	Rogers	M	Virginia	Prince William	Data Science	2019-05-19
15	130658	Emma	Powers	F	Virginia	Thomas	Back-End Developer	2019-07-21

input:

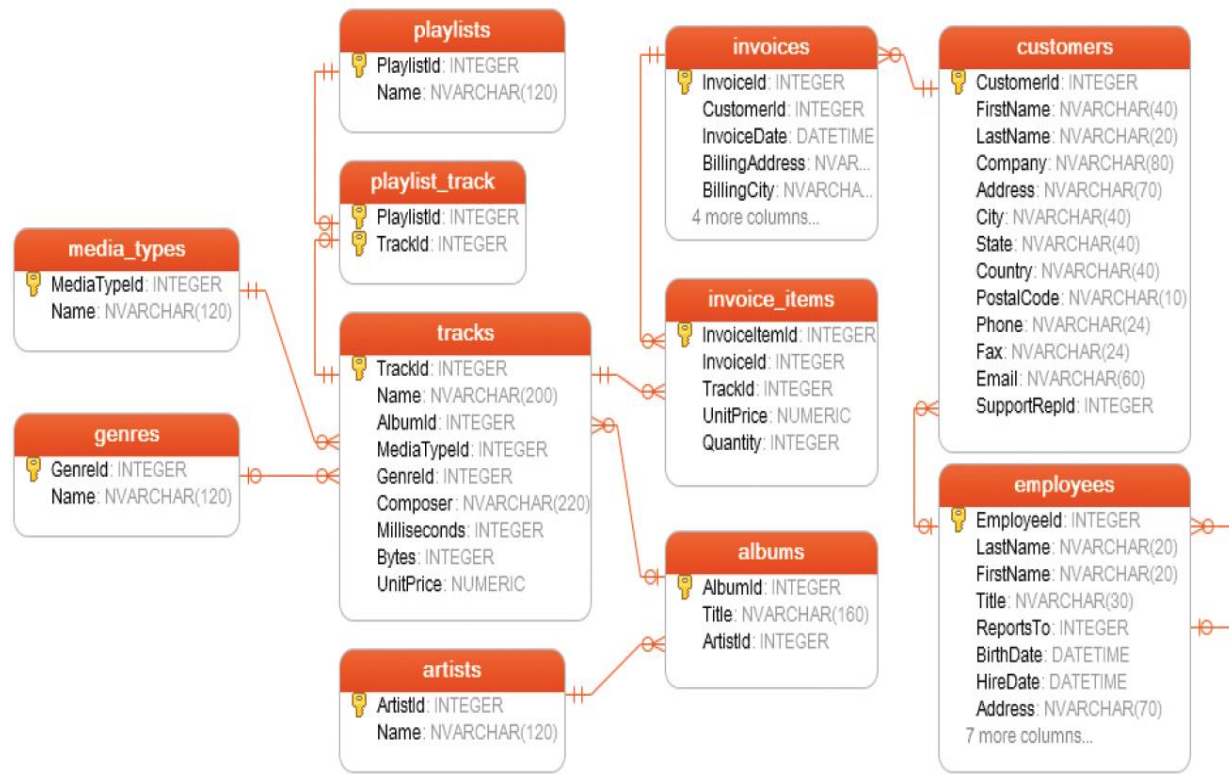
```
1 SELECT COUNT(DISTINCT field) AS count_of_field
2 FROM student_info;
3
```

output:

```
1 count_of_field
2 -----
3 6
4
```



How many composers are there in the digital music store?





MIN and MAX

MIN Function



MIN function returns the minimum value in the selected column. The **MIN** function ignores the **NULL** values.

Syntax

```
1 SELECT MIN(column_name)
2 FROM table_name;
3 |
```

MIN Function



Who gets paid the lowest wage in the company?

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

query :

```
1 SELECT MIN(salary) AS lowest_salary
2 FROM employees;
3
```

output :

```
1 lowest_salary
2 -----
3 55000
4
```

MIN Function



What is the earliest hired employees's date?

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

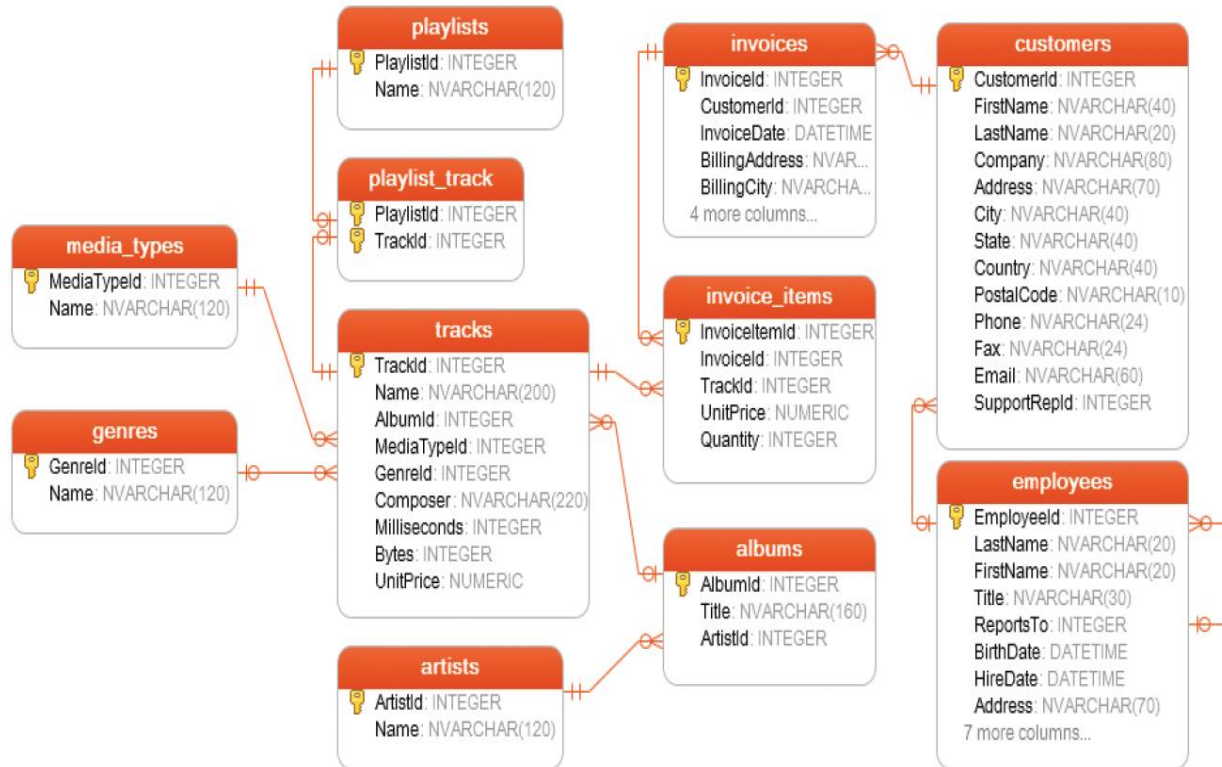
query :

```
1 SELECT MIN(hire_date) AS earliest_date
2 FROM employees;
3
```

output :

```
1 earliest_date
2 -----
3 2017-11-24
4
```


Find the track name having the minimum duration.





MAX Function



MAX function returns the maximum value in the selected column.

Syntax

```
1 SELECT MAX(column_name)
2 FROM table_name;
3 |
```



MAX Function



What is the the highest wage in the company?

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

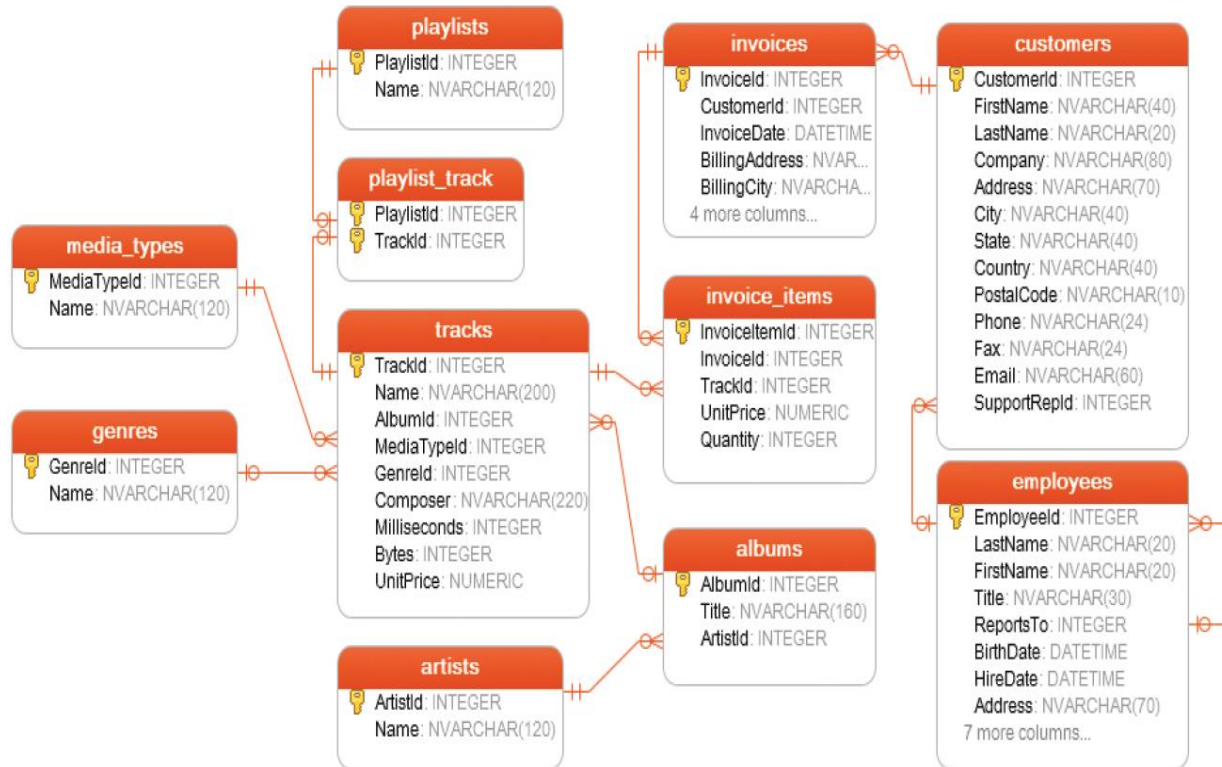
query :

```
1 SELECT MAX(salary) AS highest_salary
2 FROM employees;
3 |
```

output :

```
1 highest_salary
2 -----
3 110000
```

Find the track name having the maximum duration.





SUM and AVG

SUM Function



SUM function returns the sum of a numeric column.

Syntax

```
1 SELECT SUM(column_name)
2 FROM table_name;
3
```



SUM Function



What is total amount salary of the employees?

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

query:

```
1 SELECT SUM(salary) AS total_salary
2 FROM employees;
3
```

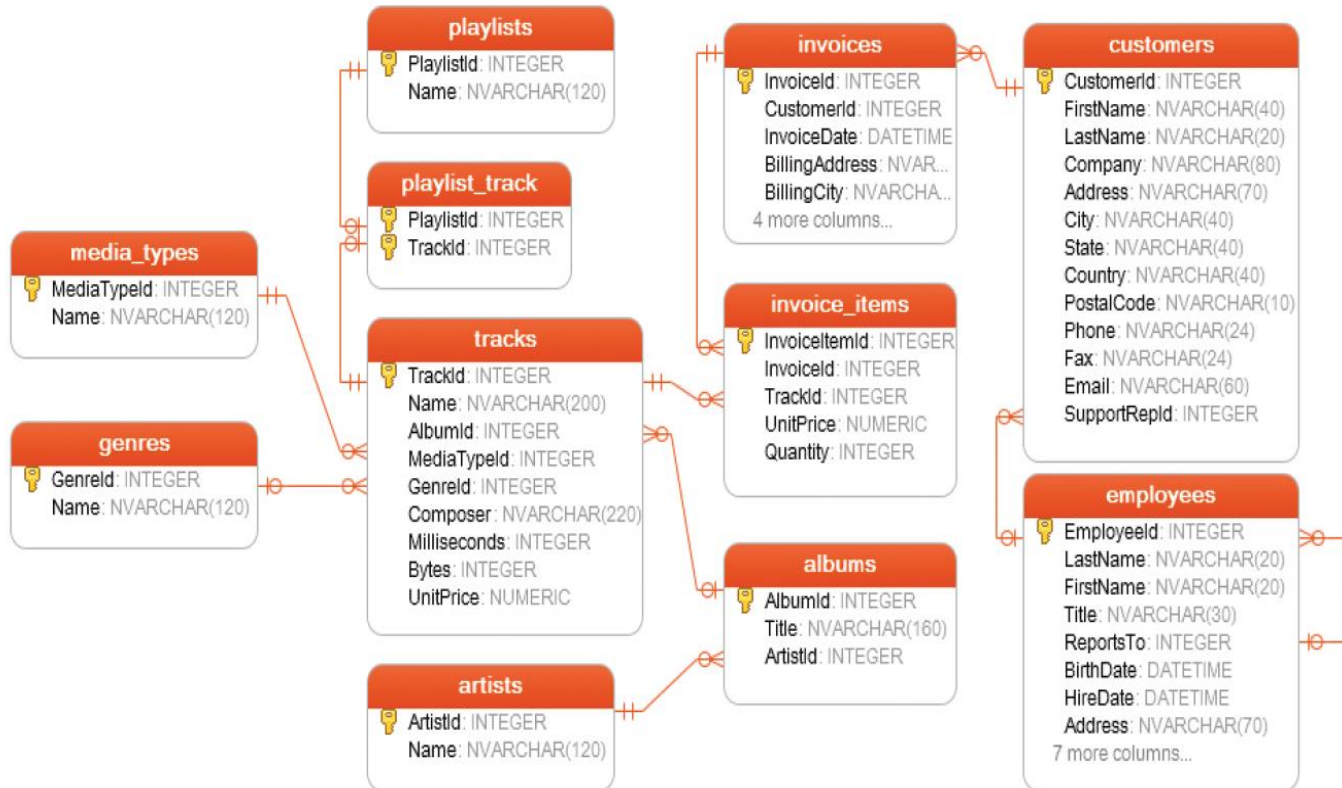
output:

```
1 total_salary
2 -----
3 836000
4
```

What is total amount salary of the male employees?

**SELECT SUM(salary) as male_salary
FROM employees
WHERE gender='Male'**

How much money did our store earn?





2 AVG Function

▶ AVG Function



AVG function calculates the average of a numeric column.

Syntax

```
1 SELECT AVG(column_name)
2 FROM table_name;
3
```

AVG Function



What is the average salary of the employees?

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

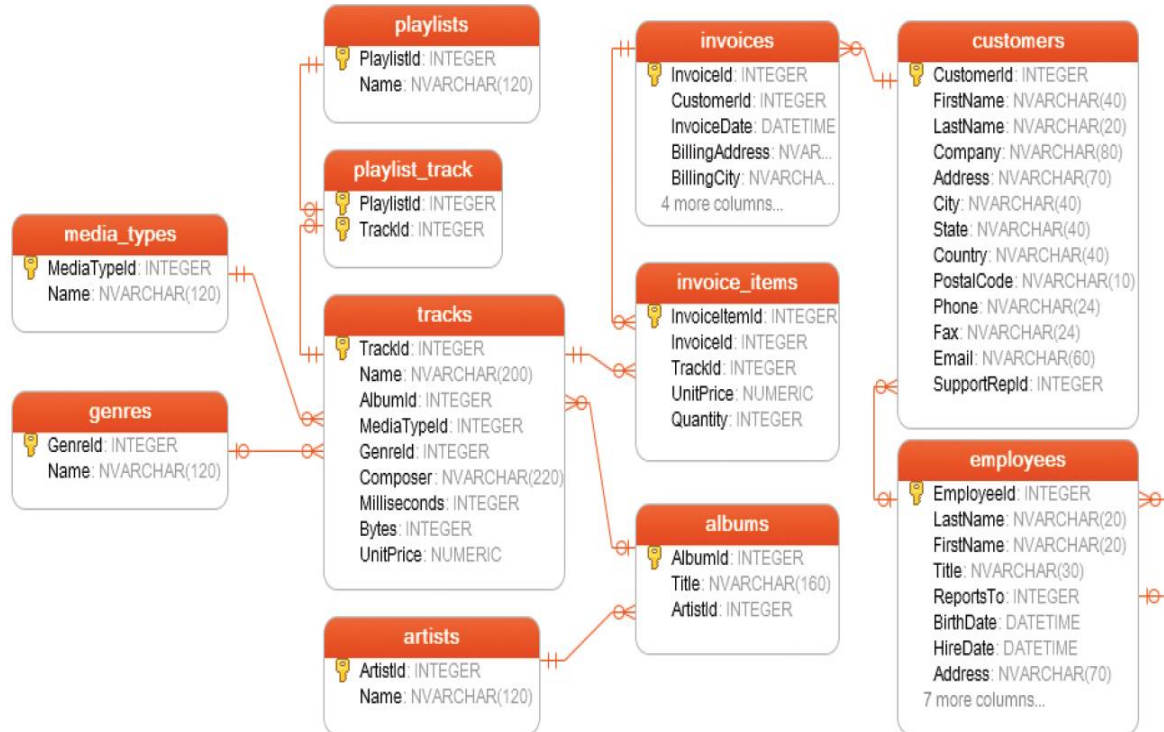
query:

```
1 SELECT AVG(salary) AS average_salary
2 FROM employees;
3 |
```

output:

```
1 average_salary
2 -----
3 83600.0
4 |
```

Find the tracks having duration bigger than the average duration.








1

GROUP BY Clause



GROUP BY Clause

The **GROUP BY** clause groups the rows into summary rows. It returns one value for each group and is typically used with aggregate functions (COUNT, MAX, MIN, SUM, AVG).

		Gender	COUNT(Gender)		4
		Male	COUNT(Gender) WHERE Gender = 'Male'		2
		Male			
		Female			
		Female	COUNT(Gender) WHERE Gender = 'Female'		2



GROUP BY Clause



Syntax

```
1 SELECT column_1, aggregate_function(column_2)
2 FROM table_name
3 GROUP BY column_1;
4 |
```

GROUP BY Clause



- GROUP BY returns only one result per group of data.
- GROUP BY Clause always follows the WHERE Clause.
- GROUP BY Clause always precedes the ORDER BY.



2

GROUP BY with COUNT Function

GROUP BY with COUNT Function



What is the number of employees per gender?

query:

```
1 SELECT gender, COUNT(gender)
2 FROM employees
3 GROUP BY gender;
4
```

output:

```
1 gender      COUNT(gender)
2 -----
3 Female      4
4 Male        6
5
```

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



GROUP BY Clause

The GROUP BY clause groups results before calling the aggregate function. This allows you to apply aggregate function to groups than the entire query.

gender
Male
Male
Male
Male
Male
Male
Female
Female
Female
Female

gender	COUNT(gender)
Male	6
Female	4

GROUP BY with COUNT Function



What is the number of employees working as a data scientist broken by gender?

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

query:

```
1 SELECT gender, COUNT(job_title)
2 FROM employees
3 WHERE job_title = 'Data Scientist'
4 GROUP BY gender;
5
```

output:

```
1 gender      COUNT(job_title)
2 -----
3 Female      1
4 Male        1
5
```



GROUP BY Clause



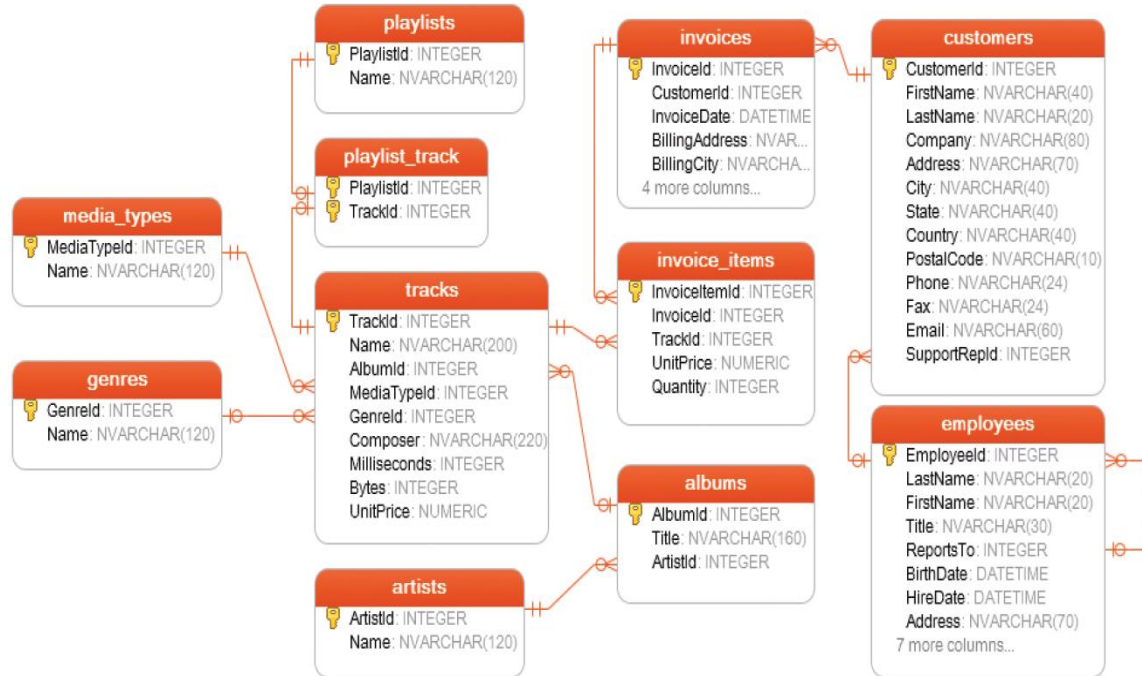
- WHERE clause operates on the data before the aggregation.
- WHERE clause happens before the GROUP BY clause.
- Only the rows that meet the conditions in the WHERE clause are grouped.



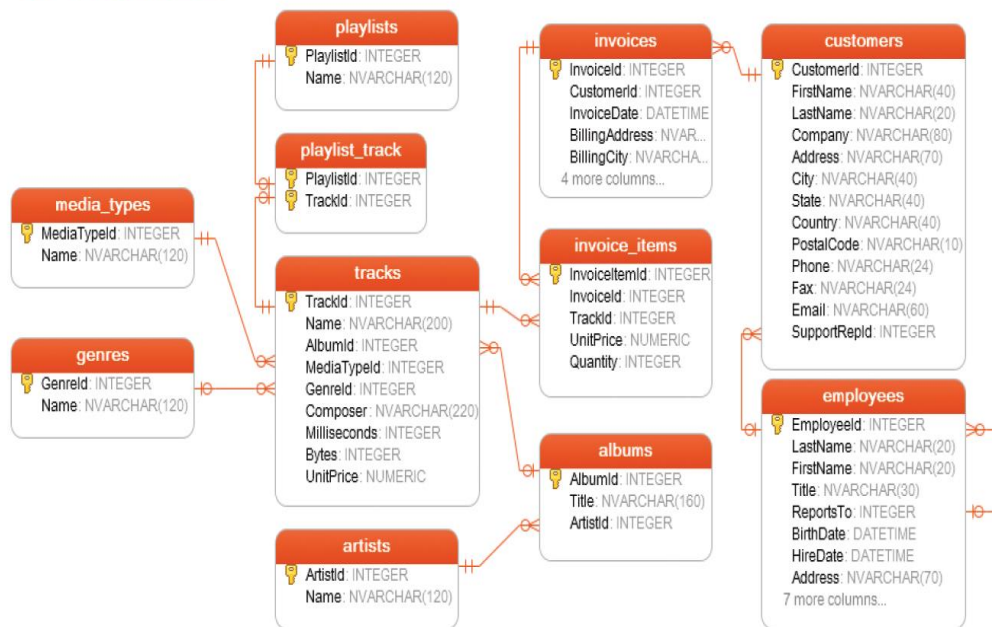
Query Time



**Find the total number of each composer's track.
Your result will include name of the composer
and number.**



How many customers do we have from each country? Your result will include name of the country and number.





3

GROUP BY with MIN&MAX Functions

GROUP BY with MIN&MAX Functions



Let's find the minimum salaries of each gender group using the **MIN** function.

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

query:

```
1 SELECT gender, MIN(salary) AS min_salary
2 FROM employees
3 GROUP BY gender;
```

output:

```
1 gender      min_salary
2 -----
3 Female      67000
4 Male        55000
5
```

GROUP BY with MIN&MAX Functions



Similarly, we can find the maximum salaries of each group using the **MAX** function. You may also use the **ORDER BY** clause to sort the salaries in descending or ascending order. The **ORDER BY** follows **GROUP BY**. For instance, sort the maximum salaries in descending order.

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

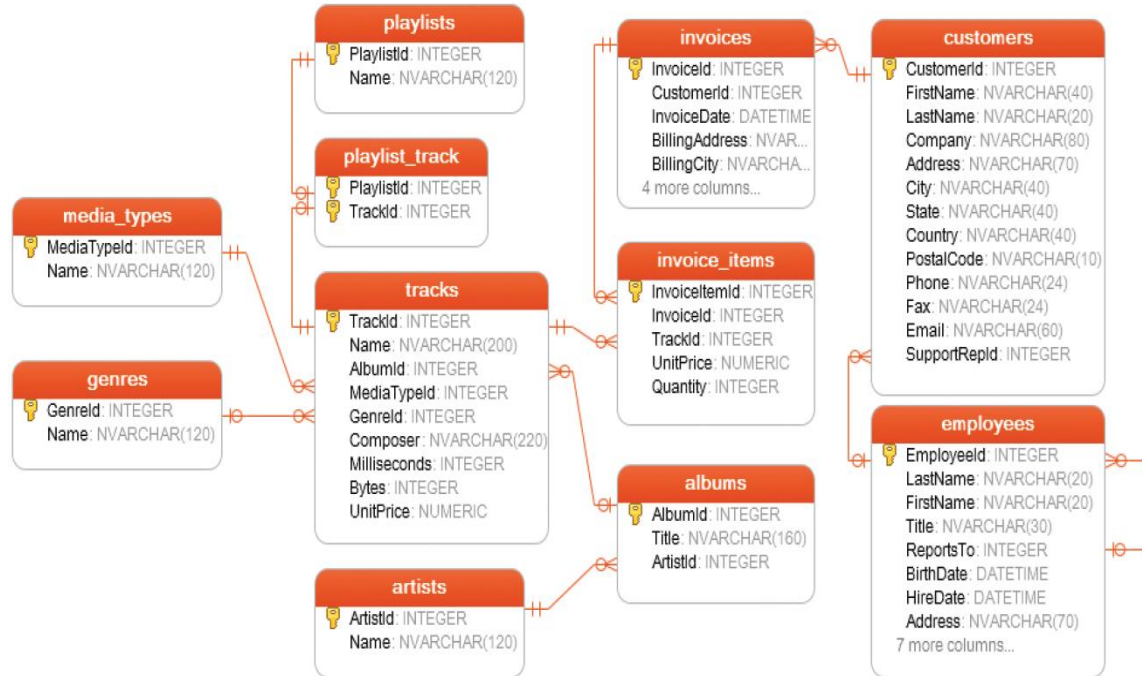
query:

```
1 SELECT gender,  
2 MAX(salary) AS max_salary  
3 FROM employees  
4 GROUP BY gender  
5 ORDER BY max_salary DESC;
```

output:

```
1 gender      max_salary  
2 -----  
3 Male        110000  
4 Female      95000  
5
```

Find the minimum duration of track for each album. Your result will include album id and min duration.





4

GROUP BY with SUM&AVG Functions

GROUP BY with SUM&AVG Functions

Let's calculate the total salaries of each group (gender).

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

query:

```
1 SELECT gender, SUM(salary) AS total_salary
2 FROM employees
3 GROUP BY gender;
```

output:

```
1 gender      total_salary
2 -----
3 Female      314000
4 Male        522000
5
```



GROUP BY with SUM&AVG Functions

Similarly, we can find the average salaries of each group using the **AVG** function.

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

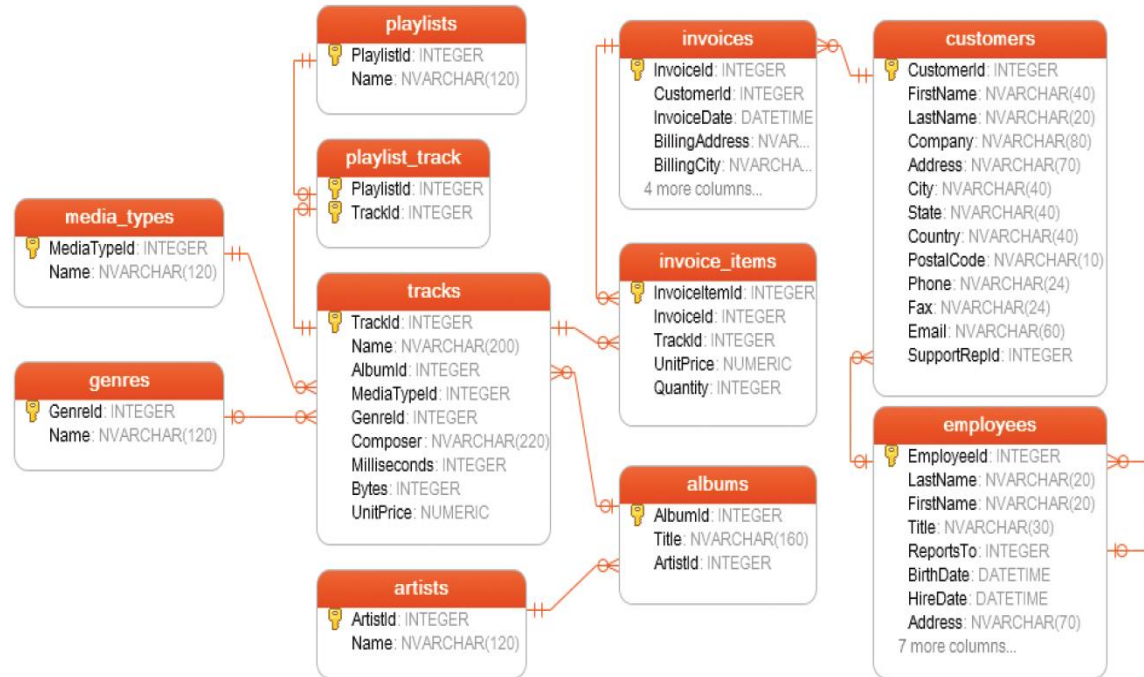
query:

```
1 SELECT gender, AVG(salary) AS average_salary
2 FROM employees
3 GROUP BY gender;
```

output:

```
1 gender      average_salary
2 -----
3 Female      78500.0
4 Male        87000.0
```

Find the total amount of invoice for each country. Your result will include country name and total amount.





JOINS

Table of Contents



- ▶ Introduction
- ▶ JOIN Types
- ▶ Inner JOIN
- ▶ Left JOIN



1 Introduction

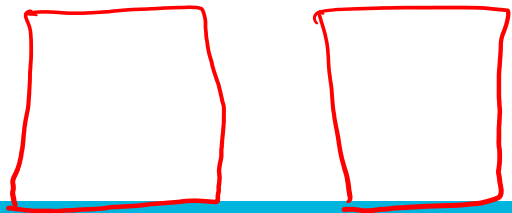
Introduction



A JOIN clause is used to combine two or more tables into a single table.

Joins are usually applied based on the keys that define the relationship between those tables or on common fields.

Introduction

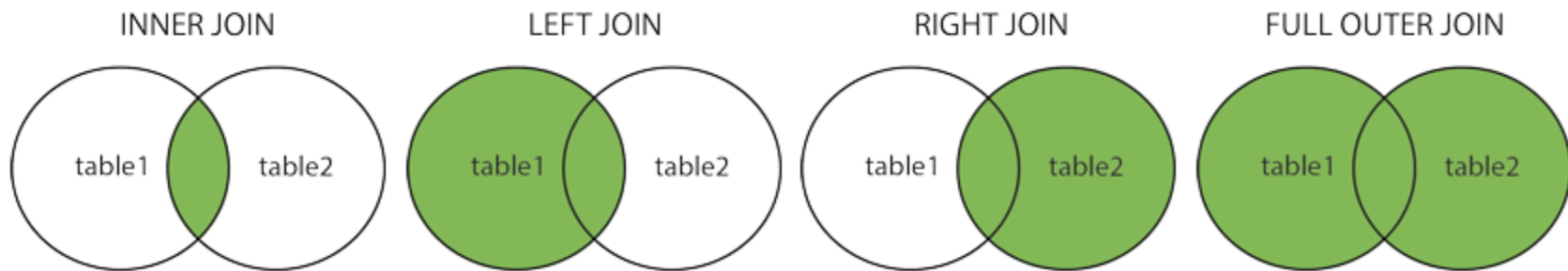


In most cases this joins are created using the primary key of one table and the foreign key of the other table we want to join it with.



2 JOIN Types

JOIN Types



- **INNER JOIN:** Returns the common records in both tables.
- **LEFT OUTER JOIN:** Returns all records from the left table and matching records from the right table.
- **RIGHT OUTER JOIN:** Returns all records from the right table and matching records from the left table.
- **FULL OUTER JOIN:** Returns all records of both left and right tables.



3

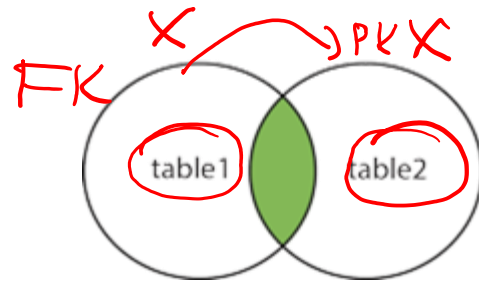
INNER JOIN

INNER JOIN

INNER JOIN is the most common type of JOINS. The INNER JOIN selects records that have matching values in both tables. INNER keyword is optional for this type of JOIN.

table1.salary

table1.*, table2.y **Syntax**



```
1 SELECT columns
2 FROM table_A
3 INNER JOIN table_B ON join_conditions
```

join_conditions

table_A.common_field = table_B.common_field

FK = table.B.PK



students

name	exam	score
John	SQL	75
Mary	AWS	80
Clark	Python	60

tests

exam	passing_score
SQL	70
AWS	80
Python	70
Network	60



```
SELECT students.name, students.exam,  
       students.score, tests.passing_score  
FROM students  
INNER JOIN tests ON students.exam = tests.exam;
```

Left

FK

PK

students

name	exam	score	P.S
John	SQL	75	70
Mary	AWS	80	80
Clark	Python	60	70

tests

inner join

exam	passing_score
SQL	70
AWS	80
Python	70
Network	60



```
SELECT students.name, students.exam,  
       students.score, tests.passing_score  
FROM students  
INNER JOIN tests ON students.exam = tests.exam;
```

students

tests

name	exam	score	exam	passing_score
John	SQL	75	SQL	70
Mary	AWS	80	AWS	80
Clark	Python	60	Python	70
			Network	60



```
SELECT students.name, students.exam,  
       students.score, tests.passing_score  
FROM students  
INNER JOIN tests ON students.exam = tests.exam;
```

students

tests

name	exam	score	exam	passing_score
John	SQL	75	SQL	70
Mary	AWS	80	AWS	80
Clark	Python	60	Python	70
			Network	60



```
SELECT students.name, students.exam,  
       students.score, tests.passing_score  
FROM students  
INNER JOIN tests ON students.exam = tests.exam;
```

students

tests

name	exam	score	exam	passing_score
John	SQL	75	SQL	70
Mary	AWS	80	AWS	80
Clark	Python	60	Python	70



```
SELECT students.name, students.exam,  
       students.score, tests.passing_score  
FROM students  
INNER JOIN tests ON students.exam = tests.exam;
```

output of the query

name	exam	score	passing_score
John	SQL	75	70
Mary	AWS	80	80
Clark	Python	60	70

INNER JOIN



Syntax of Join of Multiple Tables

```
1 SELECT columns
2   FROM table_A
3   INNER JOIN table_B
4     ON join_conditions1 AND join_conditions2
5   INNER JOIN table_C
6     ON join_conditions3 OR join_conditions4
7 ...|
```



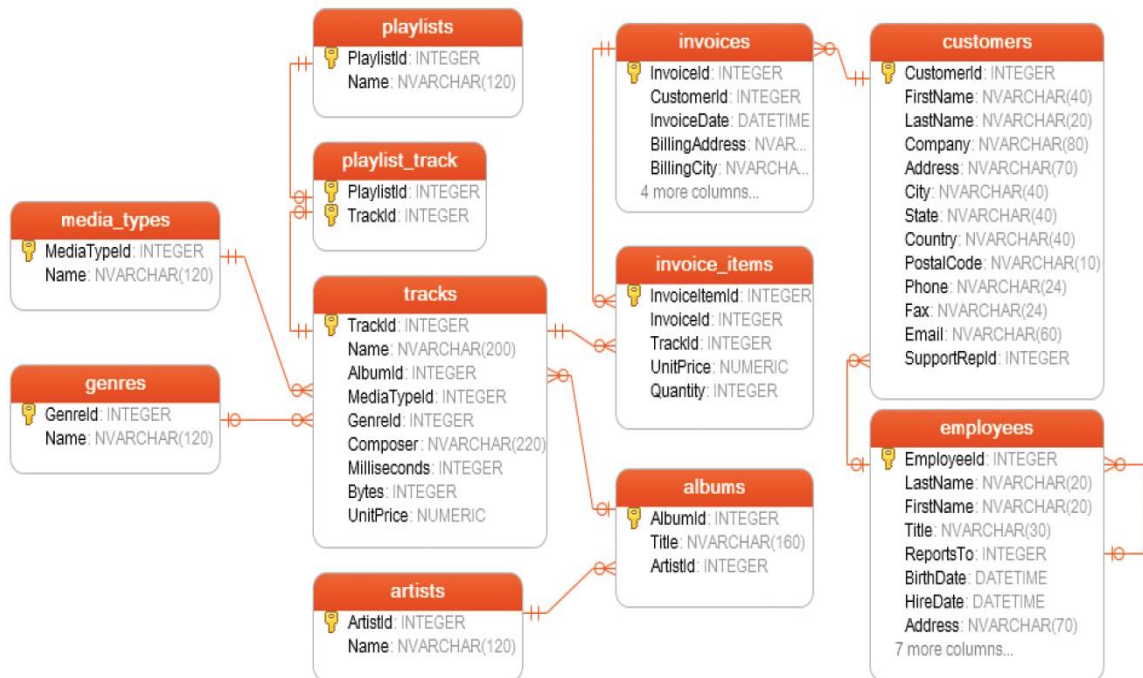

Important Concepts

Primary Key (PK):

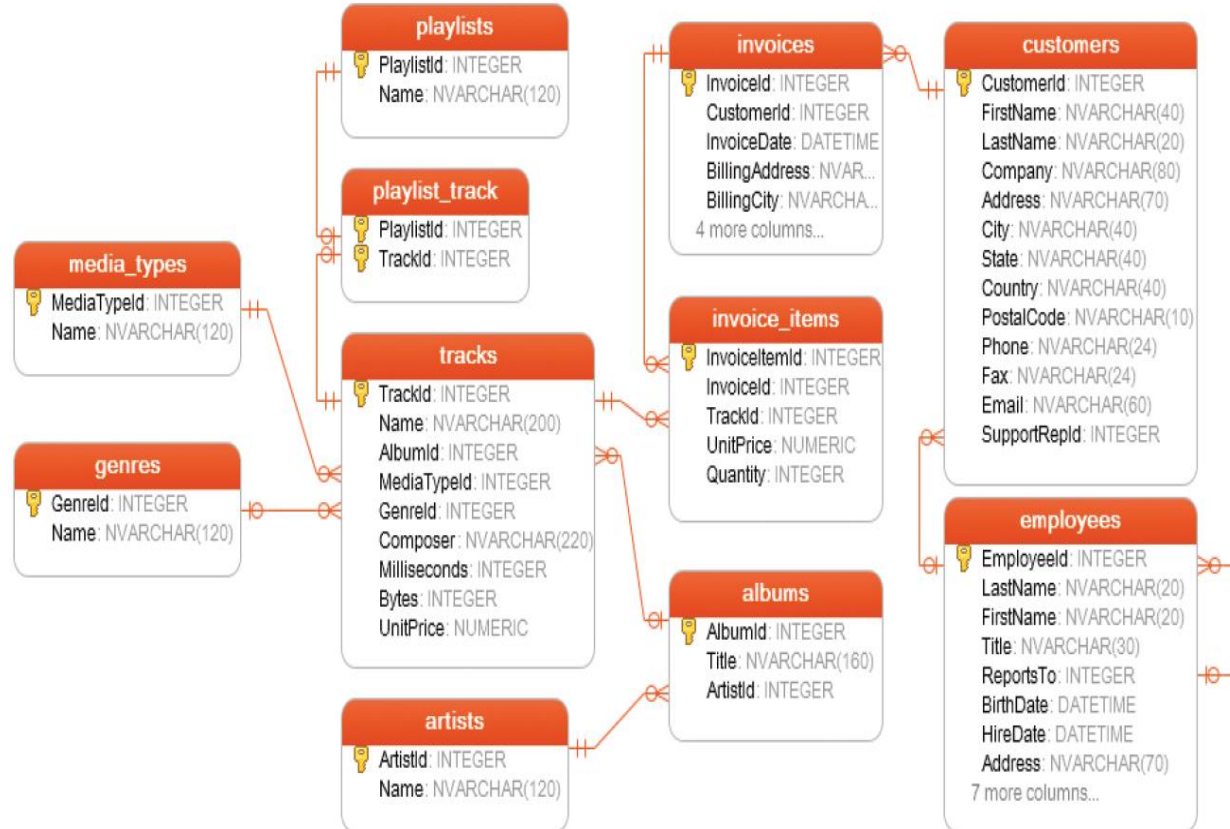
The primary key is a column in our table that makes each row (aka, record) unique.

Foreign Key (FK):

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.



Find the genre of each track.





tracks

	TrackId	Name	AlbumId	MediaTypeId	GenreId
1	1	For Those About To Rock (We Salute You)	1	1	1
2	2	Balls to the Wall	2	2	1
3	3	Fast As a Shark	3	2	1
4	4	Restless and Wild	3	2	1
5	5	Princess of the Dawn	3	2	1
6	6	Put The Finger On You	1	1	1
7	7	Let's Get It Up	1	1	1
8	8	Inject The Venom	1	1	1
9	9	Snowballed	1	1	1
10	10	Evil Walks	1	1	1

genres

	GenreId	Name
1	1	Rock
2	2	Jazz
3	3	Metal
4	4	Alternative & Punk
5	5	Rock And Roll
6	6	Blues
7	7	Latin
8	8	Reggae
9	9	Pop
10	10	Soundtrack



tracks

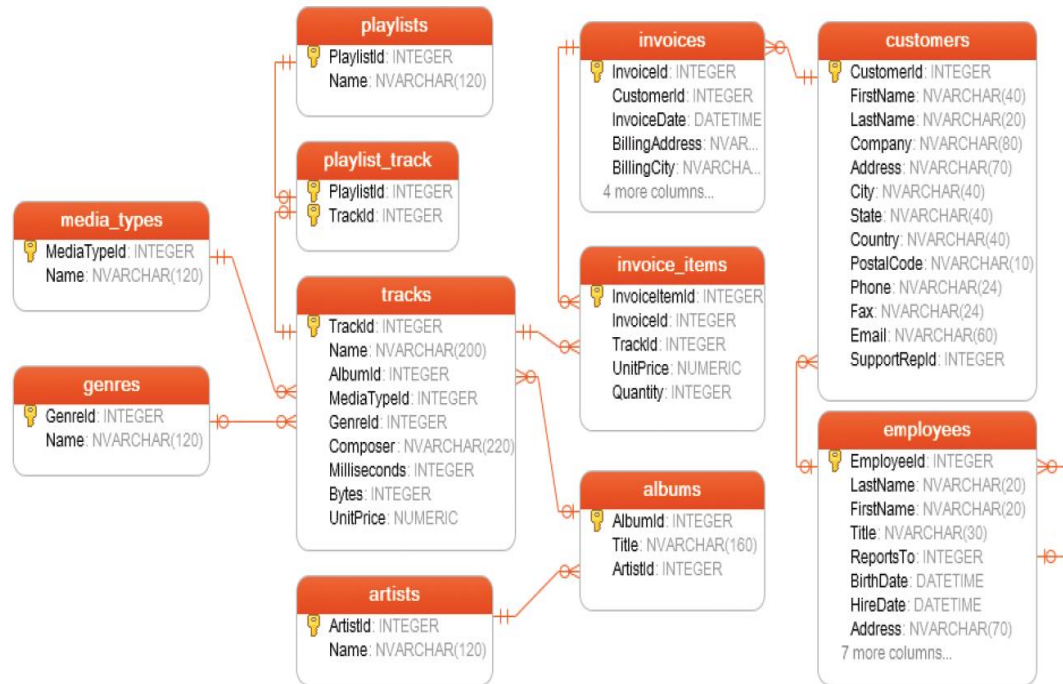
	TrackId	Name	AlbumId	MediaTypeId	GenreId
1	1	For Those About To Rock (We Salute You)	1	1	1
2	2	Balls to the Wall	2	2	1
3	3	Fast As a Shark	3	2	1
4	4	Restless and Wild	3	2	1
5	5	Princess of the Dawn	3	2	1
6	6	Put The Finger On You	1	1	1
7	7	Let's Get It Up	1	1	1
8	8	Inject The Venom	1	1	1
9	9	Snowballed	1	1	1
10	10	Evil Walks	1	1	1

genres

	GenreId	Name
1	1	Rock
2	2	Jazz
3	3	Metal
4	4	Alternative & Punk
5	5	Rock And Roll
6	6	Blues
7	7	Latin
8	8	Reggae
9	9	Pop
10	10	Soundtrack

	Name	Name
1	For Those About To Rock (We Salute You)	Rock
2	Balls to the Wall	Rock
3	Fast As a Shark	Rock
4	Restless and Wild	Rock
5	Princess of the Dawn	Rock
6	Put The Finger On You	Rock
7	Let's Get It Up	Rock
8	Inject The Venom	Rock
9	Snowballed	Rock
10	Evil Walks	Rock

**Find the customer name of each invoice.
Your result will include Invoice id and customer name.**



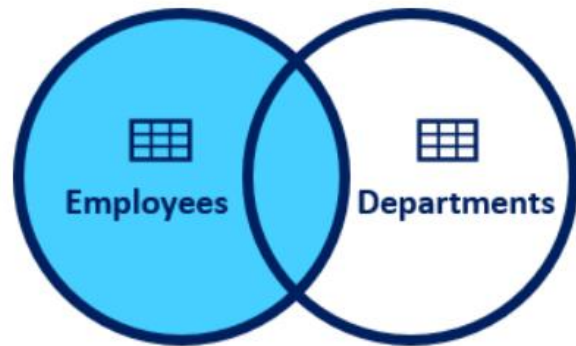


LEFT JOIN

LEFT JOIN



In this JOIN statement, all the records of the left table and the common records of the right table are returned in the query. If no matching rows are found in the right table during the JOIN operation, these values are assigned as NULL.



Visual Representation of Left JOIN

Syntax

```
1 SELECT columns
2 FROM table_A
3 LEFT JOIN table_B ON join_conditions
```

join_conditions

table_A.common_field = table_B.common_field



students

name	exam	score
John	SQL	75
Mary	AWS	80
Clark	Python	60

tests

exam	passing_score
SQL	70
AWS	80
Python	70
Network	60



```
SELECT tests.exam, tests.passing_score,  
       students.name, students.score  
FROM tests  
LEFT JOIN students ON tests.exam = students.exam;
```

tests

exam	passing_score
SQL	70
AWS	80
Python	70
Network	60

students

name	exam	score
John	SQL	75
Mary	AWS	80
Clark	Python	60



```
SELECT tests.exam, tests.passing_score,  
       students.name, students.score  
FROM tests  
LEFT JOIN students ON tests.exam = students.exam;
```

tests		students		
exam	passing_score	name	exam	score
SQL	70	John	SQL	75
AWS	80	Mary	AWS	80
Python	70	Clark	Python	60
Network	60			



```
SELECT tests.exam, tests.passing_score,  
       students.name, students.score  
FROM tests  
LEFT JOIN students ON tests.exam = students.exam;
```

tests		students		
exam	passing_score	name	exam	score
SQL	70	John	SQL	75
AWS	80	Mary	AWS	80
Python	70	Clark	Python	60
Network	60	Null	Null	Null



```
SELECT tests.exam, tests.passing_score,  
       students.name, students.score  
FROM tests  
LEFT JOIN students ON tests.exam = students.exam;
```

tests		students		
exam	passing_score	name	exam	score
SQL	70	John	SQL	75
AWS	80	Mary	AWS	80
Python	70	Clark	Python	60
Network	60	Null	Null	Null

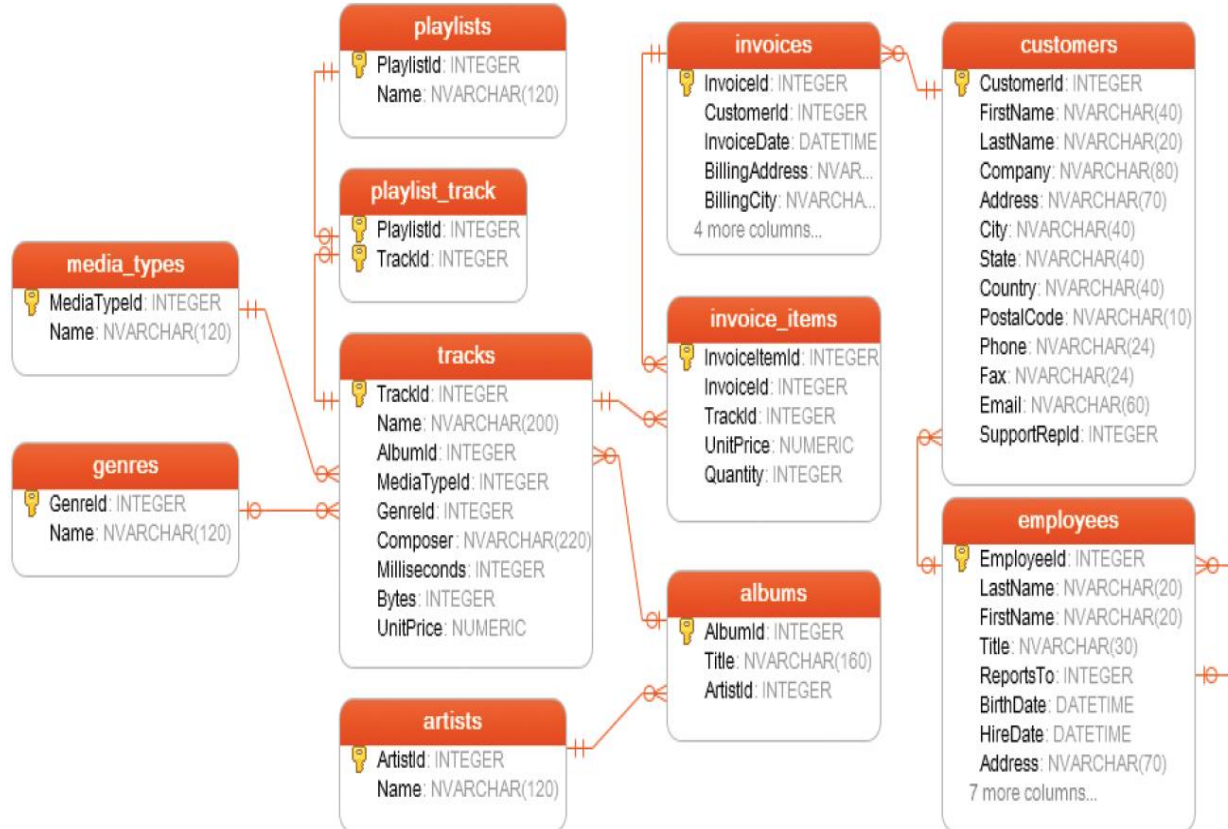


```
SELECT tests.exam, tests.passing_score,  
       students.name, students.score  
FROM tests  
LEFT JOIN students ON tests.exam = students.exam;
```

output of the query

exam	passing_score	name	score
SQL	70	John	75
AWS	80	Mary	80
Python	70	Clark	60
Network	60	Null	Null

Find the artists' album info





THANKS!

Any questions?





2 min Query Challenge:

How many distinct years in the invoices table?

Show your googling skills and build the query yourself